Oracle Game Semantics for the nu-alculus*

Pietro Di Gianantonio Marino Miculan

Dipartimento di Matematica e Informatica, Università di Udine
via delle Scienze 206, 33100 Udine, Italy
digianantonio@dimi.uniud.it miculan@dimi.uniud.it

Abstract. The long-standing problem of giving fully abstract semantics to the nu-calculus has been solved only recently with *game models* à la Hyland-Ong in the universe of Fraenkel-Mostowski sets. However, these models are still quite complicated, mainly because they aim to ensure at once both the *equivariance* of strategies, and the possibility of having strategies generating new names.
In this paper, we present an alternative, and somehow simpler, game semantics for the nu-calculus, endorsing the "object-oriented" view of references. The idea is to deal first with equivariance, and then with name generation. More precisely, we define a game model where terms are interpreted by (equivariant) strategies parametrized by some *oracle*, which can be seen as an "external supplier" of fresh names to be interrogated on the need. The oracle is provided at a second stage, when a model of complete strategies is defined by means of an extensional equivalence.

1 Introduction

Name creation and *name passing* are well-known and pervasive aspects of many computational paradigms. In languages with local variables and store, such as Idealized Algol or Standard ML, a declaration creates new *references* (storage cells) that can be passed around. Most modern process algebras, especially those deriving from the π-calculus, are built on the notion of name.

In order to study abstractly the problem of local names, Pitts and Stark introduced the *nu-calculus* [13], a simply typed call-by-value λ-calculus with a type for names. Names can be only created, compared and passed, in accordance to the intuition that they form a countable set without any internal structure.

Despite its simplicity, giving the nu-calculus a precise (i.e., fully abstract) model is very subtle. The characterization of the observational equivalence is difficult due to the interplay between higher-order functions and name creation. For this reason, standard techniques based on functor categories do not suffice, although these have been successful in modeling first-order process calculi with names, such as the π-calculus with strong or weak congruence [4,15]. Stark [14] describes models based on the functor category $\mathcal{S}et^{\mathcal{I}}$, on its subcategory known as Schanuel topos, and on categories with relations. This latter model is fully abstract for first order types, but not for higher types; in particular, the decidability of observational equivalence for higher types is still an open question.

* Work supported by EU project FP6-IST-510996 TYPES.

Recently, two different models based on game semantics have been proposed. Laird [8] presents a fully abstract model for the $\nu\sigma$-calculus, an extension of the nu-calculus with *stores*; this model is not fully abstract for the pure nu-calculus. Abramsky et al. [1] present the first fully abstract model for the original nu-calculus. A key feature of both models in [1,8] is that they use *Fraenkel-Mostowski sets*, which have been cogently advocated by Gabbay and Pitts as a simple yet expressive setting for modeling datatypes with names [5,12]. Informally, FM-sets can be seen as sets built using also a basic set of *atoms*, and where all constructions are closed under any permutation of the atoms. As a consequence we can compare two atoms, but we cannot choose a specific atom, because a choice function is not closed under permutation.

The resulting models are fully abstract but nevertheless they are quite complicated. The problem is that these models aim to capture at once both the "stability" of terms and function under permutation of names, and the possibility of generating fresh names. The tension between these antithetical requirements is the source of the complexity of these models, because specific structure is required to keep track of the already generated names.

In this work, we give a fully-abstract model for the nu-calculus following a different approach. The key observation is that the two characteristic issues of nu-calculus, namely stability under permutations and generation of fresh names, are quite orthogonal and can be dealt with separately. Thus, the interpretation of a term of the nu-calculus can be factorized into a strategy free of name generation, and a separate strategy specialized in generating names. In other words, when generation of new names is singled out and managed on its own, the rest of the calculus is interpreted quite easily in a restricted class of "good" strategies.

- For "good" strategies, all names must appear as equivalent: the set of names does not have any internal structure; we can only check whether two names are the same. Thus good strategies must be invariant under name permutations.
- Good strategies cannot create new names directly; the creation of new names is delegated to the environment (the Opponent). A fresh name is created by calling an "external function" that at each call returns a different value.

A clear advantage of this method is that it is very close to "real world" computing situations; e.g. Lisp and Unix provide the function `gensym` to generate names, which other programs can only compare and move around.

Let us describe these steps in more detail. The first step is to rule out strategies that discriminate between names. We define a category of games and *equivariant* strategies, that is strategies "stable" under permutation of names. An elegant way to enforce this condition is to build games in FM-sets, as in [1,8], or more simply in G-set theory where G is the group of permutations of names.

The second step is to impose that names are generated by the environment, or by the Opponent. Informally, we assume that there exists an *oracle* that can be interrogated to supply names. Formally, we interpret the nu-calculus assuming that in any context there exists an *oracle variable* with type[1] $\mathsf{1} \to \nu$.

[1] We use the game $\mathsf{1} \to \nu$ instead of ν for technical reasons, due to the call by value evaluation strategy of the nu-calculus.

The interpretation of new constructor of the nu-calculus is then straightforwardly defined as the strategy that interrogates the oracle variable for a (new) name, receives a name and returns it. In this approach all the different instances of new in a term M refer to the same oracle variable; thus, it is easy to keep track of the different calls of new and to assure that each one generates a new name. Thus, the (equivariant) strategy σ interpreting a term M will contain explicitly the interrogations to the oracle variable. As usual for strategies, σ must consider all possible behaviour for the oracle, also "incorrect" ones which generate a name more than once. The "effective" strategy describing M is then obtained by composing σ with any "good" *oracle strategy* δ in game $\mathsf{1} \to \nu$ that replies to each question with a different name. The strategy δ does not satisfy the equivariance restriction that other strategies must satisfy.

In order to compare our model with the previous game models for the nu-calculus [1,8], it is useful to recall two different approaches to game semantics for languages with references, such as Idealized Algol.

The model presented in this paper is close in spirit to the game model for Idealized Algol presented by Abramsky and McCusker [2], where variables are treated in a parametric way using the so-called *object oriented interpretation:* for each variable there is an associated strategy ("cell") giving the interpretation to methods of accessing the variable. This is analogous to our treatment of new with its associated oracle strategy. This approach to Idealized Algol is sometimes called the *bad variable approach*, because for definability (and hence full abstraction) to work, the language must be expressive enough so that "bad variables" are definable. The standard Idealized Algol does not have such expressiveness, hence a bad variable constructor (called `mkvar`) must be explicitly added to the language. To be precise, the "bad variable" model is equationally, but not inequationally, fully asbtract for Idealized-Algol-without-`mkvar`.

In contrast Ong [11] presents a *good variable* game semantics for Idealized Algol where the `mkvar` constructor is not necessary. That model is similar, in the kind of the definition used, with the game models for names presented in [1].

Comparing these two approaches, we find the "bad variable" approach conceptually more attractive and mathematically cleaner: it does so largely by hiding states in a remarkably effective way. The disadvantage is the need of the extra constant `mkvar`. In contrast, the "good variable" approach makes state explicit in the model construction; hence it is more complicated, and so, less attractive from a practical point of view. The advantage is that it does lead to a fully abstract model for languages without the need for `mkvar`. However, it is worth pointing out that in the case of the nu-calculus, as we present in this paper, we can use a parametric (object oriented) interpretation for names without the need of extending the language with any spurious constant.

Synopsis. In Section 2 we recall the nu-calculus. In Section 3 we introduce the category of G-games and equivariant strategies, which will be used in Section 4 for giving a first model of the nu-calculus; although incorrect, this model is useful for proving a general definability result which will be inherited by a second, correct, model. The suitable category of "games with an oracle" is introduced in Section 5; in Section 6 we will define a decidable congruence relations on

$$\frac{(x:A)\in\Gamma}{s,\Gamma\vdash x:A}(var)\quad \frac{s,\Gamma,x:A\vdash M:B}{s,\Gamma\vdash\lambda x:A.M:A\to B}(abs)\quad \frac{s,\Gamma\vdash M:A\to B\quad s,\Gamma\vdash N:A}{s,\Gamma\vdash MN:B}(app)$$

$$\frac{}{s,\Gamma\vdash \mathsf{true}:o}(tt),\ \frac{}{s,\Gamma\vdash \mathsf{false}:o}(ff)\quad \frac{s,\Gamma\vdash M:o\quad s,\Gamma\vdash N_1:A\quad s,\Gamma\vdash N_2:A}{s,\Gamma\vdash \mathsf{if}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_2:A}(ite)$$

$$\frac{n\in s}{s,\Gamma\vdash n:\nu}(n)\quad \frac{}{s,\Gamma\vdash \mathsf{new}:\nu}(new)\quad \frac{s,\Gamma\vdash M:\nu\quad s,\Gamma\vdash N:\nu}{s,\Gamma\vdash M=N:o}(Eq)$$

Fig. 1. Typing system of nu-calculus.

strategies of this category, sufficient to obtain a correct categorical model of nu-calculus, and a coarser one which generates the fully-abstract model. Conclusions are in Section 7. Appendix A recalls briefly call-by-value game semantics.

2 The nu-calculus

The nu-calculus is a call-by-value simply typed λ-calculus, extended with dynamic creation of references [13]. The types are defined as follows:

$$A,B ::= \nu \mid o \mid A\to B$$

The only ground types are that of names ν, and that of Booleans o. Terms are defined as follows, where n ranges over a set of abstract names $\mathcal{N}=\{n_0,n_1,n_2,\dots\}$:

$$M ::= x \mid \lambda x{:}A.M \mid MM \mid \mathsf{true} \mid \mathsf{false} \mid \mathsf{if}\ M\ \mathsf{then}\ M\ \mathsf{else}\ M \mid n \mid \mathsf{new} \mid M=N$$

Names form a infinite, enumerable set with no internal structure; one may think of names as natural numbers. Thus names can be only created (by new, which generates a different name each time is evaluated), compared (by =) and passed (used as arguments of a function). (We adopt the presentation with the new constructor, because it turns out to be simpler to model in our approach.)

The typing system for nu-calculus is a simple extension of the usual one for λ-calculus. The typing judgment has the form

$$s,\Gamma\vdash M:A$$

where s is a list of all distinct names. The typing rules are in Figure 1; notice that comparison is restricted to terms of type ν.

Expressions are closed terms (but possibly with free names). An expression is in *canonical form*, or a *value*, if it is either a boolean constant, an abstraction, a variable, or a name. For any type A and list of names s, we define the sets

$$\mathrm{Exp}_A(s)\triangleq\{M\mid s,\emptyset\vdash M:A\}\qquad \mathrm{Can}_A(s)\triangleq\{C\mid C\in\mathrm{Exp}_A(s), C\text{ canonical}\}$$

Operational Semantics. Similarly to Standard ML, the nu-calculus has a call-by-value reduction strategy. The operational semantics needs to keep track of the already generated names; therefore, evaluation judgments have the form

$$s\vdash M\Downarrow(s_1)C$$

$$\frac{C \in \mathrm{Can}_A(s)}{s \vdash C \Downarrow ()C}(Can) \qquad \frac{n \notin s}{s \vdash \mathsf{new} \Downarrow (n)n}(New)$$

$$\frac{s \vdash M \Downarrow (s_1)\lambda x{:}A.M' \quad s, s_1 \vdash N \Downarrow (s_2)V \quad s, s_1, s_2 \vdash M'\{V/x\} \Downarrow (s_3)R}{s \vdash MN \Downarrow (s_1, s_2, s_3)R}(App)$$

$$\frac{s \vdash M_1 \Downarrow (s_1)n \quad s, s_1 \vdash M_2 \Downarrow (s_2)n}{s \vdash M_1 = M_2 \Downarrow (s_1, s_2)\mathsf{true}}(EqTrue)$$

$$\frac{s \vdash M_1 \Downarrow (s_1)n \quad s, s_1 \vdash M_2 \Downarrow (s_2)m \quad (n \neq m)}{s \vdash M_1 = M_2 \Downarrow (s_1, s_2)\mathsf{false}}(EqFalse)$$

$$\frac{s \vdash M \Downarrow (s_1)\mathsf{true} \quad s, s_1 \vdash M_1 \Downarrow (s_2)V}{s \vdash \mathsf{if}\ M\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 \Downarrow (s_1, s_2)V}(IfTrue)$$

$$\frac{s \vdash M \Downarrow (s_1)\mathsf{false} \quad s, s_1 \vdash M_2 \Downarrow (s_2)V}{s \vdash \mathsf{if}\ M\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 \Downarrow (s_1, s_2)V}(IfFalse)$$

Fig. 2. Operational semantics of the nu-calculus.

where s, s_1 are list of all different names, and $M \in \mathrm{Exp}_A(s)$ and $C \in \mathrm{Can}_A(s, s_1)$. The intuitive interpretation is: if the names on the list s are already present (i.e., allocated), then the term M reduces to value C, generating the names in the list s_1. The rules for the operational semantics are in Figure 2.

Observational equivalence. As usual the reductions strategy induces an observational equivalence on (possibly open) terms. Informally, two terms are observational equivalent if they are not distinguishable by observing their behaviour; so, two observationally equivalent terms can be freely interchanged.

In the nu-calculus, we can restrict to observers (i.e., contexts) of boolean type. A *program* P is a closed term of boolean type (i.e., $s, \emptyset \vdash P : o$, for some s); a *program context* $P[-]$ is a program with zero or more occurrences of an hole $[-]$; if M is some closed term, we denote by $P[M]$ the program obtained by filling the hole with M.

Definition 1 (Observational Equivalence). *For $M_1, M_2 \in \mathrm{Exp}_A(s)$, we say that M_1 and M_2 are* observationally equivalent *(written $s \vdash M_1 \approx_A M_2$) if for all program contexts $P[-]$ and $b \in \{\mathsf{true}, \mathsf{false}\}$:*

$$(\exists s_1.s \vdash P[M_1] \Downarrow (s_1)b) \iff (\exists s_2.s \vdash P[M_2] \Downarrow (s_2)b)$$

Remark 1. Often the nu-calculus, and similar nominal calculi, have a ν binder instead of the constant new. The informal meaning of a term $\nu x.M$ is "generate a new, fresh name and assign it to the variable x in the scope M". The operational semantics of this binder is given by the following rule:

$$\frac{s, n \vdash M \Downarrow (s_1)C}{s \vdash \nu n.M \Downarrow (s_1, n)C}(Local)$$

In other words ν creates and binds a name before evaluating the expression; new just evaluates to a fresh name. The two formulations are equivalent; in fact, the

following operational equivalences hold:

$$s \vdash \mathsf{new} \approx_\nu \nu n.n \qquad s \vdash \nu n.M \approx_A (\lambda x{:}\nu.M)\mathsf{new}$$

In our approach we find simpler to model the constructor new, hence we chose this presentation. Still, we will use "$\nu x.M$" as a shorthand for $(\lambda x{:}\nu.M)\mathsf{new}$. Also, for $s = n_1 \dots n_k$, we define $\nu s.M = \nu n_1. \dots \nu n_k.M$.

Due to its finitary nature, one may expect that the operational equivalence is simple; however, there are some subtleties, such as the following equivalences:

- $\nu n.\nu n'.(n = n') \approx_o \mathsf{false}$
- $\nu n.(\lambda x : \nu.(x = n)) \approx_{\nu\to o} \lambda x : \nu.\ \mathsf{false}$
- $\nu n.\nu n'.(\lambda f : \nu \to o.(fn = fn')) \approx_{(\nu\to o)\to o} \lambda f.\ \mathsf{true}$
  where the equality "$M = N$" between booleans is a syntactic shorthand for the term $\mathsf{if}\ M\ \mathsf{then}\ N\ \mathsf{else\ if}\ N\ \mathsf{then\ false\ else\ true}$.

3 *G*-games

In this section we present the first step of our approach, that is a theory of games with *names*. Since this theory is a simple derivation of standard games for call-by-value λ-calculus, we start recalling the basic definition in the spirit of Honda and Yoshida [6] in a formulation closer to Laurent [9]. See Appendix A for a more detailed presentation of game semantics.

Definition 2 (Arenas). *An* arena *is a triple $A = \langle M_A, \lambda_A, \vdash_A \rangle$ where M_A is a set of moves; $\lambda_A : M_A \to \{\mathrm{PQ}, \mathrm{PA}, \mathrm{OQ}, \mathrm{OA}\}$ is a labelling function that indicates whether a given move is a P-move or an O-move, and whether it is a question (Q) or an answer (A); and $\vdash_A$ is the* justification relation*, that is a relation on $M_A \times M_A$, denoted by $m_1 \vdash_A m_2$, and a subset of M_A, of initial moves, denoted by $\vdash_A m$, satisfying the following conditions:*

(i) Every initial move is an P-answer.
(ii) If $m \vdash_A m'$ then m and m' are moves by different players.
(iii) If $m \vdash_A m'$ and m is an answer then m' is a question ("Answers may only justify questions.").

Example 1. The simplest arena is the empty arena $\mathsf{0} = \langle \emptyset, \emptyset, \emptyset \rangle$, the singleton arena is defined by $\mathsf{1} = \langle \{*\}, \{(*, \mathrm{PA})\}, \vdash_{\mathsf{1}} \rangle$ where $\vdash_{\mathsf{1}} *$.

In the following we will use also the Boolean arena o having two moves $\{\mathbf{t}, \mathbf{f}\}$, both initial, and the arena of names $\nu = (M_\nu, \lambda_\nu, \vdash_\nu)$, where M_ν is the set of names $\mathcal{N} \triangleq \{n_0, n_1, \dots\}$; for all $x \in M_\nu$: $\lambda_\nu(x) = \mathrm{PA}$ and $\vdash_\nu x$.

Set-like constructions (product, coproduct, function space...) can be defined on arenas (or better, on prearenas). Arenas, and higher-order prearenas, are inhabited by *strategies*, which can be informally described as sets of plays (i.e., lists of alternating player/opponent moves); see Appendix A for details.

An important feature of strategies (of the correct type) is that they can be composed like functions; this justifies the following definition:

Definition 3. *The category $\mathcal{G}$ of games and strategies has arenas as objects, and innocent strategies in the prearena $A \rightharpoonup B$ as morphisms from A to B. The category $\mathcal{G}_{\mathbf{t}}$ is the subcategory of $\mathcal{G}$ of* total *strategies only.*

It is a standard result of game semantics [6,9] that the category $\mathcal{G}_{\mathbf{t}}$ has $\mathtt{0}$ as initial object, $\mathtt{1}$ as terminal object, $\times$ as binary product. One can observe instead that $\mathcal{G}$ does not have binary product.

In giving the semantics of the nu-calculus, we use Moggi's approach with computational monads [10]. Therefore, we have now to define a strong monad $T : \mathcal{G}_{\mathbf{t}} \longrightarrow \mathcal{G}_{\mathbf{t}}$ with T-exponentials. The construction is the following. The inclusion functor $I : \mathcal{G}_{\mathbf{t}} \hookrightarrow \mathcal{G}$ has as a right adjoint the lift-functor $L(A) = A_\perp$ with unit $\mathtt{up}_A : A \to A_\perp$ and counit $\mathtt{dn}_A$. The associated monad $T = \langle LI, L\mathtt{dn}I : (LI)^2 \to LI, \mathtt{up} : Id \to LI\rangle$ is a strong monad with a tensorial strengh $t_{A,B} : A \times TB \to T(A \times B)$. For every pair of arenas A, B the T-exponential $(TB)^A$ is a arena such that for every arena C there is a natural bijection between $\mathcal{G}_{\mathbf{t}}(C \times A, TB)$ and $\mathcal{G}_{\mathbf{t}}(C, (TB)^A)$; in our case $(TB)^A$ is simply the arena $A \to B$.

Proposition 1. *The category $\mathcal{G}_{\mathbf{t}}$, with the strong monad T, is a λ_c-model.*

The Kleisli category $(\mathcal{G}_{\mathbf{t}})_T$ coincides with the category $\mathcal{G}$. We will interpret terms as morphisms in the category $\mathcal{G}$, mainly because strategies in $\mathcal{G}(A, B)$ can be described more simply than the corresponding strategies in $\mathcal{G}_{\mathbf{t}}(A, TB)$.

***G*-games** We present now a theory of games "up-to permutations" of names.

Arenas can be defined by set-like constructions, possibly involving the arena of names ν. Thus moves in these arenas are nested pairs whose final elements can be names. Intuitively, a name is equivalent to any other, the only relevant feature being they are all distinguished. It follows that plays or strategies differing just for a permutation of names involved should be considered isomorphic, and hence indistinguishable by other strategies.

In order to enforce the condition that functions (i.e., strategies) are stable under permutations of names, we need to define how a permutation on names induces a corresponding action on moves and plays. The simplest way to formalize this idea is via the notion of G-sets.

Definition 4 (G-sets). *Let G be a group. A* (left) G-set *is a pair $(X, \cdot_X)$ where X is a set and $\cdot_X : G \times X \to X$ is a* (left) G-action*, that is*

$$id \cdot_X x = x \qquad \pi_1 \cdot_X (\pi_2 \cdot_X x) = (\pi_1 \pi_2) \cdot_X x$$

An equivariant *function $f : (X, \cdot_X) \to (Y, \cdot_Y)$ between G-sets is a function $f : X \to Y$ such that $f(\pi \cdot_X x) = \pi \cdot_Y f(x)$ for all $x \in X$ and $\pi \in G$.*

The G-sets and equivariant functions form a category denoted by $\mathbf{B}G$.

In this paper we are interested in G-sets where the group of action G is the group, G_ϵ of permutations on the set of names $\mathcal{N}$, or one of the subgroups of G_ϵ, denoted by G_s, formed by the permutations for which the names in a list s are fixed-points.

Intuitively, for a permutation $\pi \in G_s$, the action $\pi \cdot_X -$ is the operation renaming the names appearing in the elements of X, according to π.

It is easy to see that the basic constructions in $\mathcal{S}et$ can be readily lifted to $\mathbf{B}G$ (In fact, $\mathbf{B}G$ is a topos). Let $X = (X, \cdot_X)$ and $Y = (Y, \cdot_Y)$ be two G-sets:

(i) Cartesian product: $X \times Y \triangleq (X \times Y, \cdot)$ where $\pi \cdot (x, y) \triangleq (\pi \cdot_X x, \pi \cdot_Y y)$. The G-set $1 = (\{*\}, \{(\pi, *) \mapsto *\})$ is the terminal object in $\mathbf{B}G$.
(ii) Coproduct: $X + Y \triangleq (X + Y, \cdot)$ where $\pi \cdot in_X(x) \triangleq in_X(\pi \cdot_X x)$, $\pi \cdot in_Y(y) \triangleq in_Y(\pi \cdot_Y y)$. The G-set $0 = (\emptyset, \emptyset)$ is the initial object in $\mathbf{B}G$.
(iii) Finite lists: $X^* = (X^*, \cdot)$, where for any sequence $(x_1, \ldots, x_n)$, we define $\pi \cdot (x_1, \ldots, x_n) = (\pi \cdot_X x_1, \ldots, \pi \cdot_X x_n)$.
(iv) Function space: $Y^X \triangleq (Y^X, \cdot)$, where for $f : X \to Y$ we define $\pi \cdot f : X \to Y$ as the function $(\pi \cdot f)(x) = \pi \cdot_Y f(\pi^{-1} \cdot_X x)$.
(v) We say that $(X, \cdot_X)$ is a *G-subset* of $(Y, \cdot_Y)$ if $X \subseteq Y$ and for all $\pi \in G, x \in X : \pi \cdot_X x = \pi \cdot_Y x$. Notice that if an element $y \in Y$ belongs to X, then the whole *orbit* of y is in X: for all $\pi \in G : \pi \cdot_Y y \in X$.

Remark 2. The category $\mathbf{B}G$ of G-sets is similar to that of *Fraenkel-Mostowski sets with finite support*, or *nominal sets*, used in [5,12,1]; the main difference is that only elements with *finite support* are considered there. We can enforce the "finite support" condition by restricting to the category of *continuous* G-sets, i.e. G-sets $(X, \cdot)$ whose action $\cdot : G \times X \to X$ is required to be continuous when X and G are endowed with the discrete and Baire topology respectively.

The category of continuous G-sets is a *full* subcategory of G-sets, and closed under usual object constructions (product, coproduct, ...). In fact, all types of nu-calculus are interpreted as continuous G-sets. Therefore, we can use both categories for the constructions presented in this paper. However, in order to keep the theory as simple as possible we prefer to avoid the conditions about finite support, and hence we work in the category of generic G-sets.

Now we apply G-sets to the definition of (pre)arenas and strategies.

Definition 5 (G-arena and G-strategies). *Given a group G, a G-*(pre)arena *is a (pre)arena $A = \langle M_A, \lambda_A, \vdash_A \rangle$ with the additional condition that M_A is a G-set of moves, and $\lambda_A : M_A \to \{PQ, PA, OQ, OA\}$ is an equivariant function (where the action of the G-set $\{PQ, PA, OQ, OA\}$ is the identity).*

*A G-*strategy *σ on the G-prearena A is a strategy σ with the additional condition that σ is a G-subset of positions.*

It is almost immediate to see that the composition of G-strategies is a G-strategy. Thus we can lift the definition of categories of games to this setting:

Definition 6. *Given a group G, we denote by $G\mathcal{G}$ the category of G-arenas and innocent G-strategies, and by $G\mathcal{G}_{\mathbf{t}}$ the category of G-arenas and total, innocent G-strategies.*

Also the construction of products, and of the strong monad T as above, can be lifted to these categories of G-games.

The definition of G-arena is almost the same of FM-arena in [1], and of ν-arena in [8]. On the other hand the definition of G-strategy is quite different from the corresponding definitions in [1,8]. Our definition of G-strategy is simply the translation, inside G-set theory, of the standard definition of strategy. As a consequence, a G-strategy cannot make any choice among the names not belonging to the list s, and so cannot generate any fresh name, as shown in the next example.

Example 2. The arena of names ν of Example 1 can be obviously endowed with a G_s-action $\pi \cdot_\nu x \triangleq \pi(x)$, becoming thus a G_s-arena. Note that there is only one (never answering) G_ϵ-strategy in the prearena $1 \rightharpoonup \nu$. In fact if $*n \in \sigma : 1 \rightharpoonup \nu$ then also $\pi \cdot (*n) = *(\pi \cdot_\nu n) \in \sigma$ since σ is a G_ϵ-subset of moves, and thus for all $n' \in \mathcal{N} : *n' \in \sigma$, which contradicts the determinacy condition. By the same arguments, the G_s-strategies on the prearena $1 \rightharpoonup \nu$ can only answer with a name in s to the initial move of the opponents .

In [1,8] strategies are allowed to choose a fresh name. This is obtained by relaxing the determinacy condition and allowing strategies with multiple choice: in order to choose a fresh name, a strategy chooses all available fresh names.

4 A game semantics for the nu-calculus by translation

In this section we use the categories introduced in Section 3 for defining a first, albeit incorrect, game semantics for the nu-calculus. This semantics is given by translating the nu-calculus into a traditional CBV λ-calculus. By a standard construction of the monadic approach to denotational semantics [10,14], it is possible to define a model of this CBV λ-calculus (and hence, to interpret the nu-calculus) in the categories $\mathcal{G}$ and $G_s\mathcal{G}$. However, as we will see in Section 6, only the latter will give rise to a fully-abstract model, via an extensional collapse.

As target language, let us consider a CBV λ-calculus $\mathcal{L}$, built from the ground types $o, \nu, 1$, where 1 is the singleton type inhabited only by the constant $()$. The expressions of $\mathcal{L}$ coincide with those of the nu-calculus, with the exception of the constructor new, which is omitted, and the constant $()$, which is introduced.

We define the translation $(\!|-|\!)$ from terms in context of the nu-calculus to terms in contexts of $\mathcal{L}$, as follows:

$$(\!|s, \Gamma \vdash M : A|\!) \triangleq \mathsf{g} : 1 \to \nu, \Gamma \vdash M[(\mathsf{g}())/\mathsf{new}] : A$$

where g is a variable, reminiscent of the `gensym()` function in Unix or Lisp.

Then, the monadic interpretations $\mathcal{L}$ in the categories of games $\mathcal{G}$ and $G_s\mathcal{G}$ is readily defined. The interpretation of ground types $o, \nu, 1$ of $\mathcal{L}$ is simply the G-arenas $o, \nu, 1$; functional types are interpreted as usual.

The interpretation of an $\mathcal{L}$-term M, with names contained in a list s, in a context $[\![x_1 : A_1, \ldots x_n : A_n \vdash M : B]\!]_{\mathcal{L}}$ is a G_s-strategy from the arena $A_1 \times \ldots \times A_n$ to the arena B. According to the monadic approach we need to define a morphism for each basic constant; in particular a names n, contained in a list s, is interpreted by the strategies answering with n to any intial move of the Opponent. The constructor (constant) "=" is interpreted by a strategy $\mathsf{eq} : \nu \times \nu \rightharpoonup o$ checking for the equality between two names. It is immediate to check that eq is equivariant and so it is a G_s-strategy. Moreover, in order to interpret the test functions it is necessary to define a family of total strategies $\mathsf{cond}_A : (o \times A \times A) \rightharpoonup A$. Exploiting the fact that $G_s\mathcal{G}_{\mathbf{t}}$ has coproducts and that $o = 1 + 1$, cond_A can be defined as $i_A; [\pi_1, \pi_2]$ where i_A is the natural isomorphism between $(A \times A) + (A \times A)$ and $(1 + 1) \times A \times A = o \times A \times A$.

The semantics of the nu-calculus judgement $s, \Gamma \vdash M : A$ in the category $\mathcal{G}$ (or $G_s\mathcal{G}$) is then defined by:

$$[\![s, \Gamma \vdash M : A]\!]_t \triangleq [\![(\!|s, \Gamma \vdash M : A|\!)]\!]_{\mathcal{L}}$$

It is important to notice that this "semantics-by-translation" approach exploits the peculiarity of game semantics of being an intensional model; it will not work so easily in extensional semantic models, such as domain theory. Without introducing explicit additional machinery for bookkeeping names (like, e.g., global stores), semantics-by-translations in extensional models will interpret all occurrences of new with the same value, and hence terms with different operational behaviours may be identified in the model. This does not happen in intensional models based on games: there are no unwanted identifications of terms caused by substitution of each occurrence of new with a call to g.

Example 3. The two terms $(\nu x_1 \nu x_2 . x_1 = x_2)$ and $(\nu x_1 \nu x_2 . \mathsf{true})$ are behaviourally different, however they will be identified by the semantic-by-translation in any extensional model of CBV λ-calculus. Instead, the strategy

$$\begin{array}{rcl} [\![\vdash (\nu x_1 \nu x_2 . x_1 = x_2) : o]\!]_t & = & [\![\vdash (\lambda x_1 \lambda x_2 . x_1 = x_2)\ \mathsf{new}\ \mathsf{new} : o]\!]_t \\ & = & [\![\mathsf{g} : 1 \to \nu \vdash (\lambda x_1 \lambda x_2 . x_1 = x_2)\ \mathsf{g}()\ \mathsf{g}() : o]\!]_{\mathcal{L}} \end{array}$$

is different from the strategy

$$\begin{array}{rcl} [\![\vdash (\nu x_1 \nu x_2 . \mathsf{true}) : o]\!]_t & = & [\![\vdash (\lambda x_1 \lambda x_2 . \mathsf{true})\ \mathsf{new}\ \mathsf{new} : o]\!]_t \\ & = & [\![\mathsf{g} : 1 \to \nu \vdash (\lambda x_1 \lambda x_2 . \mathsf{true})\ \mathsf{g}()\ \mathsf{g}() : o]\!]_{\mathcal{L}} \end{array}$$

Both strategies interrogate the environment twice asking for a name ($\mathsf{g}()$), and while the second strategy always returns the answer true the first strategy checks whether the Opponent has been "coherent" in his behaviour, returning false if the Opponent returns two different values. So we exploit the fact that strategies consider any possible behaviour the Opponent, also behaviours that are inconsistent.

The strategy

$$[\![\vdash (\lambda x . x = x)\ \mathsf{new}]\!]_t \ = \ [\![\mathsf{g} : 1 \to \nu \vdash (\lambda x . x = x)\ \mathsf{g}()]\!]_{\mathcal{L}}$$

is equal to the strategy

$$[\![\vdash (\lambda x . \mathsf{true})\ \mathsf{new}]\!]_t \ = \ [\![\mathsf{g} : 1 \to \nu \vdash (\lambda x . \mathsf{true})\ \mathsf{g}()]\!]_{\mathcal{L}}$$

In fact the first strategy interrogates just once the environment asking for the name $\mathsf{g}()$ and returns the value true. □

However, the model makes too many distinction between terms and fails to be correct. For example, $[\![\vdash \mathsf{true}]\!]_t \neq [\![\vdash (\lambda x . \mathsf{true})\mathsf{new}]\!]_t$, because the model counts how many times the term new has been called. Another wrong inequality is $[\![\vdash \mathsf{new} = \mathsf{new}]\!]_t \neq [\![\vdash (\lambda x_1 \lambda x_2 . \mathsf{false})\ \mathsf{new}\ \mathsf{new}]\!]_t$, because the model does not force the different evaluations of new to have distinct behaviour.

As we will see in Section 6, a correct model of the nu-calculus can be obtained by supplying a correct interpretation for the parameter g; this amounts to provide an oracle strategy generating a new name each time it is interrogated, or from another point of view, this ammount to take a restricted (and decidable) form of extensional collapse between strategies.

A motivation for introducing this first, incorrect, model is that the definability theorem for this semantics is an immediate result: it suffices to rephrase some already known results. Moreover it will be simple to transport the definability result from this semantics to the correct one, later on.

Definition 7. *A strategy σ is* totally defined *if for every even length position $s \in \sigma$ there exists a move m such that $sm \in \sigma$.*

A strategy σ in $G_s\mathcal{G}$ is compact *if the set of its P-views is generated by transposing a finite set of sequences (i.e. the set of the orbits of P-views is finite).*

Proposition 2 (Definability).

(i) For any list of names s, for all arenas $A_1, \ldots, A_n, B$ definable in $\mathcal{L}$ and for every compact and totally defined G_s-strategy $\sigma : (A_1 \times \ldots \times A_n) \rightharpoonup B$, there exists a $\mathcal{L}$-term M such that $[\![x_1 : A_1, \ldots, x_n : A_n \vdash M : B]\!]_{\mathcal{L}} = \sigma$.

(ii) For any list of names s, for all arenas $A_1, \ldots, A_n, B$ definable in the nu-calculus and for every compact and totally defined G_s-strategy $\sigma : ((\mathsf{1} \to \nu) \times A_1 \times \ldots \times A_n) \rightharpoonup B$, there exists a term M such that $[\![s, x_1{:}A_1, \ldots, x_n{:}A_n \vdash M{:}B]\!]_t = \sigma$.

Proof. (i) This point can be proved by standard techniques, in particular rephrasing the corresponding proof for CBV λ-calculus in [6]. Briefly we associate definable strategies to terms in a *finite canonical form*. There main difference being that [6] uses case statements having a natural number n as first argument, thus making a finite choice depending on the value of n; in our case we have to consider a case statement that takes, as first argument, a list s of names and makes a finite choice depending on the repetions of names in the list s.
(ii) Follows directly from (i). □

5 Games with Oracle

In the previous section we have introduced the main idea of our approach, namely, the creation of new names is delegated to an "oracle" outside the game. Every time a new name is needed (i.e., new is evaluated), this oracle is asked to generate the name. Thus an oracle can be seen as a parameter g of type $\mathsf{1} \to \nu$. However in order to have a correct model we need to explicitly supply the oracle in such a way that the required identities hold.

The idea of working with parametrized morphisms can be formalized using the *simple slice* construction. We recall here the basic definition from [7].

Definition 8. *Given a category $\mathcal{C}$ with binary products and $E \in \mathcal{C}$, the* simple slice category *$\mathcal{C}/\!\!/E$ is defined as follows:*

(i) $Obj(\mathcal{C}/\!\!/E) \triangleq Obj(\mathcal{C})$, *and* $\mathcal{C}/\!\!/E(A,B) \triangleq \mathcal{C}(E \times A, B)$;
(ii) *the identity map on* A *in* $\mathcal{C}/\!\!/E$ *is the projection* $\pi' : E \times A \to A$ *in* $\mathcal{C}$;
(iii) *given* $f : A \to B$ *and* $g : B \to C$ *in* $\mathcal{C}/\!\!/E$, *their composition is defined as* $g \bullet f \triangleq g \circ \langle \pi, f \rangle$ *(where* $\circ$ *is the composition in* $\mathcal{C}$*):*

For any $E \in \mathcal{C}$, there is an immersion functor $E^* : \mathcal{C} \longrightarrow \mathcal{C}/\!\!/E$ defined as follows:

(i) $E^*(A) \triangleq A$ for every $A \in \mathcal{C}$,
(ii) $E^*(f) \triangleq f \circ \pi'$ for every $f \in \mathcal{C}(A,B)$.

It is easy to check that if $\mathcal{C}$ has finite products, then $\mathcal{C}/\!\!/E$ has finite products, which are also preserved by E^*.

Using this construction we define family of categories of G_s-games parametric on the object $\nu_\perp = 1 \to \nu$, which will be used for the semantics of nu-calculus.

Definition 9 (O_s-games). *Given a list of names* s, *the category of total* oracle games *is* $O_s\mathcal{G}_{\mathbf{t}} \triangleq G_s\mathcal{G}_{\mathbf{t}}/\!\!/\nu_\perp$; *thus, the objects are* G_s*-arenas while the morphism from* A *to* B *are total, innocent* G_s*-strategies in the* G_s*-prearena* $((1 \to \nu) \times A) \rightharpoonup B$.

A similar construction of parametric games can be applied also to the category $G_s\mathcal{G}$ of general G_s-strategies; however, since $G_s\mathcal{G}$ does not have products, the construction is a little more involved.

Let us consider the strong monad $T : G_s\mathcal{G}_{\mathbf{t}} \longrightarrow G_s\mathcal{G}_{\mathbf{t}}$ as defined in Section 3, with strength t. We define a functor $T' \triangleq T/\!\!/\nu_\perp : O_s\mathcal{G}_{\mathbf{t}} \longrightarrow O_s\mathcal{G}_{\mathbf{t}}$ as follows:

– $T'(A) \triangleq TA$,
– $T'(f) \triangleq Tf \circ t_{\nu_\perp, A}$, for every $f \in O_s\mathcal{G}_{\mathbf{t}}(A,B)$.

By general results about simple slice categories and strong functors [7], it turns out that also this functor T' is strong. Thus, we simply define the category $O_s\mathcal{G}$ as the Kleisli category $(O_s\mathcal{G}_{\mathbf{t}})_{T'}$; the construction is necessary for our aims, but we do not present its explicit definition. As one can expect, morphisms from A to B in $O_s\mathcal{G}$ are (isomorphic to) G_s-strategies $(1 \to \nu) \times A \rightharpoonup B$.

Notice that finally, in the category $O_s\mathcal{G}$, there are strategies from 1 to ν whose answer are not limited to the names in the list s. Indeed, an O_s-strategy in the O_s-prearena $1 \rightharpoonup \nu$ is a G_s-strategy in the G_s-prearena $(1 \to \nu) \times 1 \rightharpoonup \nu$, i.e. $(1 \to \nu) \rightharpoonup \nu$. A play in a strategy of this prearena can interrogate the oracle several times, receiving several names $n_1, \ldots, n_k$, and finally can return any n_i picked among them, as in the figure aside. If we apply a permutation to such a play, we still obtain a legal play; hence, an O_s-strategy $1 \to \nu$ is a set of plays of this kind, closed under any permutation of names.

	$(1 \to \nu) \rightharpoonup \nu$			
O		$*$		
P	$*$			
O			n_1	
$\vdots$		$\vdots$		
P	$*$			
O			n_k	
P				n_i

Using the monadic approach we can directly define the interpretation $[\![\ ]\!]_O$ of the nu-calculus in the Kleisli categories $O_s\mathcal{G} = (O_s\mathcal{G}_{\mathbf{t}})_{T'}$. We define $[\![s, \Gamma \vdash M : A]\!]_O$ as the O_s-strategies is given by a standard monadic approach: the interpretation of types is that given in Section 4, the morphisms eq and cond_A are given by

the immersion through the functor $(1 \to \nu)^*$ of the corresponding morphisms in $G_s\mathcal{G}$. Finally the interpretation of new is given by an $O_s\mathcal{G}$-morphism from 1 to ν, i.e. an $O_s\mathcal{G}_{\mathbf{t}}$-morphism $1 \to \nu_\perp$, i.e. a total G-strategy in the prearena $(1 \to \nu) \times 1 \rightharpoonup \nu_\perp$. Therefore, we interpret new simply as the first projection.

It is straighforward to prove that for every term in a context $s, \Gamma \vdash M : A$ we have that the two strategies $[\![s, \Gamma \vdash M : A]\!]_t$ and $[\![s, \Gamma \vdash M : A]\!]_O$ are the same. It follows that also this second semantics is not correct; we need to provide an oracle to obtain a correct semantics, as we are going to see next.

6 Categorical quotienting

We consider two different quotients on the set of morphisms of $O_s\mathcal{G}$, that is, two different congruence relations on strategies: a finer and decidable one, sufficient to obtain a correct categorical model of nu-calculus, and a coarser one generating the fully abstract model. Some definitions are here required.

Definition 10. *Given a list of name s and an injective function $l_s : \mathbb{N} \to \mathcal{N}\backslash s$, the* s-oracle strategy $\delta_{l_s} : 1 \rightharpoonup \nu_\perp = 1 \rightharpoonup (1 \to \nu)$ *is composed of the subsequences of the infinite sequence $* * * \; l_s(0) \; * \; l_s(1) \; * \; l_s(2) \; * \ldots$.*

The table aside shows in which component of the arena each of the moves in δ_{l_s} lies. Every time the strategy δ_{l_s} is interrogated, it generates a new name, not contained in the list s. Obviously the strategy δ_{l_s} is not innocent (and not even a G_s-strategy). Formally, we could define a suitable category of games and (history dependent) strategies, with δ_{l_s} a morphism in it. For sake of simplicity, we can avoid to introduce this further category, because the only place we need to consider δ_{l_s} is the next definition: the constructions used there (i.e., product, composition and equality) are the usual ones from the standard set-theoretic treatment [6], where strategies are seen just as sets of plays. Strategies are seen as sets of play also in the following definition.

	$1 \rightharpoonup (1 \to \nu)$		
O	$*$		
P		$*$	
O		$*$	
P			$l_s(0)$
O		$*$	
P			$l_s(1)$
O		$*$	
P			$l_s(2)$
		$\vdots$	$\vdots$

Definition 11. *Given a list of name s and an injective function $l_s : \mathbb{N} \to \mathcal{N}\backslash s$, two $O_s\mathcal{G}$ ($O_s\mathcal{G}_{\mathbf{t}}$) morphisms σ, τ from A to B, i.e. two (total) G_s-strategies in $((1 \to \nu) \times A) \rightharpoonup B$, are* equivalent*, written $\sigma \cong_s \tau$, if the G_s subset of plays generated by $\sigma \circ (\delta_{l_s} \times \mathsf{id}_A)$ and $\tau \circ (\delta_{l_s} \times \mathsf{id}_A)$ are equal, in other words, if the set of G_S-orbits of elements in $\sigma \circ (\delta_{l_s} \times \mathsf{id}_A)$ is equal to the set of G_S-orbits of elements in $\tau \circ (\delta_{l_s} \times \mathsf{id}_A)$.*

It is immediate to see that the relation $\cong_s$ is independent from the chosen injective function l_s. An alternative characterization of the equivalence $\cong_s$ is the following $\sigma \cong_s \tau$ iff for any s-oracle strategy δ_{l_s} and play p in $\sigma \circ (\delta_{l_s} \times \mathsf{id}_A)$ there exits an s-oracle strategy $\delta_{l'_s}$ such that the play p is contained in $\tau \circ (\delta_{l'_s} \times \mathsf{id}_A)$, and vice versa.

The relation $\cong_s$ is decidable on finite strategies, in fact, the supports of these strategies are finite and hence there is only a finite number of permutations to check. Hence, models in $O_s\mathcal{G}$ still can be effective.

The proof that $\cong_s$ respects the composition $\bullet$ can be derived using the alternative characterization of $\cong_s$ and following property. For any pair of $O_s\mathcal{G}$ morphism, $\sigma : A \to B,\ \tau : B \to C$, for any s-oracle strategy δ_{l_s}, and play p in $(\tau \bullet \sigma) \circ (\delta_{l_s} \times \mathsf{id}_A)$ there exist two injective functions $l'_s, l''_s : \mathbb{N} \to \mathcal{N} \setminus s$ ranging over disjoint sets of names such that p is a play in $(\tau \circ (\delta_{l'_s} \times \mathsf{id}_B)) \circ (\sigma \circ (\delta_{l_s} \times \mathsf{id}_A))$ and vice versa, for any pair of injective functions $l'_s, l''_s : \mathbb{N} \to \mathcal{N} \setminus s$ ranging over disjoint set of names, and for any play p in $(\tau \circ (\delta_{l'_s} \times \mathsf{id}_B)) \circ (\sigma \circ (\delta_{l_s} \times \mathsf{id}_A))$ there exists s-oracle strategy δ_{l_s} such that p is contained in $(\tau \bullet \sigma) \circ (\delta_{l_s} \times \mathsf{id}_A)$.

It is not difficult to show that $\cong_s$ respects also the cartesian product $\times$ in $O_s\mathcal{G}_{\mathbf{t}}$, and the strong monad T'. We define the quotient category $\widetilde{O_s\mathcal{G}}$, whose objects are G-arenas and whose arrows are equivalence classes of strategies. It follows that $\widetilde{O_s\mathcal{G}}$ is a monadic model of the CBV λ-calculus, inducing the interpretation of the nu-calculus as previously given on $O_s\mathcal{G}$. We denote with $[\![\ ]\!]_{\widetilde{O_s}}$ the semantic interpretation of the nu calculus in $\widetilde{O_s\mathcal{G}}$.

In several aspects the model $\widetilde{O_s\mathcal{G}}$ is similar to the categories of ν-strategies and S-strategies presented in [1,8]; in all cases, strategies are equivalence classes closed by G_s-actions.

The category $\widetilde{O_s\mathcal{G}}$ yields a correct and adequate model of the nu-calculus (as proved by the next proposition) and it is a categorical model in the sense of [14].

Proposition 3. *1. (Correctness) For every closed term M, if $s \vdash M \Downarrow (s_1)V$ then $[\![s \vdash M : A]\!]_{\widetilde{O_s}} = [\![s \vdash \nu s_1.V : A]\!]_{\widetilde{O_s}}$.*

2. (Adequacy) For every closed term M of type o, $\exists s_1\,.\, s \vdash M \Downarrow (s_1)\mathsf{true}$ if and only if $[\![s \vdash M : o]\!]_{\widetilde{O_s}} = [\![\vdash \mathsf{true} : o]\!]_{\widetilde{O_s}}$.

3. (Definability) For every arenas $A_1, \ldots, A_n, B$ definable in the nu-calculus and for every compact and totally defined $O_s\mathcal{G}$ morphism $\sigma : A_1 \times \cdots \times A_n \to B$, there exists a nu-calculus term M such that $[\![x_1{:}A_1, \ldots, x_n{:}A_n \vdash M{:}B]\!]_{\widetilde{O}} = \sigma$.

Proof. 1. Correctness can be proved by structural inductions on the derivation of $s \vdash M \Downarrow (s_1)V$. Here we consider only the case of the rule (EqFalse) which is the only rule that is not valid in the model on $O_s\mathcal{G}$. By inductive hypothesis we have that $[\![\vdash \nu s.M_1 : \nu]\!]_{\widetilde{O_\epsilon}} = [\![\vdash \nu s s_1.n : \nu]\!]_{\widetilde{O_\epsilon}}$ and $[\![\vdash \nu s s_1.M_2 : \nu]\!]_{\widetilde{O_\epsilon}} = [\![\vdash \nu s s_1 s_2.m : \nu]\!]_{\widetilde{O_\epsilon}}$, it follows that $[\![\vdash \nu s.M_1 = M_2 : o]\!]_{\widetilde{O_\epsilon}} = [\![\vdash \nu s s_1 s_2.n = m : o]\!]_{\widetilde{O_\epsilon}}$, moreover one readily sees that $[\![\vdash \nu s s_1 s_2.n = m : o]\!]_{\widetilde{O_\epsilon}} \approx [\![\vdash \nu s s_1 s_2.\mathsf{false} : o]\!]_{\widetilde{O_\epsilon}}$ from which the thesis.

2. Since in the nu-calculus every boolean term converges, from correctness it follows that $s \vdash M \Downarrow (s_1)\mathsf{true}$ if and only if $[\![s \vdash M : o]\!]_{\widetilde{O_s}} = [\![s, s_1 \vdash \mathsf{true} : o]\!]_{\widetilde{O_s}}$, moreover we have $[\![s, s_1 \vdash \mathsf{true}]\!]_{\widetilde{O}} \approx [\![\vdash \mathsf{true}]\!]_{\widetilde{O}}$, from which the thesis.

3. From Proposition 2. □

As usual in game semantics, to obtain a fully abstract model we need to perform the extensional collapse of the model $O_s\mathcal{G}$.

Definition 12. *(i) We say that an $O_s\mathcal{G}$ morphism $\sigma : 1 \to o$, i.e. a G_s-strategy in the arena $(1 \to \nu) \times 1 \rightharpoonup o$, is* truthful *if $\langle \delta_{l_s}, \mathsf{id}_1 \rangle ; \sigma$ for some (or, equivalently any) injective function $l_s : \mathbb{N} \to \mathcal{N} \setminus s$) contains the play $*\boldsymbol{t}$.*

(ii) Two $O_s\mathcal{G}$ morphisms σ, τ from $A \to B$ are extensionally equal*, written $\sigma \approx_s \tau$, if for every pair of $O_s\mathcal{G}$ morphism $\rho : 1 \to A$, $\xi : B \to o$ we have that $\xi \circ \sigma \circ \rho$ is truthful iff $\xi \circ \tau \circ \rho$ is truthful.*

It is not difficult to prove that $\cong_s \subseteq \approx_s$, that $\approx_s$ respects the composition of morphisms, the cartesian product $\times$ in $O_s\mathcal{G}_{\mathbf{t}}$, and the strong monad T'. We define the quotient category $\widehat{O_s\mathcal{G}}$ whose morphisms are equivalence classes of strategies with respect to $\approx_s$. It follows that $\widehat{O_s\mathcal{G}}$ is a quotient category of $\widehat{O_s\mathcal{G}}$ and thus inheriting from it the semantic interpretation of the nu-calculus. We denote with $[\![\ ]\!]_{\widehat{O_s}}$ this interpretation.

Finally we can state our main result of full abstraction:

Theorem 1. *For every pair of typing judgments $s, \emptyset \vdash M_1 : A$, $s, \emptyset \vdash M_2 : A$,*

$$s \vdash M_1 \approx_{s_A} M_2 \quad \Longleftrightarrow \quad [\![s, \emptyset \vdash M_1 : A]\!]_{\widehat{O_s}} = [\![s, \emptyset \vdash M_2 : A]\!]_{\widehat{O_s}}$$

Proof. It follows from the previous Proposition 3 by standard proof techniques. In particular the implication $\Rightarrow$ follows from the definability result; the implication $\Leftarrow$ follows from the adequacy result. □

7 Conclusions

In this paper we have presented, considering the concrete case of the nu-calculus as example, what we think to be a general recipe to deal with names and name creation in a game semantics. The basic step of this approach are the following:

1. take your favorite category of games $\mathcal{G}$;
2. repeat the construction of $\mathcal{G}$ (definition of games and strategies) inside the theory of G_s-sets instead of the ordinary set theory, thus obtaining a category $G_s\mathcal{G}$; define the arena of "names" by taking as moves the set of atoms.
3. delegate the creation of names to the environment, through the interrogation of an external parameter; this can be formalized using a slice category of $G_s\mathcal{G}$.
4. perform the extensional collapse in the slice category, by suppling a strategy creating a sequence of unique names.

Of course there is no guarantee that this recipe will work also in other contexts: it is always necessary to check that the required categorical properties enjoyed by the category $\mathcal{G}$ are preserved in this tour. However we think that this method has the advantage of using just simple notions of game semantics, and it is general enough to be applied also to other calculi featuring generation of fresh names.

References

1. S. Abramsky, D. Ghica, A. Murawski, C.-H. Ong, and I. Stark. Nominal games and full abstraction for the nu-calculus. In *Proc. LICS*, pages 150–159. IEEE, 2004.
2. S. Abramsky, G. McCusker. Lineary, sharing and state: a fully abstract game semantics for Idealized Algol. In *Algol-like Languages.* (Birkhauser), 317–348, 1997.
3. S. Abramsky and G. McCusker. Full abstraction for idealized algol with passive expressions. *Theor. Comput. Sci.*, 227(1-2):3–42, 1999.

4. M. Fiore, E. Moggi, and D. Sangiorgi. A fully-abstract model for the π-calculus. In *Proc. 11th LICS*. IEEE, 1996.
5. M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2002.
6. K. Honda and N. Yoshida. Game-theoretic analysis of call-by-value computation. *Theor. Comput. Sci.*, 221(1-2):393–456, 1999.
7. B. Jacobs. Parameters and parametrization in specification using distributive categories. *Fund. Informaticae*, 24(3), 1995.
8. J. Laird. A game semantics of local names and good variables. In I. Walukiewicz, editor, *Proc. FoSSaCS*, volume 2987 of *LNCS*, pages 289–303. Springer, 2004.
9. O. Laurent. Polarized proof-nets and $\lambda\mu$-calculus. *Theor. Comput. Sci.*, 290(1):161–188, 2003.
10. E. Moggi. Notions of computation and monads. *Informa. Compu.*, 1, 1993.
11. C.-H. L. Ong. Observational equivalence of 3rd-order idealized algol is decidable. In *Proc. LICS*, pages 245–256. IEEE Computer Society, 2002.
12. A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186:165–193, 2003.
13. A. M. Pitts and I. D. B. Stark. Observable properties of higher order functions that dynamically create local names, or "what's new?" In *Proc. MFCS*, volume 711 of *Lecture Notes in Computer Science*, pages 122–141. Springer, 1993.
14. I. Stark. Categorical models for local names. *Lisp and Symbolic Computation*, 9(1):77–107, 1996.
15. I. Stark. A fully abstract domain model for the π-calculus. In *Proc. LICS'96*, pages 36–42. IEEE, 1996.

A Call by value game semantics

In this section we recall the main definition of call by value game semantics, using ideas by Honda and Yoshida [6] in a formulation closer to Laurent [9].

An ***arena*** is a triple $A = \langle M_A, \lambda_A, \vdash_A \rangle$ where M_A is a set of moves; $\lambda_A : M_A \to \{\mathrm{PQ}, \mathrm{PA}, \mathrm{OQ}, \mathrm{OA}\}$ is a labelling function that indicates whether a given move is a P-move or an O-move, and whether it is a question (Q) or an answer (A); and $\vdash_A$ is the *justification relation*, that is a relation on $M_A \times M_A$, denoted by $m_1 \vdash_A m_2$, and a subset $I_A \subseteq M_A$ of *initial* moves, denoted by $\vdash_A m$, satisfying the following conditions:

(i) Every initial move is an P-answer.
(ii) If $m \vdash_A m'$ then m and m' are moves by different players.
(iii) If $m \vdash_A m'$ and m is an answer then m' is a question ("Answers may only justify questions.").

We write $\overline{I_A}$ for the set $M_A \setminus I_A$ of non initial moves of A, and $\overline{(-)}$ for the function that inverts the P/O-designation of a move, so that e.g. $\overline{\mathrm{PQ}} = \mathrm{OQ}$ and $\overline{\mathrm{OA}} = \mathrm{PA}$ etc.

Let A and B be arenas. The ***product arena*** $A \times B$ is a sort of smash union of the arena graphs of A and B. Formally:

$$M_{A\times B} = (I_A \times I_B) + \overline{I_B} + \overline{I_A} \qquad \lambda_{A\times B}(m) = \begin{cases} \text{PA} & \text{if } m \in I_A \times I_B \\ \lambda_A(m) & \text{if } m \in \overline{I_A} \\ \lambda_B(m) & \text{if } m \in \overline{I_B} \end{cases}$$

$$\vdash_{A\times B} m \iff m = (a, b)$$
$$m \vdash_{A\times B} n \iff m \vdash_A n \ \vee\ m \vdash_B n \ \vee\ (m = (a, b) \ \wedge\ (\, a \vdash_A n \ \vee\ b \vdash_B n\,)).$$

Given an arena A, we write $\overline{A}$ for the arena obtained by inverting the P/O-label of each move in A. Then, roughly speaking, the ***function space prearena*** $A \rightharpoonup B$ is obtained by grafting B just under the initial moves of $\overline{A}$. The precise definition is the following:

$$M_{A\rightharpoonup B} = M_A + M_B \qquad \lambda_{A\rightharpoonup B}(m) = \begin{cases} \text{OQ} & \text{if } m \in I_A \\ \overline{\lambda_A(m)} & \text{if } m \in \overline{I_A} \\ \lambda_B(m) & \text{if } m \in M_B \end{cases}$$

$$\vdash_{A\rightharpoonup B} m \iff \vdash_A m$$
$$m \vdash_{A\rightharpoonup B} n \iff (\vdash_A m \wedge\ \vdash_B n) \ \vee\ m \vdash_A n \ \vee\ m \vdash_B n.$$

Notice that the functions space prearena is not an arena since initial moves are O-question. The ***function space arena*** $A \to B$ is obtained by adding an initial P-answer to the functions space prearena.

$$M_{A\to B} = \{*\} + M_{A\rightharpoonup B} \qquad \lambda_{A\to B} = \{(*, \text{PA})\} \cup \lambda_{A\rightharpoonup B}$$

$$\vdash_{A\to B} m \iff m = *$$
$$m \vdash_{A\to B} n \iff (\, m = * \wedge\ \vdash_{A\rightharpoonup B} n\,) \ \vee\ m \vdash_{A\to B} n$$

The notion of prearena has been introduced because strategies and composition of strategies can be defined more naturally using prearenas, than arenas.

The ***lifted arena*** $A_\perp$ is obtained from A by adding two moves, namely, $\perp^{\text{PA}}$, which is the new initial move, and $\perp^{\text{OQ}}$, which is a O-question, such that $\perp^{\text{PA}}$ justifies $\perp^{\text{OQ}}$ which justifies each initial move of A, and moves from A inherit the relation $\vdash_A$. Observe that $A_\perp$ is isomorphic to $1 \to A$.

For example, the arena $1 \to \nu = \nu_\perp$ is $(M_\nu \cup \{*_1, *_2\}, \lambda, \vdash)$ where for all $x \in M_\nu, \lambda(x) = \text{PA}$, $\lambda(*_1) = \text{PA}$, $\lambda(*_2) = \text{OQ}$, and $\vdash *_1 \vdash *_2 \vdash x$.

Justified sequences and strategies. A ***justified sequence*** over a prearena A is a finite sequence of alternating moves such that, except the first move which is initial, every move m has a *justification pointer* (or simply *pointer*) to some earlier move m^- satisfying $m^- \vdash_A m$; we say that m is *explicitly justified* by m^-. A question (respectively answer) in a justified sequence s is said to be ***pending*** just in case no answer (respectively question) in s is explicitly justified by it. We define the ***P-view*** $\ulcorner s \urcorner$ of a justified sequence s as :

$$\ulcorner s\, m \urcorner = \ulcorner s \urcorner m \qquad \text{if } m \text{ is a P-move}$$
$$\ulcorner s\, m_0\, u\, m \urcorner = \ulcorner s \urcorner m_0\, m \qquad \text{if the O-move } m \text{ is explicitly justified by } m_0$$

In $\ulcorner s\, m_0\, u\, m \urcorner$ the pointer from m to m_0 is retained, similarly for the pointer from m in $\ulcorner s\, m \urcorner$ in case m is a P-move.

Definition 13. *A justified sequence s over A is said to be a **legal position** (or a **play**) just in case it satisfies:*

Visibility: *Every P-move (respectively non-initial O-move) is explicitly justified by some move that appears in the P-view (respectively O-view) at that point.*
Well-Bracketing: *Every answer is explicitly justified by the last pending question at that point.*

Finally, a ***strategy*** σ for a (pre)arena A is defined to be a non-empty, prefix-closed set of legal positions of A satisfying:

1. For any odd-length $s \in \sigma$, if sm is a legal position then $sm \in \sigma$.
2. *(Determinacy)* For any even-length s, if sm and sm' are in σ then $m = m'$.

A strategy is said to be ***innocent*** [6] if for any odd-length $sm \in \sigma$ and for any even-length $s' \in \sigma$ such that $\ulcorner s \urcorner = \ulcorner s' \urcorner$, we have $s'm \in \sigma$. That is to say, σ is completely determined by a partial function f (say), which maps P-views p to *justified P-moves* (i.e. $f(p)$ is a P-move together with a pointer to some move in p). We write f_σ for the minimal such function that defines σ. We say that an innocent strategy σ is *compact* just in case f_σ is a finite function (or equivalently σ contains only finitely many odd-length P-views).

A strategy $\sigma : A \rightharpoonup B$ is said to be ***total*** if it replies to every initial move in A with an initial move in B.

Composition of strategies. For arenas A_1, A_2 and A_3, a ***local sequence*** over (A_1, A_2, A_3) is a sequence u of elements from the set $M_{A_1} + M_{A_2} + M_{A_3}$ such that every element m in u other than the first (which must be initial in A_1) has a pointer to some earlier element m^- satisfying:

(i) for $i = 2, 3$, if m is initial in A_i then m^- is initial in A_{i-1}
(ii) if m is non-initial in A_i, then m^- is in A_i and $m^- \vdash_{A_i} m$

further u satisfies *locality*: If m' and m'' occur consecutively in s such that $m' \in M_{A_i}$ and $m'' \in M_{A_j}$ then $|i - j| \leq 1$. We write $\mathcal{L}(A_1, A_2, A_3)$ for the set of *local sequences* over (A_1, A_2, A_3).

Now suppose σ and τ are strategies over prearenas $A \rightharpoonup B$ and $B \rightharpoonup C$ respectively. The set of ***interaction sequences*** arising from σ and τ, written $\mathbf{ISeq}(\sigma, \tau)$, consists of local sequences $u \in \mathcal{L}(A, B, C)$ such that

(i) $u \upharpoonright (A, B) \in \sigma$,
(ii) $u \upharpoonright (B, C) \in \tau$,

where $u \upharpoonright (A', A'')$, called the (A', A'')-*component* of u, is the subsequence of u consisting of moves from the prearena $A' \rightharpoonup A''$. We can now define the composite strategy $\sigma \,;\, \tau$ over $A \rightharpoonup C$:

$$\sigma \,;\, \tau \;=\; \{u \upharpoonright (A, C) \mid u \in \mathbf{ISeq}(\sigma, \tau)\}.$$

In $u \upharpoonright (A, C)$ the pointer of every initial A-move is to the unique initial C-move. By standard arguments it is possible to show that composition preserves innocence and totality [6].