Abstract

Game semantics has been used with considerable success in formulating fully abstract semantics for languages with higher-order procedures and a wide range of computational effects. Recently, nominal games have been proposed for modeling functional languages with names. These are ordinary games cast in the theory of nominal sets developed by Pitts and Gabbay. Here we take nominal games one step further, by developing a fully abstract semantics for a language with nominal general references.

1 Introduction

One of the most challenging problems in denotational semantics of programming languages is that of modeling languages with general references. General references are references which can store not only values of ground types (integers, booleans, etc.) but also of higher types (procedures, higher-order functions, or references themselves). The general reference is a very useful and powerful programming construct, and it can be used to encode a wide range of computational effects and programming paradigms (e.g. object-oriented programming). The added expressiveness of general references makes their denotational models complicated, mainly because of the phenomena of dynamic update and interference present in the language.

Fully abstract models for general references have been achieved via game semantics in [3], and via abstract categorical semantics in [9]. The presentation in [9] does not distinguish between λ-abstraction and local fresh-reference creation (ν-abstraction), and hence is distance from the common use of references in programming languages. On the other hand, the calculus examined in [3] distinguishes between λ- and ν-abstractions, yet encodes references as variables of a read/write product type. This leads to the presence of bad variables, as read/write-product semantical objects may not necessarily denote references. Bad variables lead to unwanted behaviors and prohibit the use of equality tests for references.

In this paper we obtain the first full-abstraction result for a statically-scoped language with general references, good variables and reference-equality tests, which faithfully reflects the practice of real programming languages such as ML. We follow the alternative (nominal) approach of treating references separately from variables, as names, extending the ν-calculus of Pitts and Stark [14]. The ν-calculus is a paradigmatic λ-calculus with names, in which names are constant terms of ground type that "...are created with local scope, can be tested for equality and can be passed around via function application, but that is all”. Here we use names for references, so names are of reference types and may also be dereferenced and updated, introducing thus a λ-calculus with nominal general references, the νρ-calculus.

Nominal games were introduced in [2] as the basis for the first fully abstract model of the ν-calculus. They constitute a version of Honda-Yoshida CBV-games [6] built in the universe of nominal sets of Pitts and Gabbay [5, 13]. Nominal sets are sets whose elements entail a finite number of names, and which are acted upon by finite name-permutations. Thus, the nominal games of [2] are CBV-games played using moves-with-names, that is moves attached with a finite set of names representing the names introduced so far. Our intention was to build a model for the νρ-calculus using nominal games, yet we discovered discrepancies arising from the use of name-sets in moves: the unordered nature of name-creation is incompatible with the deterministic behavior of strategies and, in fact, nominal games do not form a category.

Hence, we recast nominal games using moves attached with name-lists instead of name-sets, and rectifying other discrepancies. Moreover, since names model references of several types, our construction is based on nominal sets over countably infinitely many sets of names—one for each type. From the basic category of nominal games we obtain an adequate model for νρ by using a store arena, which is obtained as the canonical solution to the domain equation (SE)

---

1. By “bad variables” we mean read/write constructs of reference type which do not yield references, like mkvα of [3].

---

2. A different version of nominal games was introduced in [8], yet it did not yield a fully abstract model for the ν-calculus.

3. Also, the use of name-lists allows us to construct nominal games in nominal sets with strongly supported elements (ν. definition 1).
of page 5. For full abstraction we need to apply some further constraints on the way the store is accessed in nominal games, obtaining thus tidy strategies.

Summarising, the contributions of this paper are: a) the introduction of a lambda-calculus with nominal general references, name-equality test and good variables; b) the rectification of nominal games; c) the construction of a fully abstract model using nominal games with tidy strategies. An appealing further direction is that of abstracting the basic nominal games model to a categorical level, in the spirit of [1, 9]; a first step in this direction has already been taken in the abstract description of a lambda-model (section 3.1).

2 Theory of nominal sets

We give a short overview of nominal sets, which will be used as the basis for all constructions presented in this paper. Intuitively, nominal sets are sets whose elements entail a finite number of names, and which are acted upon by finite name-permutations. We present these following [13].

Assume a countably infinite set of names \( N \). The elements of \( N \) are names to type \( A \) and are denoted by \( a^A, b^A, \ldots \). We write \( \text{PERM}(N_A) \) for the group of finite permutations of \( N_A \). We let \( N \triangleq \bigcup_{A \in TY} N_A \) be the set of (general) names and \( \text{PERM}(N) \triangleq \bigoplus_{A \in TY} \text{PERM}(N_A) \) be the group of (finite) permutations. Names are denoted by \( \alpha, \beta, \gamma, \ldots \), and permutations by \( \pi, \pi', \ldots \); in particular, \( (\alpha \beta) \) denotes the permutation that only swaps names \( \alpha \) and \( \beta \) (of same type) and \( \text{id} \) denotes the identity permutation.

A nominal set \( X \) is a set equipped with an action from \( \text{PERM}(N) \), that is a function \( \_ \circ \_ : \text{PERM}(N) \times X \to X \) such that, for any \( \pi, \pi' \in \text{PERM}(N) \) and \( x \in X \),

\[
\pi \circ (\pi' \circ x) = (\pi \circ \pi') \circ x \quad \text{id} \circ x = x \quad (P)
\]

Moreover, all \( x \in X \) have finite support \( S(x) \), where

\[
S(x) \triangleq \{ \alpha \in N \mid \text{for infinitely many } \beta, (\alpha \beta) \circ x \neq x \} \quad (S)
\]

We can see that \( N \) in particular is a nominal set. For \( x \in X \) and \( \alpha \in N \), \( \alpha \) is fresh for \( x \), written \( \alpha \notin x \), if \( \alpha \notin S(x) \). \( x \) is equivariant iff it has empty support. \( N^\# \) stands for the nominal set of finite lists of distinct (i.e. pairwise fresh) names.

If \( Y \) is a nominal set and \( X \subseteq Y \) then \( X \) is a nominal subset of \( Y \) iff \( X \) is closed under permutations, these acting on \( Y \). If \( X, Y \) are nominal sets then their product \( X \times Y \) is also a nominal set, with permutations defined componentwise. Moreover, a relation \( R \subseteq X \times Y \) is a nominal subset of \( X \times Y \) iff, for any permutation \( \pi \) and \( (x, y) \in X \times Y \),

\[
xRy \iff (\pi(x), \pi(y)) \in R \quad (\pi, \pi') \text{ are \( R \)-equivariant.}
\]

We call such an \( R \) a nominal relation. Accordingly, \( f : X \to Y \) is a nominal function iff \( f(\pi \circ x) = \pi \circ f(x) \), for any \( x \in X \) and \( \pi \). For example, \( S(\_): X \to \mathcal{P}_{\text{fin}}(N) \) is a nominal function.

We let \( \text{Nom}_{TY} \) be the category of nominal sets (on \( N \)) and nominal functions. In nominal sets we can succinctly define name-abstraction: for each \( \alpha \in N \) and \( x \in X \) let

\[
\langle \alpha \rangle x \triangleq \{(\beta, y) \in N \times X \mid (\beta = \alpha \lor \beta \neq x) \land y = (\alpha \circ \beta) \circ x \}
\]

We can show \( S((\alpha) x) = S(x) \setminus \{ \alpha \} \). Another form of abstraction involves restricting the support of an element to that of a given name-list: for any \( x \in X \) and \( \alpha \in N^\# \) let

\[
[x]_{\alpha} \triangleq \{ y \in X \mid \exists \pi, \pi = \alpha \land y = \pi \circ x \}
\]

If \( S(x) \supseteq S(\alpha) \) then \( S([x]_{\alpha}) = S(\alpha) \). The notion of support can be strengthened to model ordered entailment of names.

Definition 1 (Strong support) For any nominal set \( X \), any \( x \in X \) and any \( S \subseteq N \), \( S \) strongly supports \( x \) if, for any permutation \( \pi, \pi \) fixes \( x \) iff \( \pi \) fixes each element in \( S \).\( \▲ \)

The notion of strong support is indeed stronger than that of support, which employs only the "if"-part of the above assertion. For example, if \( \alpha, \beta \in N \) then the set \( \{ \alpha, \beta \} \) only has weak support \( \{ \alpha, \beta \} \), whereas the list \( \alpha, \beta \) has strong support \( \{ \alpha, \beta \} \). Strong support coincides with weak support when the former exists.

3 The \( \lambda \)-calculus

The \( \lambda \)-calculus is a lambda-calculus with nominal general references. Leaving aside the use of name-lists instead of name-sets in the operational semantics, it is an extension of the \( \nu \)-calculus of Pitts and Stark [14] (and of the \( \nu \)-calculus with intref of [15, chapter 5]) using names for general references. We present its syntax in nominal sets, and thus obtain nominal notions such as name-freshness and name-permutation for free.

Definition 2 The \( \lambda \)-calculus is a functional calculus of nominal references. Its types are given as follows.

\[\begin{align*}
TY & \ni A, B ::= 1 \mid N \mid [A] \mid A \to B \mid A \otimes B
\end{align*}\]

So references to type \( A \) are of type \([A]\). Terms compose TE:

\[
\begin{align*}
\text{TE} & \ni M, N ::= x \mid \lambda x.M \mid M N \\
& \mid \text{skip} \mid \text{nil} \mid \text{pred} M \mid \text{suc} N \mid \text{return/arith} \\
& \mid \text{if}_0 M \text{ then } N_1 \text{ else } N_2 \mid \text{if}_\text{then/else} \\
& \mid \langle M, N \rangle \mid \text{fst} M \mid \text{snd} N \mid \text{pair/projections} \\
& \mid \alpha \mid \nu. M \mid \text{name/\( \nu \)-abstraction} \\
& \mid [M = N] \mid \text{name-equality test} \\
& \mid M ::= N \mid !M \mid \text{update/dereferencing}
\end{align*}
\]

TE is a nominal set in \( \text{Nom}_{TY} \): each name \( \alpha = a^A \) is taken from \( N_A \) and \( \nu. M \) stands for \( \nu(\langle \alpha \rangle) M \). Of the terms
above, the values are:

\[ V, W ::= \text{null} | \text{skip} | \alpha \mid x \mid \lambda x. M \mid (V, W) \]

The typing system involves terms in environments \( \vec{\alpha} \mid \Gamma \), where \( \vec{\alpha} \) a list of (distinct) names and \( \Gamma \) a finite set of variable-type pairs. Some of its rules are the following.

\[
\begin{align*}
\alpha \mid \Gamma, x: A & \vdash x : A \\
\vec{\alpha}, \alpha \mid \Gamma & \vdash M : B \\
\vec{\alpha} \mid \Gamma & \vdash \nu \lambda. M : B \\
\vec{\alpha} \mid \Gamma & \vdash M : [A] \\
\vec{\alpha} \mid \Gamma & \vdash N : [A] \\
\vec{\alpha} \mid \Gamma & \vdash M := N : A \\
\vec{\alpha} \mid \Gamma & \vdash M := N : A
\end{align*}
\]

The reduction calculus is defined in store environment \( S \):

\[ S ::= c \mid \alpha, S \mid \alpha ::= V, S \]

For each store environment \( S \) we define its domain, \( \text{dom}(S) \), to be the list of names stored in \( S \). We only consider environments with domains in \( \mathbb{N}^\# \) (i.e. lists of distinct names). Reduction rules are as below,

\[
\begin{align*}
\text{DRF} & \quad \text{NEW} \\
\text{UPD} & \quad \text{EQ} \\
\text{PRD} & \quad \text{LAM} \\
\text{CTX}
\end{align*}
\]

plus standard CBV rules for \text{fst}, \text{snd}, \text{id}, \text{if0}, \text{pred} and \text{succ}. Evaluation contexts \( E[\_] \) are of the forms:

\[
\begin{align*}
[\_ = N], \ [\alpha = \_], \ [\lambda x.M], \ [\_ := N], \ \alpha ::= \_ \quad \text{(\lambda x.M)}, \ [\_ := N], \ \alpha ::= \_ \quad \text{fast}, \ \_ := N, \ \alpha ::= \_ \quad \text{if0} \_ \text{then} N \text{ else } N' \\
\text{snd}\_, \ \text{pred}\_, \ \text{succ}\_, \ (\_, N), \ (V, \_) \quad \text{a pullback}
\end{align*}
\]

We take \text{observable terms} to be the constants of type \( \text{null} \), and around them we build the notions of observational equivalence.

\[
\text{Definition 3 (\( \subseteq \))} \quad \text{For typed terms} \ \vec{\alpha} \mid \Gamma \vdash M : N : A \text{ define} \ \vec{\alpha} \mid \Gamma \vdash M \subseteq N \text{ to be the assertion:}
\]

\[
\exists S: (\vec{\alpha} \mid \Gamma \vdash M \Rightarrow S = 0) \implies \exists S': (\vec{\alpha} \mid \Gamma \vdash M' \Rightarrow S = 0)
\]

\[\text{We usually omit} \ \vec{\alpha} \mid \Gamma \text{ and write simply} \ M \subseteq N.\]

\section{3.1 Semantics}

We examine sufficient conditions for a fully abstract categorical semantics of \( \text{sp} \), following a development similar to that of [14, chapter 3]. Note that, translating each term \( M \) into a morphism \( [M] \) and assuming a preorder \( \sim \) in the semantics, full-abstraction will amount to the assertion:

\[ M \trianglelefteq N \iff [M] \trianglelefteq [N] \]

\textbf{Soundness.} We examine semantics in a family of categories \( \langle M^\beta \rangle_{\beta \in \mathbb{N}^\#} \) so that each typed term \( \vec{\alpha} \mid \Gamma \vdash M : A \) is translated into a map \( [M] : [\Gamma] \rightarrow [T[A]] \) in \( M^\beta \). \( T \) is a computational monad, so our semantics is a \textit{monadic} one (v. [111]). Computation in \( \text{sp} \) is store-update and fresh-name creation. These requirements define \( \lambda_{\text{sp}} \)-models.

\[
\text{Definition 4} \quad \text{A} \lambda_{\text{sp}} \text{-model} M \text{ is a family of categories and monads} (M^\beta, (T, \eta, \mu, \tau)_{\beta \in \mathbb{N}^\#}) \text{ such that, for each} \ \vec{\alpha}:
\]

\( I. \) \( M^\beta \) has finite products, with 1 being the terminal object and \( \times B \) the product of \( A \) and \( B \).

\( II. \) \( M^\beta \) and \( (T, \eta, \mu, \tau) \) form a \( \lambda_{\text{sp}} \)-model (v. [111]). The \( T \)-exponential \( TBA \) is denoted by \( A \Rightarrow TB \).

\( III. \) \( M^\beta \) contains a natural numbers object \( N \) equipped with successor/predecessor arrows and \( n : 1 \rightarrow N \), each \( n \in N \).

\( IV. \) \( M^\beta \) contains, for each \( A \in \text{TY} \), an \( A \)-names object \( N_A \), a 0/1-valued name-equality arrow \( \text{eq}_A : N_A \otimes N_A \rightarrow N \), and, for each \( \alpha \in (N_A \cap S(\vec{\alpha})) \), an arrow \( \alpha : 1 \rightarrow N_A \). These make \( T [N_A \Delta N_A \otimes N_A] \text{ a pullback.} \)

\( V. \) Taking \( [1] \trianglelefteq 1 \), \( [N] \trianglelefteq N \), \( [[A]] \trianglelefteq N_A \), \( [[A \rightarrow B]] \trianglelefteq [A] \triangleright T[B] \) and \( [A \otimes B] \trianglelefteq [A] \otimes [B] \), \( M^\beta \) contains, for each \( A \in \text{TY} \), arrows

\[
\text{drf}_A : N_A \rightarrow [T[A]] \quad \text{and} \quad \text{upd}_A : N_A \otimes N_A \rightarrow T1
\]

\textbf{such that, for} \( \alpha \neq \beta \) and \( \text{upd}_A \trianglelefteq (\lambda x. \text{id}) ; \text{upd}_A \), \( A \) \text{ generating the following diagrams (which describe the specifications for dereferencing and updating) commute.}

\[
\begin{align*}
[A] & \xrightarrow{(1) \text{ upd}_A} [A] \otimes T1 \quad \downarrow \tau, \vdash TNA \\
T[A] & \xrightarrow{(2) \text{ drf}_A} \mu[Tdrf_A]
\end{align*}
\]

\[
\begin{align*}
[A] & \otimes [B] \xrightarrow{(1) \text{ upd}_B} [A] \otimes T1 \quad \downarrow Tdrf_B \\
T[A] & \xrightarrow{(2) \text{ drf}_A} \mu[Tdrf_A]
\end{align*}
\]

\[
\begin{align*}
A & \xrightarrow{(1) \text{ upd}_A} [A] \otimes T1 \quad \downarrow Tdrf_B \\
T[A] & \xrightarrow{(2) \text{ drf}_A} \mu[Tdrf_A]
\end{align*}
\]

\[
\begin{align*}
A & \xrightarrow{(1) \text{ upd}_B} [A] \otimes T1 \quad \downarrow Tdrf_A \\
A & \xrightarrow{(2) \text{ upd}_B} [A] \otimes T1 \quad \downarrow Tdrf_B \\
A & \xrightarrow{(1) \text{ upd}_B} [A] \otimes T1 \quad \downarrow Tdrf_A \\
T[A] & \xrightarrow{(2) \text{ drf}_A} \mu[Tdrf_A]
\end{align*}
\]

\[
\begin{align*}
T[A] & \xrightarrow{(2) \text{ drf}_A} \mu[Tdrf_A]
\end{align*}
\]

\[
\begin{align*}
T[A] & \xrightarrow{(2) \text{ drf}_A} \mu[Tdrf_A]
\end{align*}
\]

\[
\begin{align*}
T[A] & \xrightarrow{(2) \text{ drf}_A} \mu[Tdrf_A]
\end{align*}
\]
Moreover, $Ob(M^\vec{\alpha})$ is a nominal set with equivariant elements and all $M^{\vec{\alpha}}$'s contain the same objects, so we let $Ob(M) \triangleq Ob(M^{\vec{\alpha}})$, any $\vec{\alpha}$. For each $A, B \in Ob(M)$ there exists a nominal set $(A, B)$, such that

$$\mathcal{M}^{\vec{\alpha}}(A, B) = \{ (\bar{x}, \vec{\alpha}) \mid x \in \mathcal{M}(A, B) \land S(x) \subseteq S(\vec{\alpha}) \}$$

We write $f = (\langle f \rangle, \vec{\alpha})$, each $f \in \mathcal{M}^{\vec{\alpha}}(A, B)$. Moreover, the structure defined in I-V above is equivariant in the following sense:

$$S((\alpha^{(\vec{\alpha})})^\bar{\alpha} \otimes \vec{\beta}) = \varnothing \land ((\alpha^{(\vec{\alpha})})^\bar{\alpha} \otimes \vec{\beta}) = 1 \in A$$

$$S((\alpha^{(\vec{\alpha})})^\bar{\alpha} \otimes \vec{\beta}) = \varnothing \land ((\alpha^{(\vec{\alpha})})^\bar{\alpha} \otimes \vec{\beta}) = \eta_A$$

etc.

$$S(\eta^{(\vec{\alpha})}) = \{ \alpha \} \land (\alpha^{(\vec{\alpha})})^\bar{\alpha} \otimes \vec{\beta} = \alpha^\bar{\alpha} (\text{if } \alpha \neq \vec{\beta})$$

Also, for each $\alpha \neq \vec{\alpha}$ and each $A, B$, the nominal mapping

$$\langle \alpha \rangle : \mathcal{M}^{\vec{\alpha} \alpha}(A, TB) \rightarrow \mathcal{M}^{\vec{\alpha} \alpha}(A, TB)$$

such that, for all relevant $f, g, \beta$, the SN-diagrams commute:

$$f : \langle \alpha \rangle g = \langle \alpha \rangle (f^\alpha \otimes g) \land \langle \alpha \rangle f : Tg = \langle \alpha \rangle (f : T(g^\alpha))$$

$$(\alpha f : \mu) = (\alpha f : \mu) \land (\alpha \otimes f) \otimes \tau = (\alpha \otimes f) \otimes \tau$$

$$(\text{upd}_B^\vec{\alpha} \otimes \langle f \rangle) : \psi = \langle \alpha \rangle ((\text{upd}_B^\vec{\alpha} \otimes \langle f \rangle) : \psi)$$

(where $\psi = \tau \cdot T \cdot \mu$, see [11]). $\vec{\alpha}$ is name-translation and $(\alpha \circ \alpha)$ is name-abstraction, not to be confused with nominal name-abstraction $(\alpha \circ \alpha)$.

Our semantics is cast inside $\text{Nom}_{\text{ty}}$. The reason for describing morphisms as pairs $(\bar{x}, \vec{\alpha})$ comes from the fact that we give semantic translations of sequents, not terms, and sequents may contain superfluous names in their name-environments. Thus, if $f$ models $\alpha \bar{\alpha} \Gamma \vdash A$ then $(\bar{f})^\vec{\alpha}$ models $\langle \vec{\alpha} \rangle \cdot \bar{\alpha} \bar{\alpha} \Gamma \vdash A$, where $\vec{\alpha} \cdot \bar{\alpha}$ is $\vec{\alpha}$ with all names that are fresh for $M$ removed. Moreover, this description allows us to form a family of categories that have essentially the same structure, and gives us a means to relate the semantics of sequents like $\bar{\alpha} \cdot \bar{\alpha} \Gamma \vdash M : A$ and $\vec{\alpha} \cdot \bar{\alpha} \Gamma \vdash M : A$.

Recall that in a $\lambda_c$-model $M^{\vec{\alpha}}$ there exists, for each $A, B, C$, a bijection natural in $A$:

$$\Lambda^{\vec{\alpha}}_{A, B, C} : \mathcal{M}^{\vec{\alpha}}(A \otimes B, TC) \rightarrow \mathcal{M}^{\vec{\alpha}}(A, B \Rightarrow TC)$$

Let $ev_{\vec{\alpha}}(\bar{A}) : (A \Rightarrow TB) \otimes A \rightarrow TB \triangleq (\Lambda^{\vec{\alpha}})^{(1)}(id_{A \Rightarrow TB})$.

We give the semantics of $\nu\psi$ in a $\lambda_{\nu\psi}$-model $M$.

**Definition 5** Let $\langle \mathcal{M}^{\vec{\alpha}}, \mathcal{M}^{\vec{\beta}} \rangle_{\vec{\alpha} \in \text{Ne}}$ be a $\lambda_{\nu\psi}$-model. A typing judgement $\vec{\alpha} \bar{\alpha} \Gamma \vdash M : A$ is translated into an arrow $[M] : [\Gamma] \rightarrow [T(A)]$ in $M^{\vec{\alpha}}$ as follows.

$$[\bar{n}] : \Gamma \rightarrow \bar{1} \triangleq \bar{\alpha} \triangleright \triangleright \nabla \rho, TN_A \Rightarrow \bar{1} \triangleq \bar{\alpha} \triangleright \triangleright \nabla \rho, TN_A$$

$$\nu\alpha.M : \Gamma \rightarrow (A \Rightarrow TB) \Rightarrow (T(A \Rightarrow TB))$$

$$[\bar{\alpha}^\nu \psi] : \nu\alpha.M : \Gamma \rightarrow T(\langle \bar{\alpha} \rangle \Rightarrow TB) \Rightarrow T(A \Rightarrow TB)$$

$$[\bar{\alpha}^\nu \psi \Rightarrow \bar{T}] : \nu\alpha.M : \Gamma \rightarrow T \Rightarrow T$$

$$[\bar{\alpha}^\nu \psi \Rightarrow \bar{\alpha} : \bar{T}] : \nu\alpha.M : \Gamma \rightarrow T \Rightarrow T \Rightarrow T$$

$$[\bar{\alpha}^\nu \psi \Rightarrow \bar{T} \Rightarrow \bar{T}] : \nu\alpha.M : \Gamma \rightarrow T \Rightarrow T \Rightarrow T$$

plus standard translations for other term constructors.

We proceed to show correctness. Note that we write $S \Rightarrow M \Rightarrow S' \Rightarrow M'$, with $r$ being a reduction rule different from CTX, if the non-CTX rule in the derived relation is $r$. We write $M : N$ for the term $(\lambda \alpha.N)M$, some $d$ not in $N$, and relate to any store $S$ a term $\bar{S}$ of type 1 by:

$$\bar{\alpha} \triangleq \text{skip}, \bar{\alpha} \triangleq S \triangleq \bar{S}, \alpha :: V, \bar{S} \triangleq \bar{S} (\alpha :: V) \triangleright \bar{S}$$

**Proposition 6 (Correctness)** For any $\bar{\alpha} \Gamma \vdash M : A$, any $S$ with dom$(S) = \bar{\alpha}$ and any $\bar{x} \neq \text{NEW}$,

$$S \vdash M \Rightarrow S' \Rightarrow M' \Rightarrow [\bar{S} ; M] = [\bar{S}' ; M']$$

$$S \vdash M \Rightarrow S' \Rightarrow M' \Rightarrow \nu\bar{\alpha}(\bar{S} ; M) = [\nu\bar{\alpha}^{\triangleright} ; (\bar{S} ; M')]$$

Hence, $S \vdash M \Rightarrow S' \Rightarrow M' \Rightarrow \nu\bar{\alpha}(\bar{S} ; M) = [\nu\bar{\alpha}^{\triangleright} ; (\bar{S} ; M')]$, with dom$(S') = \bar{\alpha}$. 

Soundness does not follow from correctness; we need to add an adequacy specification.

**Definition 7 (Adequacy)** Let $M = \langle \mathcal{M}^{\vec{\alpha}}, T^{\vec{\alpha}} \rangle_{\vec{\alpha} \in \text{Ne}}$ be a $\lambda_{\nu\psi}$-model and $[\cdot]$ the respective translation of $\nu\psi$. $M$ is adequate if, for any typed term $\bar{\alpha} \bar{\alpha} \Gamma \vdash M : N$, if $[M] = \langle \bar{\alpha} \rangle \Rightarrow \bar{S} ; 0 \rangle$, some $S$, then there exists $S'$ such that $\bar{\alpha} \bar{\alpha} \Gamma \vdash S' \Rightarrow S' \Rightarrow 0$.

Assume now our running $M$ is an adequate $\lambda_{\nu\psi}$-model.

**Proposition 8 (Equational Soundness)**

$$[M] : N \Rightarrow M \Rightarrow N$$

**Completeness.** To achieve completeness we need to introduce a preorder in the semantics to match the observational preorder of the syntax, as in (FA). This step, which is essentially a quotiening procedure, is found in many (but by no means all) fully abstract models based on game semantics.
Definition 9 (p-Observability)  An adequate \( \lambda_{op} \)-
model \( \mathcal{M} \) is (reorder)-observational if, for all \( \vec{x} \):

- There exists \( O^\vec{x} \subseteq \mathcal{M}^\vec{x}([1,T\mathbb{N}]) \) such that, for all \( \vec{\alpha} \models \varnothing \models \mathcal{M} \):
  \[
  [M] \in O^\vec{x} \iff \exists S, \vec{\beta}. [M] = \langle \vec{\beta} \rangle [S ; 0] \quad (9a)
  \]
- The induced intrinsic preorder on arrows in \( \mathcal{M}^\vec{x}(A,TB) \), defined by \( f \lessdot^\vec{x} g \iff \forall \rho : A \Rightarrow TB \rightarrow T\mathbb{N}. (\Lambda^T(f) ; \rho \in O^\vec{x} \Rightarrow \Lambda^T(g) ; \rho \in O^\vec{x}) \)
  satisfies, for all \( \alpha \# \vec{\alpha} \) and relevant \( f, g, f', g' \):  
  \[
  f \lessdot^\vec{x} g \Rightarrow f^+ \lessdot^\vec{x} g^+ \quad (9b)
  \]
  \[
  f' \lessdot^\vec{x} g' \Rightarrow (\alpha f') \lessdot^\vec{x} (\alpha g') \quad (9c)
  \]

We write \( \mathcal{M} \) as \( (\mathcal{M}^\vec{x}, T^\vec{x}, O^\vec{x}, \lessdot^\vec{x}) \subseteq \mathbb{N} \).

So \( O^\vec{x} \) contains those arrows that have an observable behavior in the model, and the semantic preorder is built around this notion. In particular, due to (9a), terms that yield \( 0 \) have observable behavior. On the other hand, by (9b) and (9c) we have that \( \lessdot^\vec{x} \) is a congruence, i.e. it passes through contexts. Now assume our running model is p-observational.

Lemma 10 (Inequational Soundness)
\[
[M] \subseteq [N] \implies M \lessdot N
\]

In order to achieve completeness, and hence full-abstraction, we need some definability requirement.

Definition 11 (p-Definability) A p-observational \( \lambda_{op} \)-
model \( \mathcal{M} \) satisfies p-definability if, for any \( \vec{\alpha}, A, B, \) there exists \( D^\vec{\alpha}_{A,B} \subseteq \mathcal{M}^\vec{\alpha}([A], T[B]) \) such that:

- For each \( f \in D^\vec{\alpha}_{A,B} \) there exists term \( M \) with \( [M] = f \).
- For any \( f, g \in D^\vec{\alpha}_{A,B} \), \( f \lessdot^\vec{x} g \) iff \( \forall \rho \in D^\vec{\alpha}_{A,B,N}. (\Lambda^T(f) ; \rho \in O^\vec{x} \Rightarrow \Lambda^T(g) ; \rho \in O^\vec{x}) \)

P-Definability states that definable test-arrows suffice for defining the semantic preorder. Now assume our model satisfies p-definability.

Proposition 12 (Full-Abstraction)
\[
[M] \subseteq [N] \iff M \lessdot N
\]

Proof: Soundness is by previous lemma. For completeness, \( \langle \lessdot \rangle \), assume \( \vec{\alpha} \models \varnothing \models \mathcal{M} \lessdot N \); we do induction on the size of \( \Gamma \). The base case of \( \Gamma = \varnothing \) is encompassed in the case of \( \Gamma = \{ x : A \} \) -just add a dummy \( x \), which we now show. Suppose \( \vec{\alpha} \models x : A \models N \) and take any \( \rho \in D^\vec{\alpha}_{A,B,N} \) such that \( \Lambda^T([M]) ; \rho \in O^\vec{x} \). Let \( \rho = [\vec{\alpha} \models y : A \models B \models L] \), some \( L \), so \( \Lambda^T([M]) ; \rho \in O^\vec{x} \). The latter being in \( O^\vec{x} \) implies that it equals \( \langle \vec{\beta} \rangle [S ; 0] \), some \( S \). Now, \( M \lessdot N \) implies \( \langle \lambda y.L(\lambda x.M) \rangle \lessdot \langle \lambda y.L(\lambda x.N) \rangle \), hence \( \nu \langle \vec{\beta} \rangle (\vec{S} ; 0) \lessdot \langle \lambda y.L(\lambda x.N) \rangle \), by soundness. But this implies that \( \vec{\alpha} \models (\lambda y.L(\lambda x.N) \models S' \models 0) \), so \( [(\lambda y.L(\lambda x.N)) \lessdot N] \in O^\vec{x} \), by correctness. Hence, \( \Lambda^T([N]) ; \rho \in O^\vec{x} \), \( [M] \lessdot [N] \), by p-definability.

For the inductive step, let \( \Gamma = x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n \), and let \( z \) not appear in \( \Gamma \); if \( \vec{\alpha} \models \varnothing \models \mathcal{M} \lessdot N \) then \( \vec{g} \models z : A_1 \otimes A_2 \models T' \models T[B] \), i.e. \( \vec{g} \models z : A_1 \otimes A_2 \models T' \models T[B] \), and let \( \vec{\alpha} \models \varnothing \models \mathcal{M} \lessdot N \) as required.

4 The nominal names model

We embark on the adventure of modeling \( \lambda_{op} \) in a category of nominal arenas and strategies. Our presentation of nominal names rectifies the one presented in [2] by using name-lists instead of name-sets and introducing innocent plays. The basic category construction will be \( \mathbf{Nom}^\vec{x} \), the category of nominal arenas and total \( \vec{\alpha} \)-strategies. \( \mathbf{Nom}^\vec{x} \) will be constructed in \( \mathbf{Nom}_{TY} \), so there will be, for each type \( A \), an arena \( N_A \) for references to type \( A \). The translation \( [A] \) of a general type will make use of a store arena \( \xi = \bigotimes_A (N_A \Rightarrow [A]) \), which will literally serve as a reference store. This will naturally lead us to a monad semantics, with computation monad \( T \) defined (on arenas) by \( T.A \models \xi \models A \otimes \xi \). Since arrow types involve the monad in their translation and the monad involves all types, we will have to first solve the domain equation:

\[
[A \rightarrow B] = [A] \Rightarrow (\xi \Rightarrow [B] \otimes \xi)
\]

\[
\xi = \bigotimes_A (N_A \Rightarrow [A]) 
\]

Construct a category. We assume a set of types \( TY \) and build our constructions inside \( \mathbf{Nom}_{TY} \). We start with nominal arenas and prearenas. Arenas will be used for type-translation, while terms will be translated to strategies between arenas, i.e. strategies for prearenas.

Definition 13 A nominal arena \( A \models (M, \models A, \models \lambda_A) \) contains:

- A nominal set of moves \( M_A \), the elements of which have strong support.
- A nominal justification relation \( \models A \subseteq (M_A \times [1]) \times M_A \).
- A nominal labeling function \( \lambda_A : M_A \rightarrow ([O,P] \times [Q,A]) \), so that each move can be played by Opponent or Player, and is a Question or an Answer.

\[\text{Notation will be clarified below.}\]

\[\text{Strong support is essential in proving basic properties of nominal games, e.g. that these form a category, yet their proofs are omitted here.}\]
These satisfy the conditions:

(f) For each \( m \in M_A \), there exists unique \( k \geq 0 \) such that \( \vdash_A m_1 \vdash_A \cdots \vdash_A m_k \vdash_A m \), for some \( m_i \)'s in \( M_A \). \( k \) is called the level of \( m \).

Level-0 moves, denoted by \( i, i', \ldots \), are called initial.

(11) Initial moves are P-Answers.

(12) If \( m_1, m_2 \in M_A \) are at consecutive levels then \( \lambda_A \) assigns them complementary OP-labels.

(13) Answers may only justify Questions.

A prearena is an arena with its initial moves labeled OQ. Given arenas \( A \) and \( B \), construct the prearena \( A \rightarrow B \) as:

\[
M_{A \rightarrow B} \triangleq M_A + M_B, \\
\lambda_{A \rightarrow B} \triangleq [(i_A \mapsto OQ, \overrightarrow{\lambda_A(m_A)})], \quad \lambda_B]
\]

\( \vdash_{A \rightarrow B} \triangleq \{(i, i_A), (i_A, i_B)\} \cup \{(m, n) \mid m \vdash_{A \rightarrow B} n\}. \uparrow \) \( I_q \) is the set of initial (level-0) moves of \( A \), and \( J_A \) the set of initial (level-1) moves. Then, \( \overrightarrow{A} = M_A \setminus I_A \), and \( \overrightarrow{J_A} = M_A \setminus J_A. \) In general, we use \( m \) to denote moves in \( M_{A \rightarrow B} \) for \( m \in I_A \cup I_B \) for moves in \( \overrightarrow{A} \), and \( m \) in \( J_A \) for moves in \( J_A \). Finally, \( \lambda_A \) denotes the OP-complement of \( \lambda_A \).

Condition (f) states that arenas can be represented by directed connected graphs with no directed cycles. Note that the nominal arenas of [2] do satisfy the above conditions, although a different set of conditions is used there.

Example 14 (Basic arenas) The simplest arena is \( 0 = (\emptyset, \emptyset, \emptyset) \). Now let \( A \) be an arbitrary type. Define the (flat) arenas \( N_A, N \) and \( 1 \) as follows.

\[
M_{N_A} \triangleq N_A, \quad M_{N} \triangleq N, \quad M_1 \triangleq \{\ast\} \\
\lambda_{N_A}(m) \triangleq PA, \quad \lambda_N(m) \triangleq PA, \quad \lambda_1(\ast) \triangleq PA \\
\vdash_{N_A} \triangleq \{(\ast, m)\} \cup \{m \mid 1 \vdash_{N} \}
\]

Nominal games are played using sequences of moves-with-names, that is moves attached with name-lists. Name-lists capture name-environments; this idea of attaching state-names was later followed in [2].

Definition 15 A move-with-names of a (pre)arena \( A \) is a pair, \( m, \vec{\alpha} \), where \( m \in M_A \) and \( \vec{\alpha} \in N_A \) (i.e. \( \vec{\alpha} \) a name-list).

Writing \( m, \vec{\alpha} \) as \( x \), we have \( x \equiv m \) and \( nlist(x) \equiv \vec{\alpha} \). ▲

At this point, let us introduce some handy notation for sequences. Let \( s, t \) be sequences, then:

- \( s \in t \) denotes that \( s \) is a prefix of \( t \), and then \( t = s(t - s) \).
- \( s^\rightarrow \) denotes \( s \) with its last element removed.
- if \( s = s_1 \cdots s_n \) then
  \( o \) \( n \) is the length of \( s \), and is denoted by \( |s| \).\( o \)
  \( s.i \) denotes \( s_1 \) and \( s.i^\rightarrow \) denotes \( s_{n+1} \cdots \), e.g. \( s.i \) is \( s_n \).
  \( o \)
  \( s.s \) denotes \( s_1 \cdots s_n \), and so does \( s.s.i \).

A justified sequence over a prearena \( A \) is a finite sequence \( s \) of OP-alternating moves such that, except for \( s.1 \) which is initial, every move \( s.i \) has a justification pointer to some \( s.j \) such that \( j < i \) and \( s.j \vdash_A s.i \). We say that \( s.j \) (explicitly) justifies \( s.i \). We can now proceed to plays.

Definition 16 (Plays) Let \( A \) be a prearena. A legal sequence on \( A \) is a justified sequence of moves-with-names that satisfies Visibility and Well-Bracketing (v. [10, 7]). A legal sequence \( s \) is a play if it also satisfies the following Name Change conditions:

- (NC1) The name-list of a P-move \( x \) in \( s \) contains as a prefix the name-list of its preceding O-move. It possibly contains other names, all of which are fresh for \( s_{<x} \).
- (NC2) Any name in the support of a P-move \( x \) in \( s \) that is fresh for \( s_{<x} \) is contained in the name-list of \( x \).
- (NC3) The name-list of a non-initial O-move \( x \) in \( s \) is that of the P-move explicitly justifying it.

An \( \vec{\alpha} \)-play is a play that opens with a move with name-list \( \vec{\alpha} \). The set of \( \vec{\alpha} \)-plays on a prearena \( A \) is denoted by \( P^\vec{\alpha}_A \). ▲

With \( s \) and \( x \) as above, \( P \) introduces a name \( o \alpha \in s \) iff \( o \|\| o \alpha \|\| x \in \sigma \) and \( o\#o\alpha \#o\alpha \). \( L(s) \) contains all names introduced by \( P \) in \( s \). Note also that, for any move \( x \) in an \( \vec{\alpha} \)-play, \( \vec{\alpha} \leq \#nlist(x) \).

We proceed to strategies.

Definition 17 (Strategies) An \( \vec{\alpha} \)-strategy \( \sigma \) is a set of equivalence classes \( [s]_\vec{\alpha} \) of \( \vec{\alpha} \)-plays, written \( [s] \), satisfying prefix closure, contingency completeness and determinacy:

- If \( [su] \in \sigma \) then \( [s] \in \sigma \).
- If even-length \( [s] \in \sigma \) and \( sx \) is an \( \vec{\alpha} \)-play then \( [sx] \in \sigma \).
- If even-length \( [s_1 x_1], [s_2 x_2] \in \sigma \) and \( [s_1] = [s_2] \) then \( [s_1 x_1] = [s_2 x_2] \).

An \( \vec{\alpha} \)-strategy \( \sigma \) on \( A \rightarrow B \) is written \( \sigma : A \rightarrow B \).

For example, for \( o\#\vec{\alpha} \) and \( n \in \mathbb{N} \), define the \( \vec{\alpha} \)-strategies:

\[
\alpha : 1 \rightarrow N_A \triangleq \{[s^\vec{\alpha} o\#\vec{\alpha}]\} \quad \text{and} \quad \vec{\alpha} : 1 \rightarrow \mathbb{N} \triangleq \{[s^\vec{\alpha} o\#\vec{\alpha}^\rightarrow]\}
\]

Note that \( \vec{\alpha} \)-strategies have (strong) support \( \#nlist(\vec{\alpha}) \). We define play- and strategy-composition building on [6, 10]. We let \( \vec{s} = s \) without its name-lists, and \( \#nlist(s) \) be \( s \).

Definition 18 (Composable plays) Let \( s \in P^\vec{\alpha}_A \rightarrow B \) and \( t \in P^\vec{\alpha}_B \rightarrow C \). These are almost composable, \( s \bowtie t \), if \( \delta \mid B = \delta \mid B \). They are composable, \( s \bowtie t \), if \( s \bowtie t \) and, for any \( s' \leq s \) and \( t' \leq t \) with \( s' \bowtie t' \),

- (C1) If \( s' \) ends in a P-move in \( A \) introducing some name \( o \alpha \) then \( o\#o\alpha \#t' \); dually, if \( t' \) ends in a P-move in \( C \) introducing some name \( o \alpha \) then \( o\#o\alpha \). ▲

- (C2) If both \( s', t' \) end in \( B \) and \( s' \) ends in a P-move introducing some name \( o \alpha \) then \( o\#o\alpha \#s' \). ▲
If \( s \in P_{A \to B}^\mathfrak{a} \) and \( t \in P_{B \to C}^\mathfrak{a} \) are composable then either \( s \vdash B = t = \varepsilon \), or \( s \) ends in \( A \) and \( t \) in \( B \), or \( s \) ends in \( B \) and \( t \in C \), or both \( s \) and \( t \) end in \( B \) (cf. zipper lemma of [6]). In the following we state that \( m \) is an O-move by writing \( m_{(O)} \), and similarly for P-moves.

**Definition 19 (Composition)** Let \( s \in P_{A \to B}^\mathfrak{a} \) and \( t \in P_{B \to C}^\mathfrak{a} \) with \( s \equiv t \). Their parallel interaction \( s \parallel t \) and their mix \( s \bullet t \), which returns the name-list of the final move in \( s \parallel t \), are defined by mutual induction below.

\[
\begin{align*}
\epsilon & \triangleq \epsilon \\
sm_{B}^{\mathfrak{a}} \parallel tm_{B}^{\mathfrak{a}} & \triangleq (s \parallel t)m_{B}^{\mathfrak{a} \bullet tm_{B}^{\mathfrak{a}}} \\
s \parallel tm_{C}^{\mathfrak{a}} & \triangleq (s \parallel t)m_{C}^{s \bullet tm_{C}^{\mathfrak{a}}} \\
s \bullet tm_{C}^{\mathfrak{a}(O)} & \triangleq \gamma'' \\
sm_{A(O)}^{\mathfrak{a}} \bullet t & \triangleq \gamma'' \\
sm_{A(P)}^{\mathfrak{a}} \bullet t & \triangleq s \bullet t, \tilde{\beta} \\
sm^{\mathfrak{a}}_{B(P)} \bullet t & \triangleq s \bullet t, \tilde{\beta} \\
sm^{\mathfrak{a}}_{B(O)} \bullet t & \triangleq s \bullet t, \gamma'' \\
sm^{\mathfrak{a}}_{B(O)} \bullet t & \triangleq s \bullet t, \gamma''
\end{align*}
\]

where \( \tilde{\beta} = \tilde{\beta} - \text{nlist}(s, -1) \), and \( \gamma'' \) is the name-list of \( m_{(O)(a)} \)’s justifier in \( s \parallel t \); similarly for \( \gamma'' \).

The composition of \( s \) and \( t \) is:
\[
s \vdash (s \parallel t) \equiv AC.
\]

For \( \vec{\sigma} \)-strategies \( \sigma : A \to B \) and \( \tau : B \to C \), their composition is:
\[
\sigma \vdash \tau \triangleq \{ s; t \} | [s \in \sigma \land t \in \tau \land s \equiv t \}.
\]

**Proposition 20** If \( s \in P_{A \to B}^\mathfrak{a} \) and \( t \in P_{B \to C}^\mathfrak{a} \) with \( s \equiv t \), then \( s \vdash (s \parallel t) \equiv AC \).

If \( \sigma : A \to B \) and \( \tau \vdash (s \parallel t) \equiv AC \) are \( \vec{\sigma} \)-strategies then \( \sigma \vdash \tau \). Moreover, if \( \sigma_1 : A_1 \to A_2 \), \( \sigma_2 : A_2 \to A_3 \) and \( \sigma_3 : A_3 \to A_4 \) are \( \vec{\sigma} \)-strategies then \( \sigma_1 ; \sigma_2 ; \sigma_3 = \sigma_1 ; (\sigma_2 ; \sigma_3) \).

We are interested in innocent strategies, that is strategies in which P-moves depend solely on current P-views. Recall that the P-view, \( \vec{r}^{\mathfrak{s}} \), of a justified sequence \( s \) is:

\[
\begin{align*}
\vec{r}^{s, x} & \triangleq \vec{r}^{s} | x \\
\vec{r}^{s} & \triangleq \varepsilon \\
\vec{r}^{s,sx'y} & \triangleq \vec{r}^{s} | xy \\
\end{align*}
\]

Note that the P-view of a play is not necessarily itself a play; hence, we further restrict plays.

**Definition 21** A play \( s \) is innocent if, for any \( t \leq s \), \( \vec{r}^{s} \) is a play.

It is not difficult to see that innocent plays are legal sequences satisfying (NC1), (NC3) and (NC2’). Any name in the support of a P-move \( x \) in \( s \) that is fresh for \( \vec{s}^{< \mathfrak{s}^{\mathfrak{a}}} \) is contained the name-list of \( x \). From innocent plays we move on to innocent strategies.

**Definition 22** An \( \vec{\sigma} \)-strategy \( \sigma \) is innocent if \( s \in \sigma \) implies that \( s \) is innocent, and if even-length \( \lceil s_1 \rceil \in \sigma \) and odd-length \( \lfloor s_2 \rfloor \in \sigma \) have \( [s_1] = [s_2] \) then there exists \( n_2 \) such that \( [s_2 n_2] \in \sigma \) and \( [s_1 n_2] = [s_2 n_2] \).

**Proposition 23** If \( s \in P_{A \to B}^\mathfrak{a} \), \( t \in P_{B \to C}^\mathfrak{a} \) are innocent and \( s \equiv t \) then \( s \vdash t \) is innocent. If \( \sigma : A \to B, \tau : B \to C \) are innocent \( \vec{\sigma} \)-strategies then \( \sigma \vdash \tau \).
Moreover, let \( A \Rightarrow B \triangleq A \Rightarrow B \), be the lifted function space. Note that we will usually identify graph-isomorphic arenas related by isomorphisms which simply manipulate *s. With this convention, the last construction corresponds precisely to \( A \Rightarrow B \) of [2]; also, for any \( A, 1 \Rightarrow A = A \). The previous constructions are sketched below.

\[
\begin{array}{c}
A \Rightarrow B \\
\text{natural in } A, C. \text{ Moreover, } 1 \text{ is a terminal object and } \otimes \text{ is a product constructor in } \mathcal{V}_\otimes, \text{ so } \mathcal{V}_\otimes \text{ has finite products.}
\end{array}
\]

We also have arrow-counterparts. Let \( f : A \to A', g : B \to B' \) in \( \mathcal{V}_\otimes \) and \( h : B \to B' \) in \( \mathcal{V}_\otimes \), then

- \( f_! : A_\perp \to A'_\perp \) initially plays a sequence of asterisks \([*_1^*_1 \cdots *^*_2]^2\) and then continues playing like \( f \).
- \( f \circ g : A \circ B \to A' \circ B' \) answers initial moves \([([a_i,b_i])^2]\) with \( f \)'s answer to \([i^*_1]^2\) and \( g \)'s answer to \([i^*_2]^2\). Then, according to whether Opponent plays in \( J_A \), or in \( J_B \). Player plays like \( f \) or like \( g \) respectively.
- \( f \Rightarrow h : A' \Rightarrow B \Rightarrow B' \) answers initial moves \([i^*_1]^2\) like \( h \) and then responds to \([i^*_1][i^*_2,j^*_2]_B\) with \( f \)'s answer to \([i^*_1][i^*_2]^2\) and \( h \)'s response to \([i^*_1]^2[B,j^*_2]_B\) (hence the need for totality of \( h \)). It then plays like \( f \) or like \( h \), according to Opponent’s next move.

We can also define infinite tensor products of pointed arenas, where an arena \( A \) is \textbf{pointed} iff \( I_A \) is singleton (in which case the unique initial move is necessarily equivariant). For pointed arenas \( \{A_i\}_{i \in \omega} \) construct their product \( \bigotimes A_i \) by ‘gluing them together’ at their initial moves. Since these are equivariant, the resulting initial move is also equivariant, and we denote it by “*". For any pointed \( A_i \)’s and \( B_i \)’s and any \( \{f_i : A_i \to B_i\}_{i \in \omega} \) define:

\[
\bigotimes f_i \triangleq \text{strat}(\ast_1 \ast_1 \ast_2 \ast_2) \mid \exists k : \{f_i^*_1 \ast_2^k \ast_2^k \} \in \text{view}(f_i)
\]

Take \( \mathcal{V}_\ast \) to be the full subcategory of \( \mathcal{V}_\otimes \) of pointed arenas.

Our constructions enjoy the following properties.

\textbf{Proposition 26} All of the following are functors.

\[
\begin{align*}
\otimes - & : \mathcal{V}_\otimes \times \mathcal{V}_\otimes \to \mathcal{V}_\otimes, \\
\otimes - & : (\mathcal{V}_\otimes)^{op} \times \mathcal{V}_\otimes \to \mathcal{V}_\otimes \\
(\perp) & : \mathcal{V}_\otimes \to \mathcal{V}_\perp \\
\otimes - & : \mathcal{V}_\otimes \to \mathcal{V}_\otimes \\
\otimes - & : \prod_{\mathcal{V}_\ast} \to \mathcal{V}_\ast
\end{align*}
\]

Moreover, \( \mathcal{V}_\otimes \) is a symmetric monoidal category under \( \otimes \), and is partially closed in the following sense. For any object \( B \), the functor \( \otimes B : \mathcal{V}_\otimes \to \mathcal{V}_\otimes \) has a partial right adjoint \( B \Rightarrow - : \mathcal{V}_\otimes \to \mathcal{V}_\perp \), that is for any object \( A \) and any pointed object \( A \) there exists a bijection

\[
\Lambda_{\otimes B, C}^A : \mathcal{V}_\ast(A \otimes B, C) \cong \mathcal{V}_\ast(A, B \Rightarrow C)
\]

\textbf{Solving (SE).} The full form of (SE) is the following.

\[
[1] = 1, \quad [N] = N, \quad [A] = N_A, \quad [A \circ B] = [A] \otimes [B] \\
[A \Rightarrow B] = [A] \Rightarrow (\xi \Rightarrow [B] \circ \xi) \quad \xi = \bigotimes_A (N_A \Rightarrow [A])
\]

To solve it, we will upgrade it to a recursive functor equation and then recur to minimal-invariants theory for games (v. [10]). Let us first define the following preorder on games.

\textbf{Definition 27} For any \( A, B \in \text{Ob}(\mathcal{V}_\otimes) \) and any \( \sigma, \tau \in \mathcal{V}_\otimes(A, B) \), define

\[
A \leq B \iff M_A \subseteq M_B \land \lambda_A \subseteq \lambda_B \land \top_A \subseteq \top_B \\
\sigma \subseteq \tau \iff \sigma \subseteq \tau
\]

\[\triangleq\]

It follows that \( \mathcal{V}_\otimes \) is PreCpo-enriched, with \( \bigsqcup_{\sigma} \sigma = \bigcup_{\sigma} \sigma \), for any \( \omega \)-chain \( \{\sigma_i\}_{i \in \omega} \), and that \( \text{Ob}(\mathcal{V}_\otimes) \) is a cpo, with least \( \bot \triangleq (\varnothing, \varnothing, \varnothing) \), and \( \bigsqcup_{\sigma} \sigma = (\bigcup_{\sigma} M_A, \bigcup_{\sigma} \lambda_A, \bigcup_{\sigma} \top_A) \) for any \( \omega \)-chain \( \{\sigma_i\}_{i \in \omega} \). Moreover, if \( A \leq B \) then we can define an embedding-projection pair of copycat maps \( \text{incl}_{A,B} : A \to B \) and \( \text{proj}_{A,B} : A \to B \).

Let \( C^\ast \triangleq \mathcal{V}_\ast \times \prod_{\mathcal{A} \in \text{TV}} \mathcal{V}_\ast \), with objects \( D \) of the form \( (D, A \in \text{TV}) \) and arrows \( f \) of the form \( (j_x, j_A \in \text{TV}) \).

Define \( F : (C^\ast)^{op} \times C^\ast \to C^\ast \) on objects by taking \( F(D, E) \triangleq (\xi_{D,E}, [A]_{D,E} \mathcal{A} \text{TV}) \), where

\[
\xi_{D,E} = \bigotimes_{\mathcal{A} \in \text{TV}} (N_A \Rightarrow E_A) \quad [A]_{D,E} = N_A
\]

\( [A \circ B]_{D,E} = [A]_{D,E} \circ [B]_{D,E} \quad [N]_D = N \quad [A \Rightarrow B]_{D,E} = D_A \Rightarrow (\xi_{E,D} \Rightarrow E_B \circ \xi_{E,D}) \quad [1]_{D,E} = 1 \)

and similarly for \( F(f,g) \triangleq (\xi_{f,g}, [A]_f \mathcal{A} \text{TV}) \). Now (SE) has reduced to \( D \cong F(D, D) \). We can show that \( F \) is a locally continuous functor, and continuous wrt \( \leq \). Hence the following.

\textbf{Theorem 28} In \( C^\ast \) we can form a \( \leq \)-increasing sequence \( \{\epsilon_i : D_i \to D_{i+1}\}_{i \in \omega} \) of objects and embeddings as follows.

\[
\begin{align*}
D_{0,0} &= D_{0,0} \Rightarrow 1 \quad D_{0,0} \neq N, D_{0,0} \neq N_A \\
D_{0,0} &= \bigotimes_{\mathcal{A} \in \text{TV}} (N_A \Rightarrow 0) \\
D_{i+1} &= F(D_i, D_{i+1}) \quad \epsilon_{i+1} \triangleq \text{incl}_{D_{i}, D_{i+1}} \quad \epsilon_i \triangleq F(e_i^R, e_i)
\end{align*}
\]

Taking \( D^* \triangleq \bigcup_i D_i \) and, for each \( i, \epsilon_i \triangleq \text{incl}_{D_i, D^*} \) we obtain a local bilitim \( (D^*, \epsilon_i) \).

Hence, \( D^* \) is the canonical solution to \( D \cong F(D, D) \), and it solves (SE) with the following notation.

\textbf{Definition 29} (\( \xi, \otimes \) and \( [A] \)) Let \( D^* \) be as in the previous theorem. Define the store arena \( \xi \) to be \( D^*_\xi \) and, for each type \( A \), the translation \( [A] \) to be \( D^*_A \).

\( \xi \) is pointed; we denote its unique initial move by \( \otimes \).
Tidy strategies. Using the solution $D^*$ to (SE) we can model $sp$ in the family $\langle \mathcal{V}^\xi_T, T^\xi \rangle_{\xi \in \text{Gen}^\#}$, with $T$ being the store monad on $\xi$ (i.e., $T = \xi \Rightarrow \_ \oplus \xi$). However, thus we do not obtain a fully abstract model. In the reduction calculus the treatment of the store follows a specific store-discipline; for example, if a store $S$ is updated to $S'$ then the original store $S$ is not accessible any more. In strategies we do not have such a condition: in a play there may be several $\xi'$s opened, yet there is no discipline on which of these are accessible to Player whenever he makes a move. Another condition is that, when the store is asked a name, it either returns its value or it deadlocks; there is no third option. In a play, however, when Opponent asks the value of some name, Player is free to evade answering and play elsewhere. We will therefore constrain total strategies with further conditions, defining tidy strategies.

Let $\mathcal{V}^\xi_{A,TY}$ be the full subcategory of $\mathcal{V}^\xi_T$ with objects $[A]$, $A \in TY$. For each $[A]$ let its set of store-Handles, $H_A$, be:\n
$$H_{A\rightarrow B} \triangleq H_A \cup H_B \quad H_{A} = H_1 = H_A \triangleq \emptyset$$

$$H_{A\rightarrow B} \triangleq \{(a_A, \xi_A), (b_B, \xi_B)\} \cup H_A \cup H_B \cup H_{\xi_A} \cup H_{\xi_B}$$

where we let $[A \rightarrow B]$ be $[A] \Rightarrow ([\xi_A \Rightarrow [B] \oplus \xi_B])$, and $H_\xi = \cup C H_C$ if $\xi = \bigotimes C(N_C \Rightarrow C)$. In any arena $[A]$, a store-$H$ justifies name-questions $\alpha$, which we called store-Questions. Answers to store-$Q$-s are called store-Answers. For example:

$$T_1 = \xi \Rightarrow 1 \oplus \xi \quad \text{(note } T_1 = 1 \Rightarrow T_1)$$

We can show that a move $m \in M_{[A]}$ is exactly one of the following: initial, store-$H$, store-$Q$ or store-$A$.

As store-$H$-s occur in several places in a play, we may use parenthesised indices to distinguish moves from different store-$H$-s. For example, a store-$Q$ $w$ may be denoted $w_\xi$ or $w_{(P)}$, the notation denoting also the OP-polarity. Note also that from now on we work in $\mathcal{V}^\xi_{A,TY}$, unless stated otherwise.

**Definition 30 (Tidy strategies)** A total $\tilde{\alpha}$-strategy $\sigma$ is tidy if whenever odd-length $[s] \in \sigma$ then:

(TD1) If $s$ ends in a store-$Q$ $w$ then $[sx] \in \sigma$, with $x$ being either a store-$A$ to $w$ introducing no new names, or a copy of $w$. In particular, if $w = \alpha^\tilde{\alpha}$ with $\alpha^\# \tilde{s}$ then the latter case holds.

(TD2) If $[sxw_{(P)}] \in \sigma$ with $w$ a store-$Q$ then $w_{(P)}$ is justified by last O-store-$H$ in $\tilde{s}$.

(TD3) If $\tilde{s} = s'w_{\xi}(D_{(P)})w_{y(O)}$ with $w$ a store-$Q$ then $[sy_{(P)}] \in \sigma$ with $y_{(P)}$ justified by $\tilde{s}$.

TD1 states that, whenever O(pponent) asks the value of a name, P either immediately answers with its value or copies the question to the previous store-$H$. The former case corresponds to P having updated the given name lately (i.e. between the previous O-store-$H$ and the last one), while the latter to P not having done so, and hence asking its value to the previous store configuration. Hence, the current store is, in fact, composed by layers of stores—one on top of the other—and only when a name has not been updated at the top layer is P allowed to search for it in layers underneath. TD3 further guarantees the above-described behavior. It states that when P starts a store-$Q$ copycat then he must copycat the store-$A$ he receives and all proceeding moves. TD2 guarantees the multi-layer discipline of the store: P can only see the store-$H$ played last by O in the P-view.

**Proposition 31** If $\sigma : A \rightarrow B$ and $\tau : B \rightarrow C$ are tidy strategies then so is $\sigma ; \tau$.

**Full-abstraction with tidy strategies.** Let $T^\xi$ be the lluf subcategory of $\mathcal{V}^\xi_{A,TY}$ of tidy strategies. $T^\xi$ inherits finite products from $\mathcal{V}^\xi_T$. Moreover, the endofunctor $T : \mathcal{V}^\xi_T \rightarrow \mathcal{V}^\xi_T \triangleq \xi \Rightarrow \_ \oplus \xi$ restricts to $T^\xi$, and induces a strong monad $(T, \eta, \mu, \tau)^{\tilde{\alpha}}$ on it (by a more-or-less standard monad construction). Furthermore, setting $(TB)^A \triangleq A \Rightarrow TB$ we obtain a $\lambda_e$-model.

We take $T \triangleq (T^\xi, (T, \eta, \mu, \tau)^{\tilde{\alpha}})_{\xi \in \text{Gen}^\#}$ and proceed to update and dereferencing arrows.

**Definition 32** In $T^\tilde{\alpha}$, define $\text{upd}_A : N_A \otimes [A] \rightarrow T_1$ and $\text{drf}_A : N_A \rightarrow T[\]A], for any type A, as follows.\n
$$\text{upd}_A(N_A \otimes [A]) \rightarrow T_1$$

$$\alpha^\tilde{\alpha} \rightarrow T[A]$$

(where $\alpha^\# \beta$)

**Proposition 33** In each $T^\tilde{\alpha}$ the NR-diagrams of definition 4 commute.
We introduce name-abstraction and name-addition transformations for nominal strategies.

**Definition 34** Let \( f: A \to B \) in \( T^{\vec{\alpha}, \vec{\beta}} \) and \( g: A \to B \) in \( T^{\vec{\alpha}} \). Define \((\alpha) f: A \to B \) in \( T^{\vec{\alpha}} \) and \( g^{+\alpha}: A \to B \) in \( T^{\vec{\alpha}, \vec{\alpha}} \) as:

\[
\langle \alpha \rangle f \triangleq \text{strat}\{|i A^{\vec{\alpha}} B^{\vec{\beta}}\mid [i A^{\vec{\alpha}} B^{\vec{\beta}}] \in \text{view}(f) \land \alpha \neq \lambda i \lambda s\} \\
g^{+\alpha} \triangleq \{[s^{+\alpha}] \mid [s] \in g \land \alpha \neq \lambda \langle L \rangle(s)\}
\]

where \( s^{+\alpha} \) is \( s \) with \( \vec{\beta} \) replaced by \( \vec{\alpha} \), \( \alpha \) in its name-lists. ▲

**Proposition 35** For any \( \vec{\alpha}, \vec{\alpha} \)-strategy \( f \) and any \( \vec{\alpha} \)-strategy \( g \), \( \langle \alpha \rangle f \) is an \( \vec{\alpha} \)-strategy and \( g^{+\alpha} \) is an \( \vec{\alpha}, \vec{\alpha} \)-strategy. Moreover, \( T \) satisfies the SN-equations of definition 4. □

Using name-deletion, a transformation dual to name-addition, we represent an \( \vec{\alpha} \)-strategy \( f \) as a pair \( (f^\varnothing, \vec{\alpha}) \) by deleting from \( f \) all names that are essentially-fresh for \( i^\varnothing \) and orbiting the result under all permutations with domain \( S(\vec{\alpha}) \). We then have a \( \lambda_{op} \)-model, which is also adequate.

**Theorem 36** \( T \) is an adequate \( \lambda_{op} \)-model. □

The (omitted) proof of adequacy proceeds by showing that if \( [M] = (\vec{\beta})[S]: 0 \) then \( \vec{\alpha} \vdash M \) cannot have a reduction sequence with infinitely many DRF-reduction steps; omitting DRF’s we are left with a strongly normalising calculus.

Let us proceed to an example. Consider the typed terms \( \alpha \vdash \beta \vdash \alpha := (\text{fat } ! \alpha, \alpha) \) and \( \alpha \vdash \nu \beta, \beta := \lambda x. (\beta) \text{skip} \) with \( \alpha = a^{\mathbb{N} \times \mathbb{N}} \) and \( \beta = b^{1 \to A} \). Their translations in \( T^{\alpha} \) and \( T \) respectively are as follows.

\[
\begin{array}{|c|}
\hline
1 \quad \text{\( \alpha \)} & T_1 \\text{\( \beta \)} & T_2 \\
\hline
\end{array}
\]

The reader may want to check now that the bottom arrow, \( [[\ast \ast \circ]] \), equals \( [[\nu \beta, (\beta) := \lambda x. (\beta) \text{skip}); (\beta) \text{skip}] \).

Finally, we add observability to \( T^{\vec{\alpha}} \) as follows.

---

\[\text{We say that } \alpha \text{ is essentially fresh for } f \in T^{\vec{\alpha}}(A, B), \text{ and write } \alpha \not\in f, \text{ if } \alpha \neq \lambda \epsilon \alpha \text{ or, for any } [s] \in f \text{ and any } \beta \neq \lambda \text{list}(s), \left\{[[\alpha \beta] = \square^{\text{list}(s)}] \in f\right\}.
\]