Towards nominal Abramsky

Andrzej S. Murawski[1] and Nikos Tzevelekos[2,⋆]

[1] University of Warwick
[2] Queen Mary, University of London

Abstract. Since the discovery of fully abstract models of PCF in the early 1990s, game semantics has expanded to a wide range of programming paradigms, covering effects like state, control, general references, non-determinism, probability and concurrency. Those models revealed an interesting phenomenon referred to as *Abramsky's cube*: starting from the PCF model and relaxing each of its combinatorial conditions, one was led to capture a corresponding impure effect. In this paper we initiate the construction of an analogous cube for nominal games, a strand of game semantics developed in the last ten years that incorporates *names* as semantic atoms and captures generative effects without using "bad-object" constructors. In particular, we examine the stateful axis of the cube: starting from games for higher-order references we move to full ground references, where strategies respect visibility, and from there to purely functional behaviour and innocent strategies.

Authors' note

Both authors met Samson at Oxford. Samson arrived there just in time to grill the first of us as a D.Phil. examiner and a little later started supervising the second on his MFoCS course.

Our life paths were closely intertwined afterwards. Andrzej joined the *Algorithmic Game Semantics* team, while Nikos started his D.Phil. with Samson, to be followed by a stint on the *Logic of Interaction and Information Flow* project.

Since Andrzej was supervised by Luke Ong, himself one of Samson's students, we belong to different generations of scientific offspring that can be traced back to Samson. Still, Nikos does not quite like when Andrzej calls him "uncle".

We are truly grateful to Samson for sound, reliable and insightful advice at various stages of our academic lives.

Happy Birthday, Samson!

1 Introduction

Game semantics emerged as a new semantic theory in the 1990s through the quest for a fully abstract model of PCF [AJM00,HO00]. In particular, models proposed by Hyland and Ong [HO00], and Nickau [Nic94] have brought to the fore a number of technical constraints on how games should be designed, such as innocence, bracketing and

⋆ Supported by a Royal Academy of Engineering research fellowship.

visibility. Abramsky suggested that relaxations of the constraints can lead to a whole hierarchy of models capturing a variety of impure computational features, a paradigm that came to be known as *Abramsky's cube*. The most immediate illustrations of this methodology were subsequent papers about state [AM97], control [Lai97] and general references [AHM98], in which violations of the above-mentioned three constraints were related precisely to three different programming features. The three directions could be viewed as three axes of a "cube", which has ever since grown to embrace, among others, polymorphism, non-determinism, probability and concurrency.

A related advance in the 2000s was the development of nominal game semantics [AGM$^+$04,Lai04,Tze09], which aimed to provide a more accurate account of generative effects such as references and, specifically, to target the bad-variable problem in game semantics. The problem stemmed from the fact that references were modelled as pairs of reading and writing methods, thus giving rise to objects of reference type whose behaviour need not be compatible with that of genuine reference cells. Nominal game models rely on collections of names that can be used throughout the play as part of a move. Operationally, they correspond to reference names in the implementation.

In this paper we follow the paradigm of Abramsky's cube in the nominal setting and uncover constraints corresponding respectively to the expressive power of general references (references to values of any type can be created), ground-type storage (only references to ground-type values, such as numbers or names, are allowed) and pure functional computation (no references can be created).

Our point of departure is our nominal model of RefML [MT11], which accounts for computation with general references. To study restrictions on the creation of references, we consider two sublanguages, called GrML and FunML respectively, which embody respectively ground-type storage and pure functional computation. The model from [MT11] provides a basic notion of a strategy and we set out to find restrictions corresponding to GrML and FunML. In non-nominal modelling, the borderline between general and ground-type storage could be captured by the visibility condition [AHM98]. Although this condition can be easily reintroduced in the nominal setting, it is not preserved by composition. To regain compositionality, we also need to eliminate the use of higher-order local state, that is, the generation of names of higher-order reference cells by programs. This results in a restriction that we call *groundness*, which provides a semantic match for GrML. To pass from ground-type storage to pure functional computation, we introduce a condition inspired by innocence. In order to make the course of play totally dependent on view, we forbid the creation of any names by programs and stipulate that copycat behaviour is followed whenever the environment requests the value of a reference whose name cannot be found in the current P-view.

The conditions of groundness and innocence are accompanied by factorisation results, as in the original case of the cube, but they are a little more involved due to complications that arise in the nominal setting. An elegant feature of the original cube was the fact that the essence of the passage from functional computation could be attributed to single representative strategies, corresponding respectively to a single integer-valued memory cell and a single higher-order cell. In the nominal setting, in addition to these strategies, one has to use families of name generators. Moreover, in order to pass to

$$\frac{}{\mathsf{u},\Gamma \vdash () : \mathsf{unit}} \qquad \frac{i \in \mathbb{Z}}{\mathsf{u},\Gamma \vdash i : \mathsf{int}} \qquad \frac{a \in (\mathsf{u} \cap \mathbb{A}_\theta)}{\mathsf{u},\Gamma \vdash a : \mathsf{ref}\,\theta} \qquad \frac{(x:\theta) \in \Gamma}{\mathsf{u},\Gamma \vdash x : \theta}$$

$$\frac{\mathsf{u},\Gamma \vdash M_1 : \mathsf{int} \qquad \mathsf{u},\Gamma \vdash M_2 : \mathsf{int}}{\mathsf{u},\Gamma \vdash M_1 \oplus M_2 : \mathsf{int}} \qquad \frac{\mathsf{u},\Gamma \vdash M : \mathsf{int} \quad \mathsf{u},\Gamma \vdash N_0 : \theta \quad \mathsf{u},\Gamma \vdash N_1 : \theta}{\mathsf{u},\Gamma \vdash \mathsf{if}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_0 : \theta}$$

$$\frac{\mathsf{u},\Gamma \vdash M : \mathsf{ref}\,\theta \quad \mathsf{u},\Gamma \vdash N : \theta}{\mathsf{u},\Gamma \vdash M := N : \mathsf{unit}} \qquad \frac{\mathsf{u},\Gamma \vdash M : \mathsf{ref}\,\theta}{\mathsf{u},\Gamma \vdash !M : \theta} \qquad \frac{\mathsf{u},\Gamma \vdash M : \mathsf{ref}\,\theta \quad \mathsf{u},\Gamma \vdash N : \mathsf{ref}\,\theta}{\mathsf{u},\Gamma \vdash M = N : \mathsf{int}}$$

$$\frac{\mathsf{u},\Gamma \vdash M : \theta}{\mathsf{u},\Gamma \vdash \mathsf{ref}_\theta(M) : \mathsf{ref}\,\theta} \qquad \frac{\mathsf{u},\Gamma \vdash M : \theta \to \theta' \quad \mathsf{u},\Gamma \vdash N : \theta}{\mathsf{u},\Gamma \vdash MN : \theta'} \qquad \frac{\mathsf{u},\Gamma \cup \{x:\theta\} \vdash M : \theta'}{\mathsf{u},\Gamma \vdash \lambda x^\theta.M : \theta \to \theta'}$$

Fig. 1. Syntax of RefML. $\oplus$ stands for binary integer functions, e.g. $+$, $-$, $*$, $=$.

innocence, it is necessary to provide a facility for storing an unbounded collection of names.

2 RefML

We start off by introducing the programming language RefML [MT11], which we shall work with throughout the paper. Its types are defined by the grammar below.

$$\theta, \theta' \ ::= \quad \mathsf{unit} \quad | \quad \mathsf{int} \quad | \quad \mathsf{ref}\,\theta \quad | \quad \theta \to \theta'$$

RefML is best described as the call-by-value λ-calculus over the ground types unit, int, and ref θ, augmented with basic commands (termination), primitives for integer arithmetic (constants, zero-test, binary integer functions) and higher-order reference manipulation (reference names, dereferencing, assignment, memory allocation, reference equality testing). The typing rules are given in Figure 1, where $\mathbb{A} = \biguplus_\theta \mathbb{A}_\theta$ stands for a countable set of *reference names* (one such set for each type θ), or just *names*, and u refers to a finite subset of $\mathbb{A}$.

Following standard conventions, we write $M; N$ for the term $(\lambda z^\theta.N)M$, where z does not occur in N and θ matches the type of M. let $x = M$ in N will stand for $(\lambda x^\theta.N)M$ in general. The *values* of the language are given by the syntax:

$$V ::= \ () \mid i \mid a \mid x \mid \lambda x^\theta.M.$$

To define the operational semantics of RefML, we introduce a syntactic notion of store. A *syntactic store* (or just *store*) will simply be a function from a finite set of names to values such that the type of each name matches the type of its assigned value. We write $S[a \mapsto V]$ for the store obtained by updating S so that a is mapped to V (this may extend the domain of S). Given a store S and a term M we say that the pair (S, M) is *compatible* if all names occurring in M are from the domain of S.

The small-step reduction rules are given as judgments of the shape $(S, M) \to (S', M')$, where (S, M), (S', M') are compatible and $\mathsf{dom}(S) \subseteq \mathsf{dom}(S')$. We present them in Figure 2, where we let a, b range over names. Evaluation contexts are given by

$$\begin{aligned} E \ ::= \ & (\lambda x.N)\,\_ \mid \_N \mid \_ \oplus N \mid i \oplus \_ \mid \_ = N \mid a = \_ \\ & \mid\ !\_ \mid \_ := N \mid a := \_ \mid \mathsf{ref}_\theta(\_) \mid \mathsf{if}\ \_\ \mathsf{then}\ N_1\ \mathsf{else}\ N_0. \end{aligned}$$

$$(S, \mathsf{if}\ 0\ \mathsf{then}\ N_1\ \mathsf{else}\ N_0) \to (S, N_0) \qquad (S, a = b) \to (S, 0)$$
$$(S, \mathsf{if}\ i\ \mathsf{then}\ N_1\ \mathsf{else}\ N_0) \to (S, N_1) \qquad (S, a = a) \to (S, 1)$$
$$(S, (\lambda x.M)V) \to (S, M[V/x]) \qquad (S, \mathsf{ref}_\theta(V)) \to (S[a' \mapsto V], a')$$
$$(S, !a) \to (S, S(a)) \qquad \frac{(S, M) \to (S', M')}{(S, E[M]) \to (S', E[M'])}$$
$$(S, a := V) \to (S[a \mapsto V], ())$$

Fig. 2. Small-step operational semantics of RefML (side-conditions: $i \neq 0, a \neq b, a' \notin \mathsf{dom}(S)$).

We say that (S, M) *evaluates* to (S', V) if $(S, M) \twoheadrightarrow (S', V)$, with V a value. For $\vdash M : \mathsf{unit}$ we say that M *converges*, written $M \Downarrow$, if $(\emptyset, M)$ evaluates to some $(S', ())$.

Example 1. It is well known that higher-order references are sufficiently powerful to enable one to define fixed-point combinators [AHM98] and, consequently, divergent terms Ω_θ at any type. Also, for any type θ, we define the terms new_θ by:

$$\mathsf{new}_{\mathsf{unit}} = \mathsf{ref}_{\mathsf{unit}}() \qquad \mathsf{new}_{\theta\to\theta'} = \mathsf{ref}_{\theta\to\theta'}(\lambda x^\theta.\Omega_{\theta'})$$
$$\mathsf{new}_{\mathsf{int}} = \mathsf{ref}_{\mathsf{int}}(0) \qquad \mathsf{new}_{\mathsf{ref}^i\,\theta''} = \mathsf{ref}\,(\mathsf{ref}\,(\cdots \mathsf{ref}\,(\mathsf{new}_{\theta''})))$$

where θ'' is one of $\mathsf{unit}, \mathsf{int}$ or a function type. These terms create new names and initialise them with default values.

Definition 2. *The term-in-context* $\Gamma \vdash M_1 : \theta$ **approximates** $\Gamma \vdash M_2 : \theta$ *(written* $\Gamma \vdash M_1 \sqsubseteq M_2$*) if* $C[M_1] \Downarrow$ *implies* $C[M_2] \Downarrow$ *for any context* $C[-]$ *such that* $\vdash C[M_1], C[M_2] : \mathsf{unit}$*. Two terms-in-context are* **equivalent** *if they approximate each other (written* $\Gamma \vdash M_1 \cong M_2$*).*

In the remainder of the paper we shall also discuss two specific fragments of RefML.

- GrML will comprise all RefML-terms in which all occurrences of $\mathsf{ref}_\theta(M)$ are restricted to non-functional types, i.e. θ cannot have the shape $\theta_1 \to \theta_2$. Thus, GrML is the sublanguage of RefML allowing for local storage of ground values only.
- FunML will consist of all RefML-terms that do not have any occurrences of $\mathsf{ref}_\theta(M)$. Thus, FunML is also a subset of GrML. FunML can be viewed as a purely functional fragment of RefML, since terms cannot create reference cells.

3 Game model

Here we review the game model of RefML first presented in [MT11]. Its distinctive feature is the presence of stores in moves and the possibility of justifying a move with a higher-order reference cell in addition to the standard option of justifying a move with another.

Formally, the model is constructed using mathematical objects (moves, plays, strategies) that feature names drawn from the set

$$\mathbb{A} = \biguplus_\theta \mathbb{A}_\theta\,.$$

$$
\begin{array}{l|l}
M_{A\otimes B} = (I_A \times I_B) \uplus \bar{I}_A \uplus \bar{I}_B & M_{A\Rightarrow B} = \{\star\} \uplus M_A \uplus M_B \\
I_{A\otimes B} = I_A \times I_B & I_{A\Rightarrow B} = \{\star\} \\
\lambda_{A\otimes B} = [(\mathrm{i}_A, \mathrm{i}_B) \mapsto PA,\ \lambda_A \restriction \bar{I}_A,\ \lambda_B \restriction \bar{I}_B] & \lambda_{A\Rightarrow B} = [\star \mapsto PA,\ \overline{\lambda}_A[\mathrm{i}_A \mapsto OQ],\ \lambda_B] \\
\vdash_{A\otimes B} = \{((\mathrm{i}_A, \mathrm{i}_B), m) \mid \mathrm{i}_A \vdash_A m \vee \mathrm{i}_B \vdash_B m\} & \vdash_{A\Rightarrow B} = \{(\star, \mathrm{i}_A)\} \cup \{(\mathrm{i}_A, \mathrm{i}_B)\} \cup \vdash_A \cup \vdash_B \\
\qquad \cup (\vdash_A \restriction \bar{I}_A{}^2) \cup (\vdash_B \restriction \bar{I}_B{}^2) &
\end{array}
$$

Fig. 3. Arena constructions. We write $\bar{I}_A = M_A \setminus I_A$.

Although names underpin various elements of our model, we do not want to delve into the precise nature of the sets containing them. Hence, all of our definitions preserve name-invariance, i.e. our objects are (strong) *nominal sets* [GP02,Tze09]. Note that we do not need the full power of the theory but mainly the basic notion of name-permutation. Here permutations are bijections $\pi : \mathbb{A} \to \mathbb{A}$ with finite support which respect the indexing of name-sets. For an element x belonging to a (nominal) set X we write $\nu(x)$ for its name-support, which is the set of names occurring in x. Moreover, for any $x, y \in X$, we write $x \sim y$ if there is a permutation π such that $x = \pi \cdot y$.

Our model is couched in the Honda-Yoshida style of modelling call-by-value computation [HY99]. Before we define what it means to play, we introduce the auxiliary concept of an arena.

Definition 3. *An* arena $A = \langle M_A, I_A, \lambda_A, \vdash_A \rangle$ *is given by:*

- *a set of moves M_A and a subset $I_A \subseteq M_A$ of initial ones,*
- *a labelling function $\lambda_A : M_A \to \{O, P\} \times \{Q, A\}$,*
- *a justification relation $\vdash_A \subseteq M_A \times (M_A \setminus I_A)$;*

satisfying, for each $m, m' \in M_A$, the conditions (π_i is the ith projection function):

- $m \in I_A \implies \lambda_A(m) = (P, A)$,
- $m \vdash_A m' \wedge \pi_2(\lambda_A(m)) = A \implies \pi_2(\lambda_A(m')) = Q$,
- $m \vdash_A m' \implies \pi_1(\lambda_A(m)) \neq \pi_1(\lambda_A(m'))$.

We range over moves by m, n and use i, o, p to refer to initial moves, O-moves and P-moves respectively. We let $\overline{\lambda}_A$ be the OP-complement of λ_A. Using the $\otimes$ and $\Rightarrow$ constructions on arenas (Fig. 3), for each type θ we define the corresponding arena $[\![\theta]\!]$, starting from the following definitions.

$$
\begin{array}{rr}
[\![\mathsf{unit}]\!] = \langle \{\star\}, \{\star\}, \emptyset, \emptyset \rangle & [\![\mathsf{int}]\!] = \langle \mathbb{Z}, \mathbb{Z}, \emptyset, \emptyset \rangle \\
[\![\mathsf{ref}\ \theta]\!] = \langle \mathbb{A}_\theta, \mathbb{A}_\theta, \emptyset, \emptyset \rangle & [\![\theta \to \theta']\!] = [\![\theta]\!] \Rightarrow [\![\theta']\!]
\end{array}
$$

We write 1 for $[\![\mathsf{unit}]\!]$, $\mathbb{Z}$ for $[\![\mathsf{int}]\!]$, and $\mathbb{A}_\theta$ for $[\![\mathsf{ref}\ \theta]\!]$; and set $M_\phi = \biguplus_{\theta,\theta'} M_{[\![\theta \to \theta']\!]}$.

Although types are interpreted by arenas, the actual games will be played in *prearenas*, which are defined in the same way as arenas with the exception that initial moves are O-questions. Given arenas A, B we define the prearena $A \to B$ as follows.

$$
\begin{array}{rr}
M_{A\to B} = M_A \uplus M_B & \lambda_{A\to B} = [\overline{\lambda}_A[\mathrm{i}_A \mapsto OQ], \lambda_B] \\
I_{A\to B} = I_A & \vdash_{A\to B} = \{(\mathrm{i}_A, \mathrm{i}_B)\} \cup \vdash_A \cup \vdash_B
\end{array}
$$

Our plays shall feature moves attached with stores, where the names appearing in a play take values. We let the set Val_θ of semantic values of type θ be $I_{[\![\theta]\!]}$ (so $\mathsf{Val}_{\mathsf{unit}} = \mathsf{Val}_{\theta\to\theta'} = \{\star\}$, $\mathsf{Val}_{\mathsf{int}} = \mathbb{Z}$, $\mathsf{Val}_{\mathsf{ref}\,\theta} = \mathbb{A}_\theta$), and let $\mathsf{Val} = \biguplus_\theta \mathsf{Val}_\theta$. A ***store*** Σ is a type-preserving finite partial function from $\mathbb{A}$ to Val, and Sto is the set of all stores:

$$\mathsf{Sto} = \{\, \Sigma : \mathbb{A} \rightharpoonup \mathsf{Val} \mid |\Sigma| \text{ finite} \wedge (a \in \mathsf{dom}(\Sigma) \cap \mathbb{A}_\theta \implies \Sigma(a) \in \mathsf{Val}_\theta) \,\}.$$

A move-with-store on a (pre)arena A is a pair m^Σ with $m \in M_A$ and $\Sigma \in \mathsf{Sto}$.

Definition 4. *A* justified sequence *on a prearena A is a sequence of moves-with-store from $M_A \uplus M_\phi$ such that, apart from the first move which must be of the form i^Σ with $\mathrm{i} \in I_A$, every move in s is equipped with a pointer to an earlier move, or to a name inside the store of an earlier move. These pointers are called* justification pointers *and are subject to the following constraints.*

- *If n^T points to m^Σ then either $m, n \in M_A$ and $m \vdash_A n$, or $m, n \in M_{\theta\to\theta'}$ for some θ, θ' and $m \vdash_{[\![\theta\to\theta']\!]} n$. We say that m^Σ* justifies *n^T.*
- *If n^T points to $a \in \mathsf{dom}(\Sigma)$ of m^Σ then $a \in \mathbb{A}_{\theta\to\theta'}$ for some θ, θ', and n must be an initial question in $M_{[\![\theta\to\theta']\!]}$. We say that m^Σ a-*justifies *n^T.*

An intuitive way to comprehend pointers to a name $a \in \mathsf{dom}(\Sigma) \cap \mathbb{A}_{\theta\to\theta'}$ is to think of them as pointing to the value $\star$ of a stored in Σ. Since the value of a is of function type, its structure is not revealed at once, but it can be explored by players by invoking the function, that is, by playing in $[\![\theta \to \theta']\!]$ from that initial $\star$.

Note that a justified sequence on A contains moves from M_A, called *A-moves*, and moves from M_ϕ, which hereditarily point inside stores of other moves. The latter are called *ϕ-moves*. We shall say that m^Σ is an *ancestor* of n^T (or that n^T is a descendant of m^Σ) if there is a chain of pointers from n^T to m, possibly passing through stores on the way. Similarly, we say that m^Σ is an *a-ancestor* of n^T (or that n^T is an a-descendant of m^Σ) if there is a chain of pointers from n^T to a in Σ (the chain may also be visiting other stores). Note that each ϕ-move has a unique a-ancestor from M_A.

For each $S \subseteq \mathbb{A}$ and Σ we define the closure of S under Σ as $\Sigma^*(S) = \bigcup_i \Sigma^i(S)$, where $\Sigma^0(S) = S$ and $\Sigma^{i+1}(S) = \Sigma(\Sigma^i(S)) \cap \mathbb{A}$. The set of *available names* of a justified sequence is defined inductively by $\mathsf{Av}(\epsilon) = \emptyset$ and

$$\mathsf{Av}(sn^T) = \begin{cases} \mathsf{Av}(s) & \text{there is an } a\text{-ancestor } m^\Sigma \text{ of } n^T \text{ and } a \notin \mathsf{Av}(s_{\leq m^\Sigma}) \\ \Sigma^*(\mathsf{Av}(s) \cup \nu(n)) & \text{otherwise} \end{cases}$$

where $s_{\leq m^\Sigma}$ is the subsequence of s up to m^Σ. We shall be writing $s \sqsubseteq s'$ to mean that s is a subsequence of s'.

Definition 5. *Let A be a prearena. A justified sequence s on A is called a* **legal sequence***, written $s \in L_A$, if it satisfies the conditions below.*

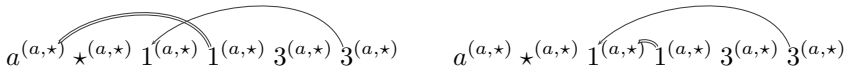
- *No adjacent moves belong to the same player, and no move points to a move (or the store of a move) of the same player (*Alternation*).*
- *The justifier of each answer is the most recent unanswered question (*Bracketing*).*

We call s a **play** *if it additionally satisfies:*

- *For any $s'm^{\Sigma} \sqsubseteq s$, $\mathsf{dom}(\Sigma) = \mathsf{Av}(s'm^{\Sigma})$ (*Frugality*).*

We write P_A for the set of plays on A.

Example 6. Here are two plays on $\mathbb{A}_{\mathsf{int}\to\mathsf{int}} \to \mathbb{Z} \Rightarrow \mathbb{Z}$ (for the sake of clarity, we omit pointers that would just point at preceding moves). We use double-line pointers to highlight the justification pointers pointing at stores.



The plays will be among those used to interpret the respective terms:

$$x : \mathsf{ref}\,(\mathsf{int}\to\mathsf{int}) \vdash\, !x : \mathsf{int}\to\mathsf{int} \qquad x : \mathsf{ref}\,(\mathsf{int}\to\mathsf{int}) \vdash \lambda h^{\mathsf{int}}.(!x)h : \mathsf{int}\to\mathsf{int}$$

Each name appearing in a legal sequence s, i.e. such that $a \in \nu(s)$, is called a *P-name* of s, written $a \in P(s)$, if it is first introduced in s by a P-move, that is, there is even-length $s'm^{\Sigma} \sqsubseteq s$ such that $a \in \nu(m^{\Sigma}) \setminus \nu(s')$. The set of *$O$-names* of s, $O(s)$, is defined dually. Clearly, $\nu(s) = O(s) \uplus P(s)$. Moreover, let us define γ to be the canonical function on justified sequences which imposes frugality by deleting unavailable names from store-domains and all ϕ-moves that they hereditarily justify. Concretely, $\gamma(\epsilon) = \epsilon$ and:

$$\gamma(sn^{T}) = \begin{cases} \gamma(s) & \text{if there is an } a\text{-ancestor } m^{\Sigma} \text{ of } n^{T} \text{ and } a \notin \mathsf{Av}(s_{\leq m^{\Sigma}}); \\ \gamma(s)\; n^{T\restriction \mathsf{Av}(sn^{T})} & \text{otherwise.} \end{cases}$$

Definition 7. *A* **strategy** *σ on a prearena A, written $\sigma : A$, is a set of even-length plays of A satisfying:*

- *If $so^{\Sigma}p^{\Sigma'} \in \sigma$ then $s \in \sigma$ (*Even-prefix closure*).*
- *If $s \in \sigma$ and $s \sim t$ then $t \in \sigma$ (*Equivariance*).*
- *If $s_1p_1^{\Sigma_1}, s_2p_2^{\Sigma_2} \in \sigma$ and $s_1 \sim s_2$ then $s_1p_1^{\Sigma_1} \sim s_2p_2^{\Sigma_2}$ (*Nominal determinacy*).*

Example 8. For each arena A there is an *identity strategy*, $\mathsf{id}_A : A \to A$, defined by

$$\mathsf{id}_A = \{\, s \in P^{even}_{A\to A} \mid \forall s' \sqsubseteq^{even} s.\; s' \upharpoonright A_l = s' \upharpoonright A_r \,\},$$

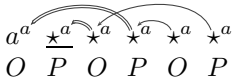
where the indices l, r distinguish the two copies of A, and $s' \upharpoonright A_x$ is the subsequence of s' containing only moves from the x-copy, along with all ϕ-moves having a-ancestors from the x-copy (for some a).

The behaviour of id_A is called *copycat*. More generally, we say that moves $n^{T}n'^{T'}$ in a play s are a *copycat pair* if they are consecutive in s, $n^{T} = n'^{T'}$, and if n^{T} is justified by $m'^{\Sigma'}$ (or by some $a \in \mathsf{dom}(\Sigma')$) then $n'^{T'}$ is justified by m^{Σ} (resp. by $a \in \mathsf{dom}(\Sigma)$) where $m^{\Sigma}m'^{\Sigma'}$ are consecutive in s. It will be useful to spot copycat behaviours occurring in plays exclusively between ϕ-moves with consecutive a-ancestors.

Definition 9. *Let s be an alternating justified sequence in A, $s' \sqsubseteq s$ be ending in $m^{\Sigma} m'^{\Sigma'}$ and let $a \in \mathsf{dom}(\Sigma) \cap \mathsf{dom}(\Sigma') \cap \mathbb{A}_\phi$ such that $m'^{\Sigma'}$ is not a-justified by m^{Σ}. We say that (s, s', a) is a* **copycat triple** *if, for all ϕ-moves n^T in s which have m^{Σ} or $m'^{\Sigma'}$ as an a-ancestor,*

- *if n has the same polarity as m then there is $n'^{T'}$ such that $n^T n'^{T'}$ are a copycat pair,*
- *if n has the same polarity as m' then there is $n'^{T'}$ such that $n'^{T'} n^T$ are a copycat pair.*

Example 10. We will be economical when writing stores and, in particular, components of the form $(a, \star)$ will often be written simply as a. Also, we will omit mentioning the empty play from strategies. Copycat behaviour is exemplified in the strategy $\sigma : \mathbb{A}_{\mathsf{unit} \to \mathsf{unit}} \to 1 = \{a^a \star^a s\}$ where each $(a^a \star^a s, a^a \star^a, a)$ is a copycat triple. For example, the play



is in σ. The copycat behaviour means that, when P plays its first move $\star^a$ (underlined), he does not change the value of a in the store. Thus, subsequent questions by O pointing to a in $\star^a$ are responded to by copycat. The strategy turns out to be the denotation of $x : \mathsf{ref}\,(\mathsf{unit} \to \mathsf{unit}) \vdash () : \mathsf{unit}$.

We now turn to defining a suitable notion of interaction between plays. Given arenas A, B, C, we define the prearena $A \to B \to C$ by:

$$M_{A \to B \to C} = M_{A \to B} \uplus M_C \quad \lambda_{A \to B \to C} = [\lambda_{A \to B}[\mathrm{i}_B \mapsto PQ], \overline{\lambda}_C]$$
$$I_{A \to B \to C} = I_A \qquad \vdash_{A \to B \to C} = \vdash_{A \to B} \cup \{(\mathrm{i}_B, \mathrm{i}_C)\} \cup \vdash_C$$

Let u be a justified sequence on $A \to B \to C$. We define $u \upharpoonright AB$ to be u in which all C-moves are suppressed, along with associated pointers and all ϕ-moves which are a-descendants of C-moves. $u \upharpoonright BC$ is defined analogously. $u \upharpoonright AC$ is defined similarly with the caveat that, if there was a pointer from a C-move to a B-move which in turn had a pointer to an A-move, we add a pointer from the C-move to the A-move. Let us write $u \upharpoonright_\gamma X$ for $\gamma(u \upharpoonright X)$ with $X \in \{AB, BC, AC\}$. Below we shall often say that a move is an O- or a P-move in X meaning ownership in the associated prearena.

Definition 11. *A justified sequence u on $A \to B \to C$ is an* **interaction sequence** *on A, B, C if it satisfies bracketing and frugality and, for all $X \in \{AB, BC, AC\}$, we have $(u \upharpoonright X) \in L_X$ and the following conditions hold.*

- $P(u \upharpoonright_\gamma AB) \cap P(u \upharpoonright_\gamma BC) = \emptyset$;
- $O(u \upharpoonright_\gamma AC) \cap (P(u \upharpoonright_\gamma AB) \cup P(u \upharpoonright_\gamma BC)) = \emptyset$;
- *For each $u' \sqsubseteq u$ ending in $m^{\Sigma} m'^{\Sigma'}$ and $a \in \mathsf{dom}(\Sigma')$ if*
  - *m' is a P-move in AB and $a \notin \mathsf{Av}(u' \upharpoonright AB)$,*
  - *or m' is a P-move in BC and $a \notin \mathsf{Av}(u' \upharpoonright BC)$,*
  - *or m' is an O-move in AC and $a \notin \mathsf{Av}(u' \upharpoonright AC)$,*

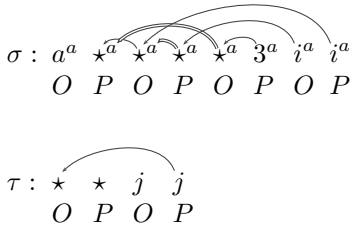
  then $\Sigma(a) = \Sigma'(a)$ and, moreover, if $a \in \mathbb{A}_\phi$ then $(u \upharpoonright X, u' \upharpoonright X, a)$ are a copycat triple, where X is the respective element of $\{AB, BC, AC\}$.

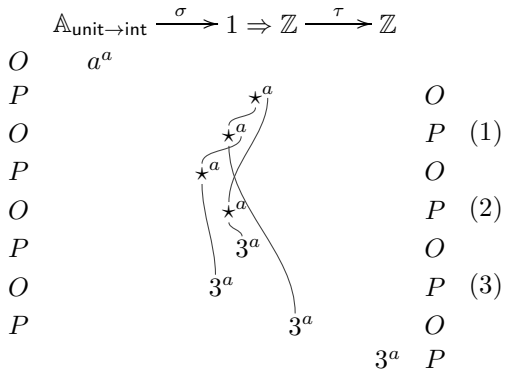
We write $Int(A, B, C)$ for the set of interaction sequences on A, B, C, and $\sigma \| \tau$ for the set of interactions between strategies $\sigma : A \to B$ and $\tau : B \to C$:

$$\sigma \| \tau = \{\, u \in Int(A, B, C) \mid (u \restriction_\gamma AB) \in \sigma \wedge (u \restriction_\gamma BC) \in \tau \,\}.$$

We shall be referring to the last condition in the definition as the *copycat condition*. According to it, during an interaction the players cannot change the parts of the store which regard names that are not available to them. Moreover, in the case that these names are of functional type, the players are obliged to copycat as far as a-descendants of these names are concerned.

Example 12. Consider strategies $\sigma : \mathbb{A}_{\mathsf{unit}\to\mathsf{int}} \to 1 \Rightarrow \mathbb{Z}$ and $\tau : 1 \Rightarrow \mathbb{Z} \to \mathbb{Z}$ given by the set of all even prefixes of plays of the form shown on the left below (for all $i \in \mathbb{Z}$).





Their interaction is on the right above. We mark polarities for σ on the left of the diagram, and for τ on the right. Consider point (1) in the interaction. In τ, P plays $\star^a$ but a is not available at that point, hence P must copycat from that point on at a-descendants of (that occurrence of) $\star^a$. This is precisely what happens in points (2) and (3).

Definition 13. *Given strategies $\sigma : A \to B$ and $\tau : B \to C$ we define the composite strategy $\sigma; \tau : A \to C$ to be $\{\, s \in P_{A\to C} \mid \exists u \in \sigma \| \tau.\ s = u \restriction_\gamma AC \,\}$.*

Strategy composition is well-defined and associative.

Definition 14. *$\mathcal{G}$ is the category of arenas and strategies, in which strategies in the prearena $A \to B$ are morphisms between A and B.*

In [MT11] we have shown how to interpret RefML in $\mathcal{G}$ so as to obtain a full abstraction result. We refer to this interpretation by writing $[\![\Gamma \vdash M]\!]$.

Example 15. Terms new_θ from Example 1 are interpreted by the strategies $\mathtt{nu}_\theta : 1 \to \mathbb{A}_\theta$, which create a fresh name of type θ and initialise it accordingly.

$$\mathtt{nu}_\theta = \{\, \star a^{(a,v)} \mid a \in \mathbb{A}_\theta \wedge v \in \{0, \star\} \,\}$$

In the remainder of the paper we identify subclasses of strategies that correspond to denotations of GrML and FunML-terms respectively.

4 Groundness

This section is devoted to finding a class of strategies that characterise ground storage, as embodied in GrML. The work predating nominal game semantics [AHM98] has established *visibility* as the condition characterising the absence of higher-order references. Visibility relies on the concept of a P-view [HO00], which can be adapted easily to our setting.

Definition 16. *The P-view $\ulcorner s \urcorner$ of a play s is inductively defined by:*

$$\ulcorner \epsilon \urcorner = \epsilon, \;\; \ulcorner m^{\Sigma} \urcorner = m^{\Sigma}, \;\; \ulcorner s\, p^{\Sigma} \urcorner = \ulcorner s \urcorner\, p^{\Sigma}, \;\; \ulcorner s\, x\, \widehat{s'\, o^{T}} \urcorner = \ulcorner s \urcorner\, x\, \widehat{o^{T}},$$

where x is some m^{Σ}, and o^{T} points either to x or to a name in its store. A play s is visible *if, for any even-length prefix $s' p^{\Sigma}$ of s, the justifier of p^{Σ} occurs in $\ulcorner s' \urcorner$. A strategy is* **visible** *if it contains only visible plays.*

Remark 17. Note that if a play satisfies visibility then its P-view is a justified sequence. Nonetheless, it may fail to be a play because of violating frugality. For example, the following odd-length play on the prearena $1 \to (\mathbb{A}_{\mathsf{unit}} \Rightarrow 1)$

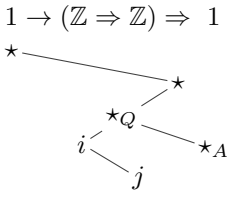
$$\begin{array}{ccccccccc} \star & & \star & & a^{a} & & \star^{a} & & b^{a,b} \\ O & & P & & O & & P & & O \end{array}$$

has P-view $\star\star\, b^{a,b}$, which breaks frugality in its last move.

In this section we first make the perhaps surprising observation that visible strategies fail to compose in the game model introduced in the previous section. We repair the failure by insisting on an additional nominal constraint.

Example 18. Consider the strategy $\sigma : 1 \to ((\mathbb{Z} \Rightarrow \mathbb{Z}) \Rightarrow 1)$ specified by the play on the left below, which breaks visibility in its last move. We label the moves of the prearena as shown on the right.

$$\begin{array}{lcccccccccccc} \sigma: & \star & & \star & & \star_Q & & \star_A & & \star_Q & & 3 & \\ & O & & P & & O & & P & & O & & P & \\ \\ \hat{\sigma}: & a^{a} & & \star^{a} & & \star^{a}_{Q} & & \star^{a}_{A} & & \star^{a}_{Q} & & \star^{a} & & \star^{a} & & 3^{a} \\ & O & & P & & O & & P & & O & & P & & O & & P \end{array}$$



We can see that $\sigma = \mathtt{nu}_{\mathsf{unit}\to\mathsf{unit}}; \hat{\sigma}$, where $\hat{\sigma} : \mathbb{A}_{\mathsf{unit}\to\mathsf{unit}} \to ((\mathbb{Z} \Rightarrow \mathbb{Z}) \Rightarrow 1)$ is specified by the play on the left above (labelling of moves follows the same pattern). Interestingly, both $\mathtt{nu}_{\mathsf{unit}\to\mathsf{unit}}$ and $\hat{\sigma}$ satisfy visibility. After composition, though, the two a-justified moves of $\hat{\sigma}$ will be deleted, since a is no longer an available name. Observe that, thanks to these two a-justified moves, the justifier of the last P-move appears in the view. With a hidden, the last move breaks visibility.

Note that the failure of compositionality stems from the special way in which P-names of higher-order reference cells are treated. This leads us to consider strategies in which such names do not occur. Below we write $\mathbb{A}_{\phi} = \biguplus_{\theta,\theta'} \mathbb{A}_{\theta\to\theta'}$.

Definition 19. *A strategy σ is* **ground** *if it is visible and $P(s) \cap \mathbb{A}_\phi = \emptyset$ for any $s \in \sigma$.*

Intuitively, the definition corresponds to removing the capability to generate higher-order reference names.

Lemma 20. *Ground strategies compose. Consequently, for any* GrML*-term $\Gamma \vdash M$, $[\![\Gamma \vdash M]\!]$ is ground.*

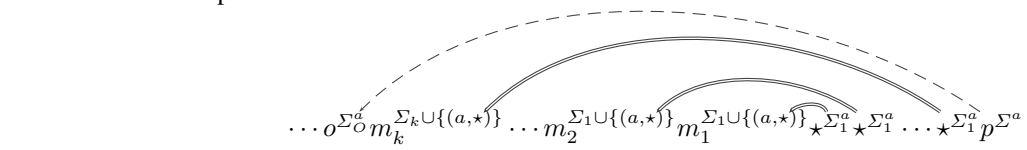
We will also have a converse of the above in the form of a definability result for finite strategies (Corollary 32). The first steps to its proof will be two factorisation arguments that remove violations of groundness.

Lemma 21. *Let $\sigma : A_1 \to A_2$. There exists a visible strategy $\overline{\sigma} : \mathbb{A}_{\mathsf{unit}\to\mathsf{unit}} \otimes A_1 \to A_2$ such that $\langle\, !_{A_1}; \mathtt{nu}_{\mathsf{unit}\to\mathsf{unit}}, \mathsf{id}_{A_1} \,\rangle; \overline{\sigma} = \sigma$. Moreover, if $P(s) \cap \mathbb{A}_\phi = \emptyset$ for any $s \in \sigma$ then $P(s) \cap \mathbb{A}_\phi = \emptyset$ for any $s \in \overline{\sigma}$.*

Proof. We are going to augment plays from σ using moves pointing to the higher-order reference represented by $\mathbb{A}_{\mathsf{unit}\to\mathsf{unit}}$ in such a way that visibility will hold. At the same time the added moves will be consistent with the copycat behaviour required during composition.

Formally, the extension $\overline{s}$ of a play $s \in \sigma$ is defined as follows. Given a store Σ, we write Σ^a for $\Sigma \cup \{(a, \star)\}$.

- $\overline{\epsilon} = \epsilon$, $\overline{so^\Sigma} = \overline{s}o^{\Sigma^a}$ (o a question), $\overline{sp^\Sigma} = \overline{s}p^{\Sigma^a}$ (p an answer).
- $\overline{sp^\Sigma} = \overline{s}\underbrace{\star^{\Sigma_1^a} \cdots \star^{\Sigma_1^a}}_{k} p^{\Sigma^a}$ (p a question), where $\overline{s} = \cdots o^{\Sigma_O^a} m_k^{\Sigma_k^a} \cdots m_2^{\Sigma_2^a} m_1^{\Sigma_1^a}$ and o^{Σ_O} (or a name in its store) justifies p^Σ in sp^Σ. Moreover, we require that the ith (counting from left to right) occurrence of $\star^{\Sigma_1^a}$ in $\star^{\Sigma_1^a} \cdots \star^{\Sigma_1^a}$ be justified by $(a, \star)$ from Σ_i^a. We depict the definition below, where the dashed line is a justification pointer to a move or to a store.



- $\overline{so^\Sigma} = \overline{s}o^{\Sigma^a}\underbrace{\star^{\Sigma^a} \cdots \star^{\Sigma^a}}_{k}$ (o an answer), where the sequence $\star^{\Sigma^a} \cdots \star^{\Sigma^a}$ answers all the a-justified questions (labelled $\star^{\Sigma_1^a}$ above) in $\overline{s}$ that occur after the justifier of o^{Σ^a}.

One can then take $\overline{\sigma}$ to be the least strategy containing all plays $\overline{s}$, where s ranges over σ. It is easy to see that no new P-names are introduced by the construction (a is an O-name). □

The second result delegates the creation of higher-order reference names and thus rids a strategy of P-names from $\mathbb{A}_\phi$. In order to overapproximate the types of names that can be used as P-names in a prearena A, we define the associated *reference set* $R(A)$. First, for each type θ, we define $R(\theta)$ by

$$R(\mathsf{unit}) = R(\mathsf{int}) = \emptyset, \quad R(\mathsf{ref}\,\theta) = \{\theta\} \cup R(\theta), \quad R(\theta_1 \to \theta_2) = R(\theta_1) \cup R(\theta_2).$$

This is extended to prearenas as follows. For each prearena A,

$$R(A) = \bigcup\nolimits_{m \in M_A} \left\{ \{\theta\} \cup R(\theta) \mid \nu(m) \cap \mathbb{A}_\theta \neq \emptyset \right\}.$$

To isolate the higher-order types in $R(A)$, we write $\mathsf{HON}(A)$ for the subset of $R(A)$ consisting of function types.

Lemma 22. *Let $\sigma : A_1 \to A_2$ and $A = \bigotimes_{\theta \in \mathsf{HON}(A_1 \to A_2)} (\llbracket \theta \rrbracket \Rightarrow \mathbb{A}_\theta)$. Let also $\mathsf{gen}_\theta = \llbracket \vdash \lambda x^\theta.\mathsf{ref}_\theta(x) \rrbracket : 1 \to (\llbracket \theta \rrbracket \Rightarrow \mathbb{A}_\theta)$. There exists a strategy $\overline{\sigma} : A \otimes A_1 \to A_2$ such that $\langle\, !_{A_1}; \langle\, \mathsf{gen}_\theta \,\rangle_{\theta \in \mathsf{HON}(A_1 \to A_2)}, \mathsf{id}_{A_1} \,\rangle; \overline{\sigma} = \sigma$. Moreover, if σ is visible, so is $\overline{\sigma}$.*

Proof. The main idea is to delegate the creation of new P-names to the additional A component by inserting extra moves from A in front of P-moves. More precisely, given $s \in \sigma$, we define an enriched play $\overline{s}$ as follows, where each v_i is a question to the appropriate $\llbracket \theta \rrbracket \Rightarrow \mathbb{A}_\theta$.

- $\overline{so^\Sigma p^T} = \overline{s}\, o^\Sigma v_1^\Sigma a_1^{\Sigma_1} \cdots v_{k-1}^{\Sigma_{k-1}} v_k p^{\Sigma_k}$, where $\mathsf{dom}(T) \setminus \mathsf{dom}(\Sigma) = \{a_1, \cdots, a_k\}$, $v_i = T(a_i)$, and $\Sigma_i = T \restriction \mathsf{dom}(\Sigma) \cup \{a_1, \cdots, a_i\}$.

In addition, $\overline{\epsilon} = \epsilon$. □

Corollary 23. *Let $\sigma : A_1 \to A_2$. There exists a ground strategy*

$$\overline{\sigma} : \mathbb{A}_{(\mathsf{unit} \to \mathsf{unit})} \otimes \Big(\bigotimes\nolimits_{\theta \in \mathsf{HON}(A_1 \to A_2)} (\llbracket \theta \rrbracket \Rightarrow \mathbb{A}_\theta) \Big) \otimes A_1 \to A_2$$

such that $\langle\, !_{A_1}; \langle\, \mathtt{nu}_{\mathsf{unit} \to \mathsf{unit}}, \langle\, \mathsf{gen}_\theta \,\rangle_{\theta \in \mathsf{HON}(A_1 \to A_2)} \,\rangle, \mathsf{id}_{A_1} \,\rangle; \overline{\sigma} = \sigma$. □

Thanks to the corollary the definability problem for finite strategies can be reduced to the same problem for ground strategies.

5 Innocence

Here we would like to find a semantic match for FunML by a notion of innocence [HO00]. FunML embodies purely functional computation in presence of reference types: although terms may receive, update and read the value of references, they cannot create new ones. The notion of innocence defined below extends the standard notion appropriately to deal with moves that carry higher-order store. Traditionally, the notion of innocence stipulates that each P-move of a strategy be determined by the P-view up to the point just before the move is played. In our case, the notion needs to be customised so as to take into account the names, and the corresponding parts of the store, which become unavailable in the P-view. The last two conditions below stipulate that P cannot play any names which do not appear in his view, and neither can he change their values.

Definition 24. *A ground strategy $\sigma : A$ is* **innocent** *if it satisfies the following conditions.*

- *If $sp^\Sigma \in \sigma$ then $\nu(p^\Sigma) \subseteq \nu(s)$* (strong determinacy).
- *If $sp^\Sigma, s' \in \sigma$ and $s'o^T \in P_A$ with $\gamma(\ulcorner s \urcorner) = \gamma(\ulcorner s' o^T \urcorner)$ then there exists $s'o^T p'^{\Sigma'} \in \sigma$ such that $\gamma(\ulcorner sp^\Sigma \urcorner) \sim \gamma(\ulcorner s' o^T p'^{\Sigma'} \urcorner)$* (innocence).

– *If* $sp^{\Sigma} \in \sigma$ *and* $\gamma(\ulcorner sp^{\Sigma} \urcorner) = s'p^{\Sigma'}$ *then* $\nu(p^{\Sigma'}) \cap \nu(s) \subseteq \nu(s')$ (innocent P-availability).
– *If* $s' \sqsubseteq s \in \sigma$ *ends in* $o^{\Sigma}p^{T}$ *and* $a \in \mathsf{dom}(T) \setminus \nu(\gamma(\ulcorner s' \urcorner))$ *then* $T(a) = \Sigma(a)$ *and, moreover, if* $a \in \mathbb{A}_{\phi}$ *then* (s, s', a) *are a copycat triple* (innocent P-storage).

Remark 25. Note that the first and third conditions above can be equivalently expressed, modulo the other conditions, as a single one:

– If $sp^{\Sigma} \in \sigma$ and $\gamma(\ulcorner sp^{\Sigma} \urcorner) = s'p^{\Sigma'}$ then $\nu(p^{\Sigma'}) \subseteq \nu(s')$.

Moreover, given the above condition, innocence can be equivalently stated as:

– If $sp^{\Sigma}, s' \in \sigma$ and $s'o^{T} \in P_A$ with $\gamma(\ulcorner s \urcorner) = \gamma(\ulcorner s'o^{T} \urcorner)$ then there exists $s'o^{T}p'^{\Sigma'} \in \sigma$ such that $\gamma(\ulcorner sp^{\Sigma} \urcorner) = \gamma(\ulcorner s'o^{T}p'^{\Sigma'} \urcorner)$.

Due to innocent P-availability and innocent P-storage, innocent strategies are uniquely determined by their behaviour on available names, that is, names appearing in the P-view after application of γ. For instance, the behaviour of an innocent strategy on an O-question asking the value of an unavailable name is a trivial copycat. Combined with the innocence condition, our observation allows us to characterise innocent strategies by their *view-functions*, defined as follows.

$$\mathsf{vf}(\sigma) = \{\gamma(\ulcorner s \urcorner) \mid s \in \sigma\}$$

Lemma 26. *Innocent strategies compose. Consequently, for any* FunML*-term* $\Gamma \vdash M$, $[\![\Gamma \vdash M]\!]$ *is an innocent strategy.*

Next we show a factorisation result for ground strategies involving innocent ones. Our first step will be to factor out violations of strong determinacy, namely, fresh-name creation, in exactly the same way as in Lemma 22. We set $\mathsf{GRN}(A) = R(A) \setminus \mathsf{HON}(A)$.

Lemma 27. *Let* $\sigma : A_1 \to A_2$ *be a ground strategy,* $A = \bigotimes_{\theta \in \mathsf{GRN}(A_1 \to A_2)} ([\![\theta]\!] \to \mathbb{A}_{\theta})$ *and* $\mathsf{gen}_{\theta} = [\![\vdash \lambda x^{\theta}.\mathsf{ref}_{\theta}(x)]\!] : 1 \to ([\![\theta]\!] \Rightarrow \mathbb{A}_{\theta})$. *There exists a strongly deterministic ground strategy* $\overline{\sigma} : A \otimes A_1 \to A_2$ *such that* $\langle !_{A_1}; \langle\, \mathsf{gen}_{\theta} \,\rangle_{\theta \in \mathsf{GRN}(A_1 \to A_2)}, \mathsf{id}_{A_1} \rangle; \overline{\sigma} = \sigma$.

In the setting without names, a factorisation to innocence would just use an integer reference, which would serve for storing the history of the play [AM97]. If one tries to apply the same rationale in the nominal setting, one soon realises that names constitute a basic obstacle because they cannot be obviously mapped into integers. In order to bridge the gap, we are going to use integers that correspond to the order in which names appear in a play, and external 'oracles' which will maintain a list of names and be able to add to it (enlisting) as well as access names at a given position (look-up).

We first fix an encoding function from plays to integers. Given a play s, the function produces a code $\#(s)$. The function first translates each name $a \in \mathbb{A}_{\theta}$ into a pair (i, θ), if a is the ith name of type θ appearing in s, and subsequently performs some standard encoding from nested strings of integers (with pointers) into integers. Therefore, the function is not injective, but orbit injective: $s \sim s' \iff \#(s) = \#(s')$. We also fix an

ordering of names appearing inside plays such that names introduced earlier (i.e. closer to the beginning of the play) appear earlier in the ordering.[3]

We next describe the family of strategies oracle_θ which we shall use. The strategy $\mathsf{oracle}_\theta : 1 \to (\mathbb{A}_\theta \Rightarrow 1) \otimes (\mathbb{Z} \Rightarrow \mathbb{A}_\theta)$ responds to the initial question with the answer $(\star_{\mathrm{e}}, \star_{\mathrm{l}})$. After that, the answer to any question a posed at $\star_{\mathrm{e}}$ will be $\star$, which should be viewed as confirmation that a has been added to the list. The strategy implements the look-up function by responding to any question i, posed at $\star_{\mathrm{l}}$ and such that the ith question posed at $\star_{\mathrm{e}}$ from the beginning of the play is a, with a.

Any strongly deterministic ground strategy $\sigma : A$ shall be converted to an innocent one which uses an external oracle for each type $\theta \in R(A)$, and a reference of type int. Inside that reference, the strategy shall keep an encoding of the whole play so far (using the function $\#(\_)$). Whenever O makes a move, say the last move in the play so^Σ, P can consult the int-reference in order to obtain a version of s where names are represented by integers. Then, P queries the external oracles (via their look-up functionality) with each integer representation in $\#(s)$, and receives the corresponding actual names as answers. At this point, P has completely reconstructed s in an innocent manner. P next updates the oracles with all the names newly introduced by o^Σ (using the enlisting functionality), and then plays his move as dictated by σ.

Lemma 28. *Let* $A = \bigotimes_{\theta \in R(A_1 \to A_2)} \big((\mathbb{A}_\theta \Rightarrow 1) \otimes (\mathbb{Z} \Rightarrow \mathbb{A}_\theta)\big)$ *and* $\sigma : A_1 \to A_2$ *be a strongly deterministic ground strategy. There exists an innocent strategy* $\overline{\sigma} : A \otimes \mathbb{A}_{\mathsf{int}} \otimes A_1 \to A_2$ *such that* $\langle !_{A_1}; \langle\langle\, \mathsf{oracle}_\theta \,\rangle_{\theta \in R(A_1 \to A_2)}, \mathsf{new}_{\mathsf{int}} \rangle, \mathsf{id}_{A_1} \rangle; \overline{\sigma} = \sigma$.

Proof. We define $\overline{\sigma}$ to be the least innocent strategy extending $\sigma' = \bigcup\{\overline{s} \mid s \in \sigma\}$, where $\overline{s}$ is a set of plays defined below by induction on the length of the play. For the base case we set $\overline{\epsilon} = \{\epsilon\}$.

Now suppose $s = \mathrm{i}^{\Sigma_{\mathrm{i}}} s' o^\Sigma p^T$. We let $\overline{s}$ contain all plays of the form

$$(\langle\langle\, (\star_{\mathrm{e}}, \star_{\mathrm{l}})_\theta \,\rangle_{\theta \in R(A_1 \to A_2)}, \hat{a}, \mathrm{i})^{\hat{\Sigma}_{\mathrm{i}}} s'' o^{\hat{\Sigma}} s_1\, s_2\, p^{\hat{T}}$$

where $\hat{a} \in \mathbb{A}_{\mathsf{int}}$ a fresh name, $\hat{\Sigma}_{\mathrm{i}} = \Sigma_{\mathrm{i}}[\hat{a} \mapsto 0]$, $\hat{\Sigma} = \Sigma[\hat{a} \mapsto \#(\mathrm{i}^{\Sigma_{\mathrm{i}}} s')]$ and $s'' \in \hat{s}'$ is given by the induction hypothesis. Let $a_1, \cdots, a_n, b_1, \cdots, b_m$ be the names in $\nu(\mathrm{i}^{\Sigma_{\mathrm{i}}} s' o^\Sigma)$ ordered according to the canonical ordering, so that $a_1, \cdots, a_n$ are the elements of $\nu(\mathrm{i}^{\Sigma_{\mathrm{i}}} s')$ and $b_1, \cdots, b_m$ those of $\nu(o^\Sigma) \setminus \nu(\mathrm{i}^{\Sigma_{\mathrm{i}}} s')$. We set s_1 to be the sequence $i_1^{\hat{\Sigma}}\, a_1^{\hat{\Sigma}}\, i_2^{\hat{\Sigma}}\, a_2^{\hat{\Sigma}} \cdots i_n^{\hat{\Sigma}}\, a_n^{\hat{\Sigma}}$, such that $a_j \in \mathbb{A}_\theta$ (some θ) is the i_jth name of type θ in $a_1, \cdots, a_n$ and $i_j^{\hat{\Sigma}}$ is justified by $\star_{\mathrm{l}\,\theta}$. Moreover, s_2 is the sequence $b_1^{\hat{\Sigma}}\, \star^{\hat{\Sigma}} \cdots b_m^{\hat{\Sigma}}\, \star^{\hat{\Sigma}}$, where each $b_j^{\hat{\Sigma}}$ is justified by the according initial $\star_{\mathrm{e}\,\theta}$. Finally, $\hat{T} = T[\hat{a} \mapsto \#(s)]$.

First, in order to show that there exists an innocent extension $\overline{\sigma}$ of σ', it suffices to show that σ' is in fact a strongly deterministic ground strategy satisfying the innocent P-availability and innocent P-storage conditions, and in addition:

– for all $s_1 p_1^{\Sigma_1}, s_2 p_2^{\Sigma_2} \in \sigma'$, if $\gamma(\ulcorner s_1 \urcorner) = \gamma(\ulcorner s_2 \urcorner)$ then $\gamma(\ulcorner s_1 p_1^{\Sigma_1} \urcorner) = \gamma(\ulcorner s_2 p_2^{\Sigma_2} \urcorner)$.

[3] Note here that names may first appear inside stores, which are not ordered. In such a case, though, and because of the availability condition, they are reachable through the store through previously introduced names, the ordering of which can be used to order the new names. E.g. in the play $a^{(a,b),(b,0)} c^{(a,b'),(c,d),(b,0),(b',0),(d,0)}$ we can order our names as a, b, c, b', d.

By construction, σ' is obviously strongly deterministic and ground, and depends solely on moves and names that are available in the P-view. We can therefore show that it satisfies the above conditions. Moreover, since $\overline{\sigma}$ contains extended versions of all plays from σ, we have that $\langle !_{A_1}; \langle \langle \, \mathsf{oracle}_\theta \, \rangle_{\theta \in R(A_1 \to A_2)}, \mathsf{new}_{\mathsf{int}} \rangle, \mathsf{id}_{A_1} \rangle; \overline{\sigma} = \sigma$. □

Corollary 29. *Let $\sigma : A_1 \to A_2$ be a ground strategy, $\Theta_1 = \mathsf{GRN}(A_1 \to A_2)$ and $\Theta_2 = R(A_1 \to A_2)$. There exists an innocent strategy*

$$\overline{\sigma} : \big(\bigotimes\nolimits_{\theta \in \Theta_1} (\llbracket \theta \rrbracket \Rightarrow \mathbb{A}_\theta) \big) \otimes \mathbb{A}_{\mathsf{int}} \otimes \big(\bigotimes\nolimits_{\theta \in \Theta_2} (\mathbb{A}_\theta \Rightarrow 1) \otimes (\mathbb{Z} \Rightarrow \mathbb{A}_\theta) \big) \otimes A_1 \to A_2$$

such that $\langle \, !_{A_1}; \langle \, \langle \, \mathsf{gen}_\theta \, \rangle_{\theta \in \mathsf{GRN}(A_1 \to A_2)}, \mathtt{nu}_{\mathsf{int}}, \langle \, \mathsf{oracle}_\theta \, \rangle_{\theta \in R(A_1 \to A_2)} \, \rangle, \mathsf{id}_{A_1} \, \rangle; \overline{\sigma} = \sigma$. □

Remark 30 (Factoring the oracle). It is interesting to note that each strategy oracle_θ can be decomposed into an innocent strategy and three reference cells, of types int, $\mathsf{ref}\ \theta$ and $\underline{\mathsf{unit} \to \mathsf{unit}}$ respectively. That is, $\mathsf{oracle}_\theta = \langle \mathtt{nu}_{\mathsf{int}}, \mathtt{nu}_{\mathsf{ref}\ \theta}, \mathtt{nu}_{\mathsf{unit} \to \mathsf{unit}} \rangle; \overline{\mathsf{oracle}_\theta}$, where $\overline{\mathsf{oracle}_\theta} : \mathbb{A}_{\mathsf{int}} \otimes \mathbb{A}_{\mathsf{ref}\ \theta} \otimes \mathbb{A}_{\mathsf{unit} \to \mathsf{unit}} \to (\mathbb{A}_\theta \Rightarrow 1) \otimes (\mathbb{Z} \Rightarrow \mathbb{A}_\theta)$ is an innocent strategy which behaves as follows:[4]

- when O provides a new name a to be enlisted, the strategy responds with $\star^{\Sigma}$, where Σ records that the ith name played by O is a (this uses the references of types int and $\mathsf{ref}\ \theta$);
- when, on the other hand, O asks what is the ith name that has been enlisted then the strategy uses the reference of type $\mathsf{unit} \to \mathsf{unit}$ in order to go back in the play (in the same fashion as in Lemma 22), until it finds a P-move $\star^{\Sigma}$ which includes a pair of values (i, a), at which point it carries back that a as an answer to i.

Although factorisation results typically deconstruct a strategy σ of a category of games $\mathcal{G}_1$ into a characteristic strategy from $\mathcal{G}_1$ and a strategy from $\mathcal{G}_2$, where $\mathcal{G}_2$ a subcategory of $\mathcal{G}_1$, the factorisation above does not follow this pattern, since the ground strategy oracle_θ is deconstructed into a strategy containing $\mathtt{nu}_{\mathsf{unit} \to \mathsf{unit}}$ (which is not ground). Note, though, that if the initial σ of Lemma 28 were finite (up to name permutations) then there would not be a need for such oracles, as σ would only contain plays with boundedly many names, which could be stored in a bounded set of external references.

Finally, we call an innocent strategy σ *compact* if $\mathsf{vf}(\sigma)$ is finite up to name-permutations, that is, if the set $\{\{\pi \cdot s \mid \pi \in \mathrm{PERM}\} \mid s \in \mathsf{vf}(\sigma)\}$ is finite.

[4] Formally, $\overline{\mathsf{oracle}_\theta}$ is the least innocent strategy which contains all plays of the form $(a_1, a_2, a_3)^{\Sigma_{\mathrm{i}}} (\star_{\mathrm{e}}, \star_{\mathrm{l}})^{\Sigma_{\mathrm{i}}[a_1 \mapsto 0]}$ and, in addition, if $(a_1, a_2, a_3)^{\Sigma_{\mathrm{i}}} (\star_{\mathrm{e}}, \star_{\mathrm{l}})^{\Sigma_{\mathrm{i}}'} s \in \overline{\mathsf{oracle}_\theta}$ then:

- $(a_1, a_2, a_3)^{\Sigma_{\mathrm{i}}} (\star_{\mathrm{e}}, \overbrace{\star_{\mathrm{l}})^{\Sigma_{\mathrm{i}}'} s \, a}^{\Sigma} \star^{\Sigma[a_1 \mapsto \Sigma(a_1)+1][a_2 \mapsto a]} \in \overline{\mathsf{oracle}_\theta}$.
- $(a_1, a_2, a_3)^{\Sigma_{\mathrm{i}}} (\star_{\mathrm{e}}, \overbrace{\star_{\mathrm{l}})^{\Sigma_{\mathrm{i}}'} s \, \overbrace{i^{\Sigma} s_1 s_2 \, a}^{\Sigma}} \in \overline{\mathsf{oracle}_\theta}$, where s_1 is a copycat sequence of questions $\star^{\Sigma[a_1 \mapsto i]}$ pointing to a_3 in preceding stores (starting from the store of i^{Σ}), and such that its last element points to a P-move $\star^{\Sigma'}$ with $\Sigma'(a_1) = i$. The sequence s_2 comprises of a series of answers $\star^{\Sigma'}$ to the questions of s_1, and $a = \Sigma'(a_2)$.

Lemma 31. *Let $\Gamma \vdash \theta$ be a typing context. For each compact innocent strategy $\sigma : [\![\Gamma \vdash \theta]\!]$ there is an* FunML*-term $\Gamma \vdash M : \theta$ such that $\sigma = [\![\Gamma \vdash M]\!]$.*

Corollary 32. *Let $\Gamma \vdash \theta$ be a typing context and $\sigma : [\![\Gamma \vdash \theta]\!]$ a strategy that is finite up to name-permutations. There is a* RefML*-term $\Gamma \vdash M : \theta$ such that $\sigma = [\![\Gamma \vdash M]\!]$. If σ is ground then there is a* GrML*-term $\Gamma \vdash M : \theta$ such that $\sigma = [\![\Gamma \vdash M]\!]$.*

6 Conclusion

We have considered three languages embodying respectively general store, ground store and pure functional computation. These have been related to three families of strategies in a nominal game model as shown below.

RefML	GrML	FunML
strategy	ground strategy	innocent strategy

In particular, our notions of groundness and innocence are nominal generalizations of the standard notions of visibility and innocence.

Another theme in research on game semantics was universality, i.e. the fact that all recursively presentable strategies were definable. We believe that, in the nominal setup, universality is bound to fail for ground and innocent strategies (wrt GrML and FunML respectively), because of the inability of these languages to store unbounded collections of names. RefML does not seem to suffer from the same limitation, as lists of names of type θ can be maintained through the type, say, $\mathsf{ref}\ (\mathsf{int} \to \mathsf{ref}\ (\theta))$.

Our results illustrate that, with some extra effort, the methodology of the semantic cube can also bear fruit in the nominal setting, though the results that are emerging are perhaps not as elegant as in the original case.

References

[AGM+04] S. Abramsky, D. R. Ghica, A. S. Murawski, C.-H. L. Ong, and I. D. B. Stark. Nominal games and full abstraction for the nu-calculus. In *Proceedings of LICS*, pages 150–159. IEEE Computer Society Press, 2004.

[AHM98] S. Abramsky, K. Honda, and G. McCusker. Fully abstract game semantics for general references. In *Proceedings of IEEE Symposium on Logic in Computer Science*, pages 334–344. Computer Society Press, 1998.

[AJM00] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 2000.

[AM97] S. Abramsky and G. McCusker. Call-by-value games. In *Proceedings of CSL*, volume 1414 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, 1997.

[GP02] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2002.

[HO00] J. M. E. Hyland and C.-H. L. Ong. On Full Abstraction for PCF: I. Models, observables and the full abstraction problem, II. Dialogue games and innocent strategies, III. A fully abstract and universal game model. *Information and Computation*, 163(2):285–408, 2000.

[HY99] K. Honda and N. Yoshida. Game-theoretic analysis of call-by-value computation. *Theoretical Computer Science*, 221(1–2):393–456, 1999.

[Lai97] J. Laird. Full abstraction for functional languages with control. In *Proceedings of 12th IEEE Symposium on Logic in Computer Science*, pages 58–67, 1997.

[Lai04] J. Laird. A game semantics of local names and good variables. In *Proceedings of FOSSACS*, volume 2987 of *Lecture Notes in Computer Science*, pages 289–303. Springer-Verlag, 2004.

[MT11] A. S. Murawski and N. Tzevelekos. Game semantics for good general references. In *Proceedings of LICS*, pages 75–84. IEEE Computer Society Press, 2011.

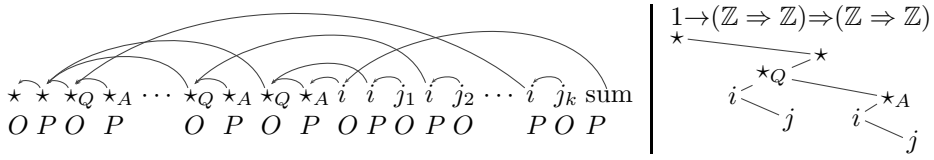
[Nic94] H. Nickau. Hereditarily sequential functionals. In *Proceedings of the Symposium of Logical Foundations of Computer Science*. Springer-Verlag, 1994. LNCS.

[Tze09] N. Tzevelekos. Full abstraction for nominal general references. *Logical Methods in Computer Science*, 5(3), 2009.

A Appendix

In this section we present some more advanced strategy examples.

Example 33. Let us consider the following strategy $\sigma : 1 \to ((\mathbb{Z} \Rightarrow \mathbb{Z}) \Rightarrow (\mathbb{Z} \Rightarrow \mathbb{Z}))$. The strategy is specified by plays of the form shown on the left below, where $\mathrm{sum} = j_1 + \cdots + j_k$, and we have labelled the moves of the prearena as on the right below.



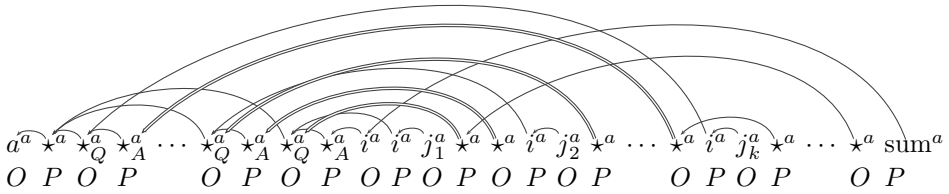
Thus, the strategy answers the initial $\star$ with the higher-order move $\star$. From that point on, at each O-question $\star_Q$ to $\star$, the strategy replies with an answer $\star_A$. When O queries the value of the returned $\star_A$, by playing some O-question i to it, the strategy propagates the question to all preceding $\star_Q$'s, and returns as an answer the sum of all the answers to those questions. This behaviour can be matched by the semantics of the following term (the last line below implicitly uses recursion).

$$\begin{aligned}&\mathsf{let}\ i = \mathsf{ref}_{\mathsf{int}}(0),\ F = \mathsf{ref}_{\mathsf{int}\to\mathsf{int}\to\mathsf{int}}(\lambda x^{\mathsf{int}}.\lambda y^{\mathsf{int}}.0)\ \ \mathsf{in}\\ &\qquad\qquad \lambda f^{\mathsf{int}\to\mathsf{int}}.\ i{+}{+};\ \mathsf{let}\ g = !F\ \mathsf{in}\ F := \lambda x^{\mathsf{int}}.\,\mathsf{if}\ (x{-}!i)\ \mathsf{then}\ f\ \mathsf{else}\ gx;\\ &\qquad\qquad\qquad \lambda y^{\mathsf{int}}.\,(!F)(!i)y + (!F)(!i{-}1)y + \cdots + (!F)1y\end{aligned}$$

The term implements the informal description of the strategy described above: it uses an internal higher-order reference F where it stores all input functions f (the ith such function is stored in $F(i)$), and returns a function which, on input y, returns the sum of applying all previous f's to y.

Example 34. Let us revisit the strategy σ from Example 33 under the light of the factorisation in Lemma 21. In particular, σ can be factorised as $\mathtt{nu}_{\mathsf{unit}\to\mathsf{unit}}; \hat{\sigma}$, where

$\hat{\sigma} : \mathbb{A}_{\mathsf{unit}\to\mathsf{unit}} \to ((\mathbb{Z} \Rightarrow \mathbb{Z}) \Rightarrow (\mathbb{Z} \Rightarrow \mathbb{Z}))$ is a ground strategy specified by plays of the form:



with $\mathrm{sum} = j_1 + \cdots + j_k$. In fact, the strategy $\hat{\sigma}$ defined above is a simplified version of the one obtained via the factorisation theorem, but it follows the same rationale of using the reference of type $\mathsf{unit} \to \mathsf{unit}$ for breaking inside the P-view in a visible way. The strategy corresponds to the term below,

$$
\begin{array}{l}
\mathsf{let}\ i = \mathsf{ref}_{\mathsf{int}}(0),\ \mathrm{inp} = \mathsf{ref}_{\mathsf{int}}(0),\ \mathrm{sum} = \mathsf{ref}_{\mathsf{int}}(0)\ \mathsf{in} \\
\qquad \lambda f^{\mathsf{int}\to\mathsf{int}}.\ \mathsf{let}\ g = (\mathsf{if}\ !i\ \mathsf{then}\ !F\ \mathsf{else}\ \lambda x^{\mathsf{unit}}.x)\ \mathsf{in} \\
\qquad\qquad i{+}{+};\ F := \lambda x^{\mathsf{unit}}.\,(\mathrm{sum} := !\mathrm{sum} + f(!\mathrm{inp});\ g());\\
\qquad\qquad \lambda y^{\mathsf{int}}.\,\mathrm{sum} := !\mathrm{sum} + fy;\ \mathrm{inp} := y;\ g();\ !\mathrm{sum}
\end{array}
$$

where F a free variable of type $\mathsf{ref}\ (\mathsf{unit} \to \mathsf{unit})$.