Full abstraction for Reduced ML

Andrzej S. Murawski[a], Nikos Tzevelekos[b]

[a] *University of Leicester, UK*
[b] *Queen Mary, University of London, UK*

Abstract

We present the first effectively presentable fully abstract model for Stark's Reduced ML, a call-by-value higher-order programming language featuring integer-valued references. The model is constructed using techniques of nominal game semantics. Its distinctive feature is the presence of carefully restricted information about the store in plays, combined with conditions concerning the participants' ability to distinguish reference names. We show how it leads to an explicit characterization of program equivalence.

Keywords: Reduced ML, contextual equivalence, full abstraction, game semantics, ML-like references, nominal techniques

1. Introduction

Reduced ML is a fragment of Standard ML introduced by Stark as a vehicle for investigating dynamic allocation of mutable storage cells. Technically, it is just the call-by-value λ-calculus with primitives for manipulating integers and integer-valued references. In this paper, which is an extended version of [21], we provide a fully abstract model for the language. This means that the interpretations of two terms of the language will coincide if and only if the terms are equivalent.

Despite its simplicity, Reduced ML gives rise to a rather subtle theory of program equivalence. It can express elements of secrecy, freshness, locality and object identity. Consider, for a start, the two equivalences below, where we use the symbol $\cong$ to denote program equivalence and $=_{\mathsf{int\,ref}}$ for the name-equality test (the latter returns 1 or 0 depending on whether or not the names under comparison are equal).

$$\vdash \mathsf{ref}\,(0) =_{\mathsf{int\,ref}} \mathsf{ref}\,(0) \;\cong\; 0 \tag{1}$$

$$\vdash \mathsf{let}\, x = \mathsf{ref}(0)\,\mathsf{in}\; \lambda y^{\mathsf{int\,ref}}.(x =_{\mathsf{int\,ref}} y) \;\cong\; \lambda y^{\mathsf{int\,ref}}.0 : \mathsf{int\,ref} \to \mathsf{bool} \tag{2}$$

Equivalence (1) shows that two consecutive calls to the reference constructor ref never return identical reference names (both referenced cells will be initialised to 0, though). In example (2) we see that the name returned by ref will remain private forever. In particular, the value of x cannot leak out or be regenerated through subsequent calls to ref.

We stress that Reduced ML features integer-valued references only. Hence, even when a name is communicated to a Reduced ML context, this does not mean that the name was inevitably recorded by the environment. Indeed, the following modification of the second equivalence still holds in Reduced ML.

$$\begin{array}{rcl} f : \mathsf{int\,ref} \to \mathsf{unit} & \vdash & \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, (f\, x);\ (\lambda y^{\mathsf{int\,ref}}.x =_{\mathsf{int\,ref}} y) \\ & \cong & (f\, (\mathsf{ref}\, 0));\ (\lambda y^{\mathsf{int\,ref}}.0) : \mathsf{int\,ref} \to \mathsf{bool} \end{array} \tag{3}$$

In contrast, in a setting in which references to reference names are allowed, the above equivalence could be broken by the context given below.

$$\mathsf{let}\, r = \mathsf{ref}\ (\mathsf{ref}\ 0)\, \mathsf{in}\, \mathsf{let}\, f = (\lambda y^{\mathsf{int\,ref}}.\, r := y)\, \mathsf{in}\, ([\,-\,]\, !r)$$

Similarly, the two terms given next will be equivalent: even though x is passed to f, the environment will be not be able to memorize it beyond the runtime of f.

$$\begin{array}{rcl} f : \mathsf{int\,ref} \to \mathsf{unit} & \vdash & \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, fx; x := 0; (\lambda y^{\mathsf{unit}}.\mathsf{if}\ !x\ \mathsf{then}\ 0\ \mathsf{else}\ 1) \\ & \cong & \mathsf{let}\, x = \mathsf{ref}(0)\, \mathsf{in}\, fx; (\lambda x^{\mathsf{unit}}.1) : \mathsf{unit} \to \mathsf{int} \end{array} \tag{4}$$

The restricted capability to remember reference names means that in some scenarios the precise identity of names will be immaterial, because the context is not powerful enough to tell them apart. This underpins the following equivalence.

$$\begin{array}{rcl} f : \mathsf{int\,ref} \to \mathsf{unit} & \vdash & \mathsf{let}\, x_1 = \mathsf{ref}\ (0)\, \mathsf{in} \\ & & \mathsf{let}\, x_2 = \mathsf{ref}\ (0)\, \mathsf{in}\, ((f x_1); (x_2 := !x_1); x_2) \\ & \cong & \mathsf{let}\ x = \mathsf{ref}\ (0)\, \mathsf{in}\, (fx); x : \mathsf{int\,ref} \end{array} \tag{5}$$

Although Reduced ML programs cannot keep track of all the names they have encountered during the course of interaction with another program, at any given execution point there is a subset of such names that a program has at its disposal. In game semantics, using the notion of P-view, we can overapproximate this set by one consisting of names that occur in the current P-view as well as those that the program created itself. We call such names P-available. Intuitively, whenever a program returns a name, it will be P-available. A corresponding condition inside our model will be called *P-availability*.

As a consequence, since a program cannot have access to reference names that are not P-available, its immediate behaviour will be independent of the associated values kept in the store, because the program is simply unable to read them. This leads us to found our game model on justified sequences with partial information about the store, restricted to P-available names. Note that this form of representation also conveys the idea that the program might depend on former (possibly outdated) values of currently unavailable references, recorded when the references were still available.

Unfortunately, P-availability and partiality of store alone do not yet suffice to establish a definability[1] result. As example (5) demonstrates, a Reduced ML

[1] A definability result typically states that certain elements of the model are denotations of terms, e.g., see Lemma 39.

context may be unable to distinguish some occurrences of names introduced by the environment. In game semantics, we can capture this oversight in concrete terms: two (occurrences of) such names are indistinguishable to the program if and only if they have never occurred within the same P-view. Consequently, regardless of whether such occurrences are the same or not, the program's behaviour should remain the same. We formalize this observation via a saturation condition, called *blindness*, and show that any finitary strategy subject to all the conditions discussed above is definable, i.e. is a denotation of a Reduced ML term. This naturally leads to a fully abstract model via the usual *intrinsic quotient* construction.

To obtain a more direct account of program equivalence we next examine the structure of the quotient in more detail. Crucially, we observe that blind strategies are determined uniquely by plays in which the environment provides a fresh name each time the name cannot be related by the program to any existing names. We call such plays *strict*. Then, by symmetrizing the model, we eventually arrive at the notion of *mutually strict complete protoplays*, in which:

- all questions are answered,
- both players only use names available to them,
- stores are restricted to mutually available names,
- whenever a name appears in a player's view for the first time, it must be fresh.

This leads to an explicit characterization of equivalence: terms of Reduced ML are equivalent if and only if they induce the same mutually strict complete protoplays. This also means that the finitary (compact) part of our model can be presented effectively.

Related Work. The first steps in the semantic analysis of Reduced ML were taken by Stark himself, who identified a matching categorical framework and examined some of its instances without a full abstraction result. Further progress became possible with the arrival of game semantics [3, 9, 26]. In particular, Abramsky and McCusker presented a fully abstract model for a closely related language called RML [4], which is essentially Reduced ML extended with the "bad-variable" constructor mkvar. Its presence is a consequence of adopting Reynolds's principle of modelling references as objects with read and write methods [31]. Thus, mkvar allows one to define terms of reference type that need not correspond to actual memory locations. Unfortunately, this affects the induced notion of program equivalence, so the full abstraction result of [4] does not fully apply to Reduced ML. More precisely, their model may not validate equivalences at types containing (negative) occurrences of int ref. Typical examples are equivalences between $x := !x$ and () (the terminating command), or between $x := 1; x := 1$ and $x := 1$. In the former case the terms are inequivalent in RML, because x may be instantiated with a mkvar-object whose reading or

writing method diverges, or causes side effects. Similarly, in the latter case, an assignment to a mkvar-object might trigger a side effect that effectively allows one to count how many assignments took place.

The "bad-variable" phenomenon, also present in the call-by-name setting, has inspired subsequent developments in game semantics. It turned out that, in the call-by-name framework, it could be circumvented by employing suitably crafted (pre)orders on plays [16, 20], but no result of this kind has been reported for call-by-value. However, an alternative and general approach to dealing with bad variables seems to have emerged in the form of *nominal game semantics* [10, 1, 34]. Nominal game semantics advocates a departure from Reynolds's modelling rule and stipulates that reference types be modelled by names rather than objects. Using this approach, Laird showed a full abstraction result for a call-by-value language $\lambda\nu!$ with storable names rather than integers [10]. As mentioned above, $\lambda\nu!$ is more expressive than Reduced ML in its ability to distinguish reference names and, consequently, the obvious adaptation of the model to Reduced ML results in a failure of full abstraction.

Modelling references as names leads to considering a particular style of games where the side-effects become, to some extent, explicit. In particular, moves in our model are attached with *stores*, comprising of pairs of names and values, which comes in stark contrast to the traditional modelling of side effects in game semantics [5, 2]. Note that this approach is not specific to nominal games and has been previously followed also in non-nominal settings [27, 13, 22].

Research into ML-like languages has also produced fully abstract game models for more significant extensions of Reduced ML, notably languages with higher-order references [2, 34, 24]. These do not extend Reduced ML conservatively. For example, the following two terms are equivalent in Reduced ML

$$\begin{array}{rcl} f : \mathsf{unit} \to \mathsf{unit} & \vdash & \mathsf{let}\, x = \mathsf{ref}\,(0)\,\mathsf{in}\,\lambda y^{\mathsf{unit}}.\mathsf{if}\,!x\,\mathsf{then}\,()\,\mathsf{else}\,(x := 1; f()) \\ & \cong & \mathsf{let}\, x = \mathsf{ref}\,(0)\,\mathsf{in}\,\lambda y^{\mathsf{unit}}.\mathsf{if}\,!x\,\mathsf{then}\,()\,\mathsf{else}\,(f(); x := 1) : \mathsf{unit} \end{array}$$

but there exists a context with general references, which can separate them:

$$\mathsf{let}\, r = \mathsf{ref}\,(\lambda x^{\mathsf{unit}}.x)\,\mathsf{in}\,\mathsf{let}\, f = \lambda y^{\mathsf{unit}}.(!r)()\,\mathsf{in}\,(r := [-]; (!r)()).$$

It seems possible to adapt the approach of [34] to Reduced ML. However, it leans on quotienting in order to achieve full abstraction (information on local state and store update is too explicit in the intensional model and leads to substantial undesirable distinctions). Therefore, it does not lead to an explicit characterization of program equivalence, which we obtain in the present paper.

2. Reduced ML

Reduced ML [33] is an idealised programming language combining higher-order types and integer references in the style of ML. Formally, it is the call-by-value λ-calculus over the ground types unit, int, int ref augmented with basic commands (termination, divergence), primitives for integer arithmetic (constants,

$$\frac{}{\mathsf{u},\Gamma \vdash () : \mathsf{unit}} \qquad \frac{}{\mathsf{u},\Gamma \vdash \Omega : \mathsf{unit}} \qquad \frac{i \in \mathbb{Z}}{\mathsf{u},\Gamma \vdash i : \mathsf{int}} \qquad \frac{l \in \mathsf{u}}{\mathsf{u},\Gamma \vdash l : \mathsf{int\ ref}}$$

$$\frac{(x:\theta) \in \Gamma}{\mathsf{u},\Gamma \vdash x : \theta} \qquad \frac{\mathsf{u},\Gamma \vdash M_1 : \mathsf{int} \qquad \mathsf{u},\Gamma \vdash M_2 : \mathsf{int}}{\mathsf{u},\Gamma \vdash M_1 \oplus M_2 : \mathsf{int}}$$

$$\frac{\mathsf{u},\Gamma \vdash M : \mathsf{int}}{\mathsf{u},\Gamma \vdash \mathsf{ref}\, M : \mathsf{int\ ref}} \qquad \frac{\mathsf{u},\Gamma \vdash M_1 : \mathsf{int\ ref} \qquad \mathsf{u},\Gamma \vdash M_2 : \mathsf{int\ ref}}{\mathsf{u},\Gamma \vdash M_1 = M_2 : \mathsf{int}}$$

$$\frac{\mathsf{u},\Gamma \vdash M : \mathsf{int} \quad \mathsf{u},\Gamma \vdash N_1 : \theta \quad \mathsf{u},\Gamma \vdash N_2 : \theta}{\mathsf{u},\Gamma \vdash \mathsf{if}\, M\, \mathsf{then}\, N_1\, \mathsf{else}\, N_2 : \theta}$$

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

$$\frac{\mathsf{u},\Gamma \vdash M : \theta \to \theta' \quad \mathsf{u},\Gamma \vdash N : \theta}{\mathsf{u},\Gamma \vdash MN : \theta'} \qquad \frac{\mathsf{u},\Gamma \uplus \{x:\theta\} \vdash M : \theta'}{\mathsf{u},\Gamma \vdash \lambda x^{\theta}.M : \theta \to \theta'}$$

Figure 1: Syntax of Reduced ML. ($\uplus$ stands for the disjoint union.)

---

zero-test, binary integer functions) and reference manipulation (locations, dereferencing, assignment, equality testing, memory allocation). The typing rules are given in Figure 1, where $\mathcal{L}$ stands for a countable set of *locations*, u for a finite subset of $\mathcal{L}$, and $\oplus$ for binary integer functions (e.g. $+$, $-$, $*$, $=$). Their precise choice is to a large extent immaterial: only the ability to compare with integer constants is crucial.

Remark 1. Note that reference equality testing can actually be programmed [29] by writing two different values to the corresponding locations and checking how they were affected. For example, one can define $\mathsf{eq} : \mathsf{int\ ref} \to \mathsf{int\ ref} \to \mathsf{int}$ as

$$\begin{array}{ll} \lambda x^{\mathsf{int\ ref}}.\lambda y^{\mathsf{int\ ref}}. & \mathsf{let}\, v_x = \mathsf{ref}\, !x\, \mathsf{in} \\ & \mathsf{let}\, v_y = \mathsf{ref}\, !y\, \mathsf{in} \\ & \quad x := 0;\ y := 1; \\ & \quad \mathsf{let}\, b = (!x = !y)\, \mathsf{in}\, (x := !v_x;\ y := !v_y;\ b) \end{array}$$

In the above and in what follows, we write $\mathsf{let}\, x = M\, \mathsf{in}\, N$ for the term $(\lambda x^{\theta}.N)M$. Where x does not occur in N, we write $M; N$.

To define the operational semantics of Reduced ML, we introduce a notion of *store*. It will simply be a function from a finite set of locations to $\mathbb{Z}$. We write $s(l \mapsto i)$ for the store obtained by updating s so that l is mapped to i (this may extend the domain of s). Given a store $s : \{l_1, \cdots, l_n\} \to \mathbb{Z}$ and a term M we say that the pair (s, M) is *compatible* if and only if all locations occurring in M are from $\{l_1, \cdots, l_n\}$. We say that a term is *canonical* if it is either $()$, an integer constant, a location, a variable or a λ-abstraction. The big-step reduction rules are given as judgements of the shape $s, M \Downarrow s', V$, where (s, M),

$$\frac{V \text{ is canonical}}{s, V \Downarrow s, V} \qquad \frac{M \Downarrow 0 \quad N_2 \Downarrow V}{\mathsf{if}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_2 \Downarrow V} \qquad \frac{i \neq 0 \quad M \Downarrow i \quad N_1 \Downarrow V}{\mathsf{if}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_2 \Downarrow V}$$

$$\frac{M_1 \Downarrow i_1 \quad M_2 \Downarrow i_2}{M_1 \oplus M_2 \Downarrow i_1 \oplus i_2} \qquad \frac{M \Downarrow \lambda x.M' \quad N \Downarrow V' \quad M'[V'/x] \Downarrow V}{MN \Downarrow V}$$

$$\frac{s, M \Downarrow s', i \quad l \notin \mathsf{dom}(s')}{s, \mathsf{ref}\, M \Downarrow s'(l \mapsto i), l} \qquad \frac{s, M \Downarrow s', l}{s, !M \Downarrow s', s'(l)} \qquad \frac{s, M \Downarrow s', l \quad s', N \Downarrow s'', i}{s, M := N \Downarrow s''(l \mapsto i), ()}$$

$$\frac{M_1 \Downarrow l_1 \quad M_2 \Downarrow l_2 \quad l_1 = l_2}{M_1 = M_2 \Downarrow 1} \qquad \frac{M_1 \Downarrow l_1 \quad M_2 \Downarrow l_2 \quad l_1 \neq l_2}{M_1 = M_2 \Downarrow 0}$$

Figure 2: Big-step operational semantics of Reduced ML

(s', V) are compatible, $\mathsf{dom}(s) \subseteq \mathsf{dom}(s')$ and V is canonical. We present them in Figure 2, where we let l, l_1, l_2 range over locations. In line with the definition of Standard ML [17], most rules take the form

$$\frac{M_1 \Downarrow V_1 \quad M_2 \Downarrow V_2 \quad \cdots \quad M_n \Downarrow V_n}{M \Downarrow V}$$

which is meant to abbreviate

$$\frac{s_1, M_1 \Downarrow s_2, V_1 \quad s_2, M_2 \Downarrow s_3, V_2 \quad \cdots \quad s_n, M_n \Downarrow s_{n+1}, V_n}{s_1, M_1 \Downarrow s_{n+1}, V}.$$

In particular, this means that the ordering of the hypotheses is significant. Although the choice of different l's in the rule for ref yields a degree of non-determinism, reductions are unique up to choice of fresh locations (Theorem 5.4 in [33]).

We shall write $\Gamma \vdash M : \theta$ if and only if $\emptyset, \Gamma \vdash M : \theta$ can be derived using the rules of Figure 1. Similarly, $\vdash M : \theta$ is shorthand for $\emptyset, \emptyset \vdash M : \theta$. Given $\vdash M : \mathsf{unit}$ we write $M \Downarrow$ if and only if $\emptyset, M \Downarrow s, ()$ for some store s.

Definition 2. We say that the term-in-context $\Gamma \vdash M_1 : \theta$ ***approximates*** $\Gamma \vdash M_2 : \theta$ (written $\Gamma \vdash M_1 \mathrel{\underline{\lesssim}} M_2$) iff $C[M_1] \Downarrow$ implies $C[M_2] \Downarrow$ for any context $C[-]$ such that $\vdash C[M_1], C[M_2] : \mathsf{unit}$. Two terms-in-context are ***equivalent*** if one approximates the other (written $\Gamma \vdash M_1 \cong M_2$).

Remark 3. The only difference between our definition of Reduced ML and Stark's is the presence of Ω, the divergent constant without a reduction rule. Although we introduce it at type unit only, divergence at other types can be achieved by considering $\Omega; M$, where M is a closed term of the desired type (e.g. 0, $\mathsf{ref}\, 0$, $\lambda x^{\mathsf{unit}}.0$).

Thanks to non-termination, we can define $\mathrel{\underline{\lesssim}}$ and decompose reasoning about $\cong$. Also, the inclusion of Ω induces domain-theoretic structure and makes it possible

to study compact elements and their effective presentability. At the same time, the presence of Ω gives rise to the same notion of equivalence between Ω-free Reduced ML terms as in [33], because $C[M] \Downarrow$, where M is Ω-free, is equivalent to $\emptyset, C'[M] \Downarrow s', 0$ where

$$C'[-] \equiv \mathsf{let}\, x = \mathsf{ref}\, 0\, \mathsf{in}\, (C[x := 1/\Omega])[M]; !x$$

and s' is a state.

3. Game models

In this section we proceed to the construction of our game model. Let us fix a countably infinite set $\mathbb{A}$, the set of *names*, the elements of which we denote by a, b, c and variants. From now on, we shall work under the assumption:

$$\mathcal{L} = \mathbb{A}$$

In nominal game semantics two participants play a game by exchanging moves that might involve names. However, when employing such moves, we are not interested in what exactly the names are, though we would like to know how they relate to names that have already been in play. Hence, the objects of study are rather the induced equivalence classes with respect to name-invariance. Since we want all game-semantic notions and constructions to be compatible with name-invariance, their obvious adaptations would repeatedly have to include conditions that enforce closure under renamings. Fortunately, this overhead can be dealt with robustly using the language of nominal set theory [6].

3.1. Nominal sets

Definition 4. Let us write $\textsc{Perm}(\mathbb{A})$ for the group of finite permutations of $\mathbb{A}$. A ***nominal set*** X is a set $|X|$ (usually written X) equipped with a group action of $\textsc{Perm}(\mathbb{A})$.[2] Moreover, each $x \in X$ must have *finite support*, that is, there exists a finite set $\mathsf{u} \subseteq \mathbb{A}$ such that, for all permutations π,

$$(\forall a \in \mathsf{u}.\, \pi(a) = a) \implies \pi \cdot x = x\,.$$

Finite support is closed under intersection, and hence each element x of a nominal set has a least support $\nu(x)$, which we call ***the support of*** x. Intuitively, $\nu(x)$ is the set of names "involved" in x. Accordingly, we say that a *is fresh for* x if $a \notin \nu(x)$.

Trivially, every set X can be seen as a nominal set by setting $\pi \cdot x = x$ for all $x \in X$. On the other hand, $\mathbb{A}$ is a nominal set by taking $\pi \cdot a = \pi(a)$, for each π and a. The set $\mathcal{P}_{\mathrm{fin}}(\mathbb{A})$ of finite sets of names is also a nominal set, with action $\pi \cdot \mathsf{u} = \{\pi(a) \mid a \in \mathsf{u}\}$ for each permutation π and finite $\mathsf{u} \subseteq \mathbb{A}$. Moreover, $\nu(\mathsf{u}) = \mathsf{u}$. More interestingly, if X and Y are nominal sets then so are:

[2] A group action of $\textsc{Perm}(\mathbb{A})$ on X is a function $\_ \cdot \_ : \textsc{Perm}(\mathbb{A}) \times X \to X$ such that, for all $x \in X$ and $\pi, \pi' \in \textsc{Perm}(\mathbb{A})$, $\pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$ and $\mathsf{id} \cdot x = x$, where id is the identity permutation.

- their cartesian product $X \times Y$, with permutations acting componentwise, $\pi \cdot (x, y) = (\pi \cdot x, \pi \cdot y)$;
- their disjoint union $X \uplus Y$, with permutations acting as in X or Y;
- the set X^* of finite sequences of elements of X, with permutations acting elementwise, $\pi \cdot (x_1 \dots x_n) = (\pi \cdot x_1) \cdots (\pi \cdot x_n)$.

For example, the set $\mathbb{A}^*$ of finite sequences of atoms is a nominal set with

$$\pi \cdot (a_1 \dots a_n) = (\pi \cdot a_1) \dots (\pi \cdot a_n) = \pi(a_1) \dots \pi(a_n)$$

for each permutation π and $a_1, \cdots, a_n \in \mathbb{A}$. Moreover, $\nu(a_1 \dots a_n)$ contains precisely the names $a_1, \cdots, a_n$ (modulo repetitions).

We call $X' \subseteq X$ a *nominal subset* of X if X' is closed under permutations, these acting as on X. Accordingly, we say that $R \subseteq X \times Y$ is a *nominal relation* if R a nominal subset of $X \times Y$. Concretely, this means that $(x, y) \in R$ implies $(\pi \cdot x, \pi \cdot y) \in R$, for all permutations π. A *nominal function* is a function which is also a nominal relation. If $f : X \to Y$ is a nominal function then, for each $x \in X$ and permutation π, $f(\pi \cdot x) = \pi \cdot f(x)$. As a consequence, $\nu(f(x)) \subseteq \nu(x)$.

Finally, for each element x of a nominal set X, we can form its orbit

$$[x] = \{\pi \cdot x \mid \pi \in \textsc{Perm}(\mathbb{A})\}$$

under the permutation action. Taking the orbit $[x]$ of x can be seen as blurring the specific choice of names within x while retaining its underlying structure. Note that $[x] \subseteq X$. For example, for any sequence $a_1 \dots a_n \in \mathbb{A}^*$, its orbit

$$[a_1 \dots a_n] = \{\pi \cdot (a_1 \dots a_n) \mid \pi \in \textsc{Perm}(\mathbb{A})\}$$

contains all sequences of atoms $a'_1 \dots a'_n$ such that, for all $i, j \in \{1, ..., n\}$, we have $a'_i = a'_j$ if, and only if, $a_i = a_j$. On the other hand, in the nominal set $\mathbb{A} \times \mathbb{A}$ we can form just two orbits, namely $[(a, b)]$ and $[(a, a)]$, for some $a \neq b \in \mathbb{A}$. The former contains all pairs of distinct names, while the latter all pairs made of the same name.

In game semantics a particular strengthening of the notion of support, called *strong support*, has turned out to be necessary to guarantee correct behaviour under strategy composition.[3] In particular, we say that X is a ***strong nominal set*** if, for all $x \in X$ and permutations π,

$$\pi \cdot x = x \implies \forall a \in \nu(x).\, \pi(a) = a\,.$$

For example, the nominal set $\mathbb{A}^*$ is strong, whereas $\mathcal{P}_{\text{fin}}(\mathbb{A})$ is not. Note that a nominal subset of a strong nominal set is itself strong and, moreover, if X and Y are strong nominal sets then so are $X \times Y$, $X \uplus Y$ and X^*.

[3] See [35] for motivation and a detailed explanation of its significance. The same notion was pinpointed by Schöpp [32], as describing nominal sets with an *essentially simple* action.

3.2. Nominal arenas

Our semantics involves games comprising a formal exchange of *moves* between two players: a Proponent, corresponding to the modelled term; and an Opponent, corresponding to the computational environment of the term. The moves are selected from *(pre)arenas*, whose construction follows the structure of types. Arenas specify the set of available moves and the correlation between different moves within a game. Next we present these notions in detail. They are essentially the call-by-value arenas of Honda and Yoshida [8], cast inside the theory of (strong) nominal sets.

Definition 5. An ***arena*** $A = (M_A, I_A, \vdash_A, \lambda_A)$ is given by:

- a strong nominal set M_A of moves,
- a nominal subset $I_A \subseteq M_A$ of initial moves,
- a nominal enabling relation $\vdash_A \subseteq M_A \times (M_A \setminus I_A)$,
- a nominal labelling function $\lambda_A : M_A \to \{O, P\} \times \{Q, A\}$,

satisfying, for each $m, m' \in M_A$, the conditions:

- $m \in I_A \implies \lambda_A(m) = (P, A)$,
- $m \vdash_A m' \wedge \lambda_A^{QA}(m) = A \implies \lambda_A^{QA}(m') = Q$,
- $m \vdash_A m' \implies \lambda_A^{OP}(m) \neq \lambda_A^{OP}(m')$.

The role of λ_A is to label moves as *Opponent* or *Proponent* moves and as *Questions* or *Answers*. We write λ_A^{OP} and λ_A^{QA} for λ_A followed by a first and second projection functions respectively. The simplest arena is $0 = (\emptyset, \emptyset, \emptyset, \emptyset)$. Other flat arenas are 1, $\mathbb{Z}$ and $\mathbb{A}$, defined by

$$M_1 = I_1 = \{*\}, \quad M_{\mathbb{Z}} = I_{\mathbb{Z}} = \mathbb{Z}, \quad M_{\mathbb{A}} = I_{\mathbb{A}} = \mathbb{A},$$

where by "flat" we mean that, in each case, their enabling relation is empty.[4] Arenas can be seen as directed bipartite graphs, with nodes given by moves, edges by the enabling relation, and in which the partition is defined by the OP-polarities. These graphs are rooted at initial moves.

We shall make use of the following constructions on arenas. By $\bar{I}_A$ we denote $M_A \setminus I_A$, by $\bar{\lambda}_A$ the OP-complement of λ_A, while ι_A and ι_B range over initial

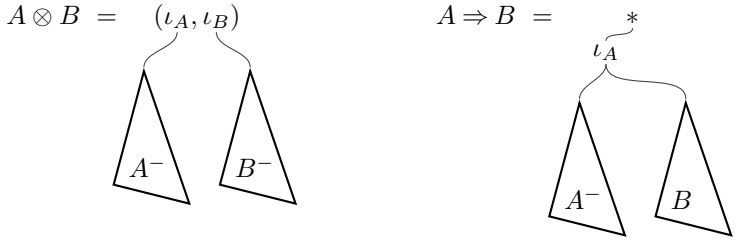
[4]Note that $M_A = I_A$ implies that $\vdash_A$ is empty: by $M_A = I_A$ all moves are P-moves and enabling can only relate moves of different polarities.

moves in the respective arenas.

$$\begin{array}{rl} M_{A\otimes B} & = (I_A \times I_B) \uplus \overline{I}_A \uplus \overline{I}_B \\ I_{A\otimes B} & = I_A \times I_B \\ \lambda_{A\otimes B} & = [((\iota_A, \iota_B), PA), \lambda_A \restriction \overline{I}_A, \lambda_B \restriction \overline{I}_B] \\ \vdash_{A\otimes B} & = \{((\iota_A, \iota_B), m) \mid \iota_A \vdash_A m \text{ or } \iota_B \vdash_B m\} \cup (\vdash_A \restriction \overline{I}_A{}^2) \cup (\vdash_B \restriction \overline{I}_B{}^2) \end{array}$$

$$\begin{array}{rl} M_{A\Rightarrow B} & = \{*\} \uplus M_A \uplus M_B \\ I_{A\Rightarrow B} & = \{*\} \\ \lambda_{A\Rightarrow B} & = [(*, PA), (\iota_A, OQ), \bar{\lambda}_A \restriction \overline{I}_A, \lambda_B] \\ \vdash_{A\Rightarrow B} & = \{(*, \iota_A), (\iota_A, \iota_B)\} \cup \vdash_A \cup \vdash_B \end{array}$$

The above constructions can be more concisely described by the following graphs,



where we write A^- for A with its initial moves removed.

The types of Reduced ML are interpreted to arenas in the following way.

$$[\![\mathsf{unit}]\!] = 1\,, \quad [\![\mathsf{int}]\!] = \mathbb{Z}\,, \quad [\![\mathsf{int\,ref}]\!] = \mathbb{A}\,, \quad [\![\theta_1 \to \theta_2]\!] = [\![\theta_1]\!] \Rightarrow [\![\theta_2]\!]\,.$$

Moreover, each finite $\mathsf{u} \subseteq \mathbb{A}$, say $\mathsf{u} = \{a_1, \cdots, a_n\}$, is mapped to:

$$[\![\mathsf{u}]\!] = (M_\mathsf{u}, I_\mathsf{u}, \lambda_\mathsf{u}, \vdash_\mathsf{u})\,, \quad M_\mathsf{u} = I_\mathsf{u} = [(a_1, \cdots, a_n)]\,.$$

Remark 6. In the above translation of types we can vividly see the difference between the *nominal approach* to modelling references, largely formulated in Pitts and Stark's work on the ν-calculus [28], and the *object-like* approach of Reynolds [31]. In the latter, references would be modelled as *products* of their read and write methods:

$$[\![\mathsf{ref}\,\theta]\!] \;\cong\; (1 \Rightarrow [\![\theta]\!]) \times ([\![\theta]\!] \Rightarrow 1).$$

Here, on the other hand, all types A_θ are given distinct atomic representations. A positive aspect of the Reynolds's interpretation is functoriality of the ref-constructor. However, although semantically pleasing, this is not necessarily meaningful from a programmer's point of view: given closed terms $M_1 : \mathsf{ref}\,\theta_1$ and $M : \theta_1 \to \theta_2$ (and even $M' : \theta_2 \to \theta_1$), one can canonically construct a term $M_2 : \mathsf{ref}\,\theta_2$, but in doing so a fresh reference for M_2 must be created (this procedure then is not functorial as identities are not preserved). In general, in programs one cannot convert references from one type to another, unless by use of *type casting*. But the latter is quite different from requiring ref to be functorial.

Although types are interpreted by arenas, the actual games are played *between arenas*, in structures called ***prearenas***, which are defined in the same way as arenas with the exception that initial moves are O-questions. For given arenas A, B we can construct a prearena $A \to B$ by

$$\begin{array}{ll} M_{A\to B} = M_A \uplus M_B & \lambda_{A\to B} = [(\iota_A, OQ) \cup (\bar{\lambda}_A \restriction \overline{I}_A)\ ,\ \lambda_B] \\ I_{A\to B} = I_A & \vdash_{A\to B} = \{(\iota_A, \iota_B)\} \cup \vdash_A \cup \vdash_B\ . \end{array}$$

Typing judgements $\mathsf{u}, \Gamma \vdash \theta$, where $\Gamma = \{x_1 : \theta_1, \cdots, x_n : \theta_n\}$, will be interpreted by strategies for the prearena

$$[\![\mathsf{u}]\!] \otimes [\![\theta_1]\!] \otimes \cdots \otimes [\![\theta_n]\!] \to [\![\theta]\!]$$

(if $n + |\mathsf{u}| = 0$ we take the left-hand side to be 1) which we shall denote by $[\![\mathsf{u}, \Gamma \vdash \theta]\!]$, or just $[\![\Gamma \vdash \theta]\!]$ if $\mathsf{u} = \emptyset$.

3.3. Plays

Analogously to the definition of store in Section 2, in this section a store will be a partial function $\Sigma : \mathbb{A} \rightharpoonup \mathbb{Z}$ such that $\mathsf{dom}(\Sigma)$ is finite.

Remark 7. Our game-semantic framework relies on moves equipped with stores. This should be contrasted with models for languages with bad variables [4, 2], which did not feature names and explicit store-related information. The incorporation of stores into plays is proving to be an effective technique for modelling scenarios without bad variables [1, 10, 12, 34, 35].

A ***basic justified sequence*** in a prearena A is a finite sequence s of moves of A satisfying the conditions:

- the first move of s must be initial,
- but all other moves m must be equipped with a pointer to an earlier occurrence of some move m' such that $m' \vdash_A m$ (we then say that m' justifies m; if m is an answer, we might also say that m *answers* m').

A ***justified sequence*** in A is a basic justified sequence s in which each move is, in addition, decorated with a store to yield a ***move-with-store***, typically denoted by m^Σ. Given a justified sequence s, we write $\underline{s}$ for the underlying basic justified sequence. It should be clear that, similarly to the set of finite sequences of moves, the set of justified sequences can be viewed as a (not necessarily strong) nominal set with permutations preserving the pointer structure, but acting on moves as in A and on stores by permuting the domain.

Below we define the notions of O-view $\llcorner s \lrcorner$ and P-view $\ulcorner s \urcorner$ of a justified sequence, using o and p to range over O-moves and P-moves respectively. We write $s' \sqsubseteq s$ if s' is a prefix of s and use $\sqsubseteq^{\text{even}}$ if s' is of even length.

$$\begin{array}{rcl@{\qquad}rcl} \llcorner \epsilon \lrcorner & = & \epsilon & \ulcorner \iota \urcorner & = & \iota \\ \llcorner s\, o^\Sigma \lrcorner & = & \llcorner s \lrcorner\, o^\Sigma & \ulcorner s\, p^\Sigma \urcorner & = & \ulcorner s \urcorner\, p^\Sigma \\ \llcorner s\, \overbrace{o^\Sigma t}\, p^{\Sigma'} \lrcorner & = & \llcorner s \lrcorner\, o^\Sigma p^{\Sigma'} & \ulcorner s\, \overbrace{p^\Sigma t}\, o^{\Sigma'} \urcorner & = & \ulcorner s \urcorner\, p^\Sigma o^{\Sigma'} \end{array}$$

A name in s is said to be ***introduced by player*** X ($X \in \{O, P\}$) if its first occurrence in $\underline{s}$ is in (the support of) an X-move. Names introduced by X in s will be referred to as ***X-names*** in s and denoted with $\mathsf{X}(s)$. We define the set $\mathsf{Av}_X(s)$ of ***X-available names after*** s by:

$$\mathsf{Av}_{\mathrm{O}}(s) = \mathsf{O}(s) \cup \nu(\llcorner \underline{s} \lrcorner) \qquad \mathsf{Av}_{\mathrm{P}}(s) = \mathsf{P}(s) \cup \nu(\ulcorner \underline{s} \urcorner)$$

Definition 8. A justified sequence s is ***legal*** if it satisfies the following conditions.

- No two adjacent moves belong to the same player (*Alternation*).
- The justifier of each answer is the most recent unanswered question (*Well-Bracketing*).
- For all $tm^{\Sigma} \sqsubseteq s$, the justifier of m is in $\ulcorner tm^{\Sigma} \urcorner$ (*Visibility*).
- For all $t \sqsubseteq s$, $\nu(t) = \nu(\underline{t})$ (*Frugality*).

The set of legal justified sequences will be denoted by L_A.

The first three conditions are standard in games for stateful computation without higher-order store (which invalidates visibility) or control operators (which break well-bracketing). Frugality stipulates that names cannot be generated in the stores 'from thin air' but, rather, they must occur earlier in the support of a move. Consequently, the support of a legal sequence is that of its underlying basic sequence, and therefore L_A is a strong nominal set. Note that frugality can be equivalently expressed as: for all $tm^{\Sigma} \sqsubseteq s$, $\mathsf{dom}(\Sigma) \subseteq \nu(\underline{t}m)$. Our model is based on plays where stores are even more restricted: only names *available* to P are allowed in stores.

Definition 9. A legal sequence s is a ***play*** if it satisfies the following additional conditions.

- For all $s'p^{\Sigma} \sqsubseteq^{\mathrm{even}} s$ and $a \in \mathbb{A}$, if $a \in \nu(p) \cap \nu(s')$ then $a \in \mathsf{Av}_{\mathrm{P}}(s')$ (*P-Availability*).
- For all $s'm^{\Sigma} \sqsubseteq s$, $\mathsf{dom}(\Sigma) = \mathsf{Av}_{\mathrm{P}}(s'm^{\Sigma})$ (*P-Storage*).

The set of plays over prearena A will be denoted by P_A.

Note that the two conditions are biased towards P. Moreover, P-availability can be equivalently restated as: for all $s'p^{\Sigma} \sqsubseteq^{\mathrm{even}} s$ and $a \in \mathbb{A}$, if $a \in \mathsf{O}(s')$ then $a \in \nu(\underline{\ulcorner s' \urcorner})$. It is worth observing that, given $s \in P_A$, we have $\mathsf{Av}_{\mathrm{P}}(s) = \nu(\ulcorner s \urcorner)$.

Remark 10. Note that our definition of a play is founded on that of a basic justified sequence, which underpins game-semantic frameworks in the spirit of Hyland and Ong [9, 14]. Similarly, all of the conditions not referring to stores (alternation, visibility, well-bracketing) are standard. Consequently, we will be able to reuse numerous old results, which do not concern stores, such as preservation of visibility under composition [14].

3.4. Strategies

Definition 11. A ***strategy*** σ on a prearena A is a set of even-length plays of A satisfying:

- If $so^{\Sigma}p^{\Sigma'} \in \sigma$ then $s \in \sigma$ (*Even-Prefix Closure*).
- If $s \in \sigma$ then, for all $\pi \in \textsc{Perm}(\mathbb{A})$, $\pi \cdot s \in \sigma$ (*Equivariance*).
- If $sp_1^{\Sigma_1}, sp_2^{\Sigma_2} \in \sigma$ then $sp_1^{\Sigma_1} = \pi \cdot sp_2^{\Sigma_2}$, some $\pi \in \textsc{Perm}(\mathbb{A})$ (*Determinacy*).

We write $\sigma : A$.

We next show how strategies are composed. First, let us set γ to be an endofunction on justified sequences which restricts any justified sequence to a frugal one by removing from the stores the names violating frugality. Formally,

$$\gamma(\epsilon) = \epsilon\,, \quad \gamma(tm^{\Sigma}) = \gamma(t)\, m^{\Sigma'}$$

where $\Sigma' = \Sigma \upharpoonright \nu(\underline{t}m)$. Moreover, let γ' be an analogous function enforcing P-storage, i.e. γ' removes O-names violating P-storage:

$$\gamma'(\epsilon) = \epsilon\,, \quad \gamma'(tm^{\Sigma}) = \gamma'(t)\, m^{\Sigma'}$$

where $\Sigma' = \Sigma \upharpoonright \mathsf{Av}_{\mathrm{P}}(tm^{\Sigma})$.

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

$$\begin{array}{ll} M_{A\to B\to C} = M_{A\to B} \uplus M_C & \lambda_{A\to B\to C} = [\lambda_{A\to B}[\iota_B \mapsto PQ], \overline{\lambda}_C] \\ I_{A\to B\to C} = I_A & \vdash_{A\to B\to C} = \vdash_{A\to B} \cup \{(\iota_B, \iota_C)\} \cup \vdash_C \end{array}$$

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

Definition 12. A justified sequence u on $A \to B \to C$ is an ***interaction sequence*** on A, B, C if $\gamma'(u \upharpoonright AB) \in P_{A\to B}$, $\gamma'(u \upharpoonright BC) \in P_{B\to C}$ and the following conditions are satisfied.

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

– or m' is an O-move in AC and $a \notin \mathsf{Av}_{\mathrm{P}}(u' \restriction AC)$,

then $\Sigma(a) = \Sigma'(a)$;

- for all $u' \sqsubseteq u$ ending in some m^{Σ},

$$\mathsf{dom}(\Sigma) = \nu(\ulcorner \underline{u}' \restriction AC \urcorner) \cup \mathsf{P}(u' \restriction AB) \cup \mathsf{P}(u' \restriction BC)\,.$$

We write $\mathit{Int}(A, B, C)$ for the set of interaction sequences on A, B, C.

The former three conditions above come originally from Laird's work [12], and were also applied in [24]. The first two ensure that name-privacy is not broken under composition, while the third one ensures that players do not change parts of the store inaccessible to them. The last condition is a version of P-storage appropriate for interactions: the "generalised Proponent" in an interaction is the one playing all P-moves, while the available names introduced by the "outside Opponent" are those in the AC P-view. Note that this condition entails frugality for the sequence u.

We next show that interactions lead to a well-defined notion of composition. In particular, we are interested in proving that projecting an ABC-interaction sequence on AC yields a play in $A \to C$. As observed in Remark 10, when stores are omitted from moves, our plays fits into the standard pattern studied, for example, in [14]. This will allow us to obtain the non-nominal properties of plays in the standard way. Thus, it remains to show that P-availability is preserved.

Given $u \in \mathit{Int}(A, B, C)$ and m^{Σ} in u, we say that m is a *generalised P-move* in u if it is a P-move in either of $\underline{u} \restriction AB$, $\underline{u} \restriction BC$. Accordingly, we write $\mathsf{P}(u)$ for $\mathsf{P}(u \restriction AB) \cup \mathsf{P}(u \restriction BC)$. Moreover, by $u_{\leq m}$ we mean the initial segment of u ending in m^{Σ} (Σ is dropped from the notation for economy).

Lemma 13. *For any $u \in \mathit{Int}(A, B, C)$,*

1. $\nu(u) = \mathsf{O}(u \restriction AC) \uplus \mathsf{P}(u \restriction AB) \uplus \mathsf{P}(u \restriction BC)\,,$
2. $\mathsf{O}(u \restriction AC) \cap \nu(\ulcorner \gamma(u \restriction AC) \urcorner) = \mathsf{O}(u \restriction AC) \cap \nu(\ulcorner \underline{u} \restriction AC \urcorner)\,.$

Proof. For 1, by definition of legal interactions the three components have no common elements. So take any $a \in \nu(u)$ which does not belong to $\mathsf{P}(u)$. Then, a must be introduced in u at AC in some O-move, so $a \in \mathsf{O}(u \restriction AC)$.
For 2, we need only show the left-to-right inclusion. Let $a \in$ LHS, so $a \notin \mathsf{P}(u)$ by second condition on interactions, and suppose it first appears in $\ulcorner \gamma(u \restriction A, C) \urcorner$ in some m^{Σ}, which is the result of applying γ to some $m^{\Sigma'} \in u$. If $a \in \nu(m)$, we are done. Otherwise, $a \in \mathsf{dom}(\Sigma')$. By the last condition on interaction sequences and $a \notin \mathsf{P}(u)$ we obtain $a \in \nu(\ulcorner \underline{u} \restriction A, C \urcorner)$. □

Our next argument uses the following notion from [14]. Given $u \in Int(A, B, C)$, the *core* of u, written $\overline{u}$, is defined by:

$$\begin{aligned}
\overline{\iota^{\Sigma}} &= \iota^{\Sigma} \\
\overline{um^{\Sigma}} &= \overline{u}m^{\Sigma} && \text{if } m \text{ is a generalised P-move} \\
\overline{u\overbrace{n^{\Sigma'}u'}m^{\Sigma}} &= \overline{u}n^{\Sigma'}m^{\Sigma} && \text{if } m \text{ is an O-move in } AC \text{ justified by } n
\end{aligned}$$

We can see that $\overline{u} \upharpoonright AC = \ulcorner u \upharpoonright AC \urcorner$. The following result is standard.

Lemma 14. *Let $u \in Int(A, B, C)$, $X \in \{AB, BC\}$ and $Y \in \{AC\}$.*

1. *If $u'm^{\Sigma}n^{\Sigma'} \sqsubseteq u$ then:*
   - *m is an O-move in X iff n is a P-move in X,*
   - *m is a P-move in Y iff n is an O-move in Y.*
2. *If $u = u'm^{\Sigma}$ with m an O-move in X then $\ulcorner \overline{u} \upharpoonright X \urcorner = \ulcorner u \upharpoonright X \urcorner$.*

Proof. Proof of 1 is by induction on u', using the switching conditions for plays in $A \to B$ and $B \to C$, see e.g. [8]. Part 2 is part 3 of [14, Lemma 3.2.3]. □

Lemma 15. *Suppose $u \in Int(A, B, C)$. Then, for any name $a \in \mathsf{O}(u \upharpoonright AC)$,*

1. *if a occurs in some generalised P-move p in u, then a occurs in an O-move of $\ulcorner u_{<p} \upharpoonright AC \urcorner$,*
2. *if $a \in \nu(\overline{u})$ then $a \in \nu(\ulcorner \underline{u} \upharpoonright AC \urcorner)$.*

Proof. For 1, we use induction on the length of u, base case trivial. By definition, $a \in \mathsf{O}(u \upharpoonright AC)$ implies $a \notin \mathsf{P}(u)$, so if a occurs in $u \upharpoonright AB$ or $u \upharpoonright BC$ then it does so as an O-name. Suppose $a \in \nu(p)$, with p a P-move in component X ($X \in \{AB, BC\}$). By $a \in \mathsf{O}(u \upharpoonright X)$ and P-availability of $\gamma'(u \upharpoonright X)$, a occurs in $\ulcorner u_{<p} \upharpoonright X \urcorner$ in an O-move o.

- If o is in A or C, using the previous lemma we have $\ulcorner u_{<p} \upharpoonright X \urcorner \upharpoonright AC = \ulcorner \overline{u_{<p}} \upharpoonright X \urcorner \upharpoonright AC \preccurlyeq \overline{u_{<p}} \upharpoonright AC = \ulcorner u_{<p} \upharpoonright AC \urcorner$, where $\preccurlyeq$ is the subsequence relation.
- Suppose o is in B. Then o is a P-move in $\overline{X}$. By IH, a occurs in an O-move o' of $\ulcorner u_{<o} \upharpoonright AC \urcorner$. Because o is in $\ulcorner u_{<p} \upharpoonright X \urcorner$, by the previous lemma o must belong to $\overline{u_{<p}}$. Consequently, $\overline{u_{<o}} \sqsubseteq \overline{u_{<p}}$. Hence, because o' appears in $\ulcorner u_{<o} \upharpoonright AC \urcorner$, it must also occur in $\ulcorner u_{<p} \upharpoonright AC \urcorner$.

For 2, let $a \in \nu(\overline{u})$ and let m^{Σ} be the first element of $\overline{u}$ containing a. If $a \in \nu(m)$ then if m is in AC we are done. Otherwise, m is a generalised P-move. Then, by 1, a occurs in some O-move o in $\ulcorner u_{<m} \upharpoonright AC \urcorner$ and therefore it occurs in $\ulcorner \underline{u}_{<m} \upharpoonright AC \urcorner$ and, since m appears in $\underline{\overline{u}}$, in $\ulcorner \underline{u} \upharpoonright AC \urcorner$. Otherwise, $a \in \mathsf{dom}(\Sigma)$ and therefore, by definition and the fact that $a \in \mathsf{O}(u \upharpoonright AC)$, $a \in \nu(\ulcorner \underline{u} \upharpoonright AC \urcorner)$. □

Proposition 16. *If $u \in Int(A, B, C)$ then $\gamma(u \upharpoonright AC) \in P_{A \to C}$.*

Proof. Showing that $u \upharpoonright AC$ satisfies alternation, bracketing and visibility follows the same lines as [14]. Frugality of $\gamma(u \upharpoonright AC)$ is by definition of γ.
For P-availability, let $u'p^{\Sigma} \sqsubseteq u$ with p a P-move in AC and $a \in \nu(p) \cap \mathsf{O}(u \upharpoonright AC)$. We need to show that $a \in \nu(\ulcorner \gamma(u' \upharpoonright AC) \urcorner)$, i.e. that $a \in \nu(\ulcorner \underline{u'} \upharpoonright AC \urcorner)$ because of Lemma 13. So let p be a P-move in component X. We have that $a \notin \mathsf{P}(u \upharpoonright X)$, so $a \in \mathsf{O}(u \upharpoonright X)$. By P-availability in $\gamma'(u \upharpoonright X)$ we then have that $a \in \nu(\ulcorner \underline{u'} \upharpoonright X \urcorner)$ and therefore $a \in \nu(\overline{\underline{u'}})$ by Lemma 14. If this happens in AC then we are done. Otherwise, a appears in some generalised P-move in $\overline{u'} \upharpoonright B$ and therefore we can obtain $a \in \nu(\ulcorner \underline{u'} \upharpoonright AC \urcorner)$ by applying Lemma 15.
The P-storage condition is preserved because, by definition, the domains of stores in interaction sequences contain precisely the requisite O-names. All necessary P-names are also present, as they must be P-names in one of the interacting strategies. □

Strategy composition can now be defined as follows.

Definition 17. For strategies $\sigma : A \to B$ and $\tau : B \to C$ we define:

$$\sigma;\tau = \{\gamma(u \upharpoonright AC) \mid u \in Int(A, B, C),\ \gamma'(u \upharpoonright AB) \in \sigma,\ \gamma'(u \upharpoonright BC) \in \tau\}$$

We can see that $\sigma;\tau$ is a set of plays in $A \to C$, by Proposition 16. In order to show that is satisfies the conditions for being a strategy, we relate our current framework to that of games with *full store* of [12].[5] In contrast to our more economical plays, the latter relies on plays $s = m_1^{\Sigma_1} \cdots m_k^{\Sigma_k}$ in which $\mathsf{dom}(\Sigma_i) = \nu(\underline{s}_{\leq m_i})$, $1 \leq i \leq k$, i.e. the store must contain exactly the names introduced in the play thus far. We write P_A^F (F stands for "full store") for the set containing all such plays in the arena A, which we call *full plays* of A, and $\sigma :_F A$ if σ is a strategy on A with respect to P_A^F, in which case we say σ is a *full strategy*.

Given a play $s \in P_A$, we can construct the corresponding set of full plays s^F by following the copy-cat phenomenon: P copies the values of all names that have appeared in the play but are not available to him. Formally, $\epsilon^F = \{\epsilon\}$ and:

$$(so^{\Sigma})^F = \{s'o^{\Sigma \cup \Sigma'} \mid s' \in s^F,\ \mathsf{dom}(\Sigma \cup \Sigma') = \nu(\underline{s}o)\}$$
$$(sp^{\Sigma})^F = \{s'p^{\Sigma \cup \Sigma'} \mid s' \in s^F,\ \mathsf{dom}(\Sigma \cup \Sigma') = \nu(\underline{s}p),\ \forall a \in \mathsf{dom}(\Sigma').\ \Sigma'(a) = s'(a)\}$$

where $s'(a)$ is the value of a in the last store of s'.

Definition 18. For each set of plays $X \subseteq P_A$, its corresponding set of full plays is:

$$X^F = \bigcup_{s \in X} s^F$$

[5] The game model of [12] is fully abstract for a language similar to ours but where names can (only) store other names. The games we have presented so far are a concise presentation of those in [12], modified in the obvious way to accommodate integers in the store instead of names.

In particular, for each strategy σ we have $\sigma^F = \{s' \in s^F \mid s \in \sigma\}$.

The following lemma gives some simple properties of the translation.

Lemma 19. *Let A be a prearena.*

1. *For all $s \in P_A$ and $s' \in s^F$, $\gamma'(s') = s$; for all sets $X \subseteq P_A$, $\gamma'(X^F) = X$.*
2. *For all $\sigma : A$, σ^F satisfies even-prefix closure, equivariance and determinacy for full plays.*

Proof. Part 1 is straightforward. For 2, only determinacy presents some complication. So let $so^\Sigma p_1^{\Sigma_1}, so^\Sigma p_2^{\Sigma_2} \in \sigma^F$ and therefore $\gamma'(so^\Sigma p_i^{\Sigma_i}) = s'o^{\Sigma'} p_i^{\Sigma'_i} \in \sigma$, for $i = 1, 2$. By determinacy of σ, there is π such that $s'o^{\Sigma'} p_1^{\Sigma'_1} = \pi \cdot (s'o^{\Sigma'} p_2^{\Sigma'_2})$ and in particular $\underline{s}'op_1 = \pi \cdot \underline{s}'op_2$ and $\Sigma'_1 = \pi \cdot \Sigma'_2$. We claim that $\Sigma_1 = \pi \cdot \Sigma_2$, Indeed, $\mathsf{dom}(\Sigma_1) = \nu(\underline{s}op_1) = \nu(\underline{s}'op_1) = \nu(\pi \cdot (\underline{s}'op_2)) = \mathsf{dom}(\pi \cdot \Sigma_2)$. Moreover, for all $a \in \mathsf{Av_P}(so^\Sigma p_1^{\Sigma_1})$, $\Sigma_1(a) = \Sigma'_1(a) = (\pi \cdot \Sigma'_2)(a) = \Sigma'_2(\pi(a)) = \Sigma_2(\pi(a)) = (\pi \cdot \Sigma_2)(a)$. Finally, for all $a \in \nu(so^\Sigma) \setminus \mathsf{Av_P}(so^\Sigma)$, by definition, $\Sigma_1(a) = \Sigma(a) = \Sigma_2(a)$. Now, by strong support, $\underline{s}'o = \pi \cdot \underline{s}'o$ implies $so^\Sigma = \pi \cdot so^\Sigma$ and therefore $so^\Sigma p_1^{\Sigma_1} = \pi \cdot (so^\Sigma p_2^{\Sigma_2})$, as required. □

In the full-store framework, there is a corresponding notion of strategy composition, which we denote by $;_F$ (see Appendix A). The next result relates the two frameworks.

Lemma 20. *Suppose $\sigma : A \to B$ and $\tau : B \to C$. Then $(\sigma;\tau)^F = \sigma^F ;_F \tau^F$.*

Proof. The proof uses interaction sequences and composition in the full-store framework and is delegated to Appendix A. □

We can now show the following.

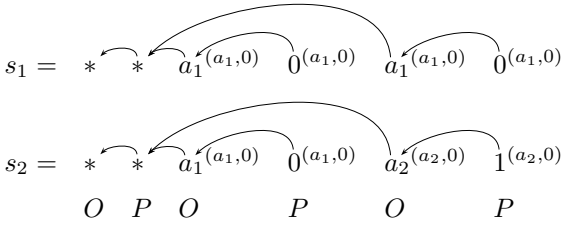
Proposition 21. *Strategy composition is well-defined and associative:*

- *If $\sigma : A \to B$ and $\tau : B \to C$ then $\sigma;\tau : A \to C$.*
- *If $\sigma : A \to B$, $\tau : B \to C$ and $\upsilon : C \to D$ then $\sigma;(\tau;\upsilon) = (\sigma;\tau);\upsilon$.*

Proof. By Proposition 16, $\sigma;\tau \subseteq P_{A \to C}$. Moreover, even-prefix closure and nominal determinacy follow from the definition of composition. For determinacy, note that by the previous two lemmata we have $\sigma;\tau = \gamma'((\sigma;\tau)^F)$ and $(\sigma;\tau)^F = \sigma^F ;_F \tau^F$, where the latter is a full strategy by [12]. Thus, if $sp_1^{\Sigma_1}, sp_2^{\Sigma_2} \in \sigma;\tau$ and $s'p_1^{\Sigma'_1}, s'p_2^{\Sigma'_2}$ are corresponding full plays in $(\sigma;\tau)^F$ then, by determinacy of the latter, we have $s'p_1^{\Sigma'_1} = \pi \cdot (s'p_2^{\Sigma'_2})$, which implies $sp_1^{\Sigma_1} = \pi \cdot (sp_2^{\Sigma_2})$, for some π.
Finally, $\sigma;(\tau;\upsilon) = \gamma'((\sigma;(\tau;\upsilon))^F) = \gamma'(\sigma^F ;_F (\tau^F ;_F \upsilon^F)) = \gamma'((\sigma^F ;_F \tau^F) ;_F \upsilon^F) = \gamma'(((\sigma;\tau);\upsilon)^F) = (\sigma;\tau);\upsilon$, where associativity of full strategies is proved in [12]. □

3.5. Blindness

Although the games we constructed thus far do capture Reduced ML, they are over-expressive for it. In particular, even if strategies are not allowed to play or change the values of O-names which do not appear in the current P-view, they can still express *knowledge* of them by behaving differently when they encounter them again in the play. For example, there is a strategy $\sigma : 1 \to (\mathbb{A} \Rightarrow \mathbb{Z})$ containing the following two plays,



(6)

where $a_1 \neq a_2$. In order to eliminate such behaviours, we need to make strategies *blind* to names that are not currently available to them. In effect, this means slightly reducing the level of history dependence of strategies towards behaviours of purely functional character. The condition we propose next can be seen as a very light version of *innocence* [9].

Given a non-empty justified sequence s, let us write s^- for s without its last element. The following definition aims to capture plays that differ up to renamings applied to names that O introduces in the P-view.

Definition 22. Given a prearena A, $s \in P_A$, $a \notin \mathsf{P}(s)$ and an O-move o in s,

- we say that a is ***P-new at*** o ***in*** s if $a \in \nu(o)$ and $a \notin \nu(\ulcorner s_{\leq o} \urcorner^-)$;
- in such a case, for any $b \in \mathbb{A}$, we say that a is ***renameable for*** b ***at*** o ***in*** s provided $b \notin \mathsf{P}(s)$ and, for all $s' \sqsubseteq s$, if o occurs in $\ulcorner s' \urcorner$ then $b \notin \nu(\ulcorner s' \urcorner)$;
- under the assumptions above, we define the ***renaming*** $(a\ b)_o \cdot s$ of s by induction on the subsequences of s:[6]

$$(a\ b)_o \cdot \epsilon = \epsilon \qquad (a\ b)_o \cdot (tm^{\Sigma}) = \begin{cases} ((a\ b)_o \cdot t)\,((a\ b) \cdot m^{\Sigma}) & o \in \ulcorner tm^{\Sigma} \urcorner \\ ((a\ b)_o \cdot t)\, m^{\Sigma} & o \notin \ulcorner tm^{\Sigma} \urcorner \end{cases}$$

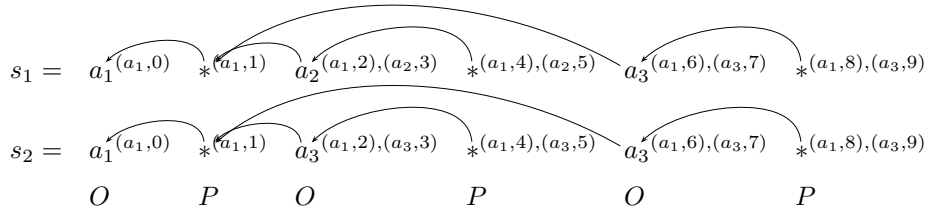
  where $(a\ b)$ is the permutation swapping a with b.

We write $s \overset{r}{\sim} s'$ if s can be obtained from s' through a sequence of renamings.

Observe that, if a is P-new at m in s, a need not be fresh for $s_{<m}$ (the converse holds, though, as long as $a \in \nu(m)$). This reflects the fact that Reduced ML programs cannot recognize whether a specific name has already been used in a play. A play s in which every a that is P-new at m in s is also fresh at $s_{<m}$ will be called ***strict***.

[6] We write $o \in s$ to mean that the distinguished occurrence of o is present in s.

Example 23. Consider the following plays of the prearena $\mathbb{A} \to (\mathbb{A} \Rightarrow 1)$.



Here a_2 is P-new at the third move (also a_2) in s_1, a_2 is renameable for a_3 at that move and $(a_2\ a_3)_{a_2} \cdot s_1 = s_2$. Note also that $(a_3\ a_2)_{a_3} \cdot s_2 = s_1$, where the subscript a_3 stands for the third move of s_2, and that s_1 is strict, whereas s_2 is not.

We next show that renamings are reversible, so $\overset{r}{\sim}$ is an equivalence relation.

Lemma 24. *Let s be a play, and let a be renameable for b at o in s. Then, b is renameable for a at $(a\ b) \cdot o$ in $(a\ b)_o \cdot s$, and $(b\ a)_{(a\ b)\cdot o} \cdot (a\ b)_o \cdot s = s$.*
If, moreover, a is renameable for b in s at $o' \neq o$ then $(a\ b)_o \cdot (a\ b)_{o'} \cdot s = (a\ b)_{o'} \cdot (a\ b)_o \cdot s$.

Proof. Note first that the name b is an O-name in $(a\ b)_o \cdot s$, since it is not a P-name in s and o is the first move where the swap of a and b is applied. Moreover, a cannot be a P-name in $(a\ b)_o \cdot s$, since its first possible occurrence in $(a\ b)_o \cdot s$ must be in a move that hasn't been changed by the renaming, and which could therefore not see a in its P-view. By P-availability, that occurrence would be an O-move.
We have that $b \in \nu((a\ b) \cdot o)$. Moreover, $a, b \notin \nu(\ulcorner s_{\leq o} \urcorner^{-})$ and therefore $b \notin \nu(\ulcorner ((a\ b)_o \cdot s)_{\leq (a\ b)\cdot o} \urcorner^{-})$; hence b is P-new at $(a\ b) \cdot o$. Now, a is not present in the moves that see $(a\ b) \cdot o$ in their P-view and neither in $\ulcorner ((a\ b)_o \cdot s)_{\leq (a\ b)\cdot o} \urcorner^{-} = \ulcorner s_{\leq o} \urcorner^{-}$. Thus, we can apply $(b\ a)_{(a\ b)\cdot o}$ to $(a\ b)_o \cdot s$. The renaming restores a in the places it had been replaced before by b, so we get back s.
Now consider o'. Since a is P-new both at o and o' in s we have that $o \notin \ulcorner s_{\leq o'} \urcorner$, which implies a is P-new at o' in $(a\ b)_o \cdot s$. Moreover, the moves in s that see o' in their P-view are not affected by $(a\ b)_o$, since otherwise we would have a move in s that sees both o and o' in its P-view, which is not possible. Hence, $(a\ b)_{o'}$ is an appropriate renaming for $(a\ b)_o \cdot s$, and similarly $(a\ b)_o$ is appropriate for $(a\ b)_{o'} \cdot s$. Since these renamings cannot both affect the same move in s, their order of application does not matter. □

The previous lemma allows us to define *concurrent renamings*, which will turn out to be useful below. If $K = \{o_1, ..., o_k\}$ is a set of moves of s such that a is renameable for b in s at each o_i, take:

$$(a\ b)_K \cdot s = (a\ b)_{o_1} \cdot \cdots (a\ b)_{o_k} \cdot s$$

Observe that for any play s there exists a strict play s' (determined uniquely up to name-invariance) such that $s \overset{r}{\sim} s'$. We write $\widetilde{s}$ for $[s']$.

Definition 25. A strategy $\sigma : A$ is ***blind*** if $s \in \sigma$ and $s \overset{r}{\sim} s'$ imply $s' \in \sigma$.

Example 26. We can now see that the plays we examined in (6) cannot both be included in a blind strategy. For suppose $s_1, s_2 \in \sigma$. If σ satisfies blindness then:

$$s_2' = (a_1\ a_2)_{a_2} \cdot s_2 = * * a_1^{(a_1,0)} 0^{(a_1,0)} a_1^{(a_1,0)} 1^{(a_1,0)} \in \sigma$$

But now observe that $s_1, s_2' \in \sigma$ violates determinacy of σ.

We next show that blindness is preserved under composition. Note first that we can define P-newness and renamings for interaction sequences as follows. A name a *is P-new in u at o* if o is an O-move in either of $u \upharpoonright AB$, $u \upharpoonright BC$, and:

$$a \in \nu(o) \wedge a \notin \mathsf{P}(u) \wedge a \notin \nu(\ulcorner u_{\leq o} \urcorner^{-})$$

where recall that $\mathsf{P}(u) = \mathsf{P}(u \upharpoonright AB) \cup \mathsf{P}(u \upharpoonright BC)$. In such a case, $(a\ b)_o \cdot u$ is defined accordingly, for any $b \notin \mathsf{P}(u) \cup \nu(\ulcorner u_{\leq o} \urcorner^{-})$ such that:

$$\forall u' \sqsubseteq u.\ o \in \ulcorner u' \urcorner \implies b \notin \nu(\ulcorner u' \urcorner)$$

Lemma 27. *Let $u \in Int(A, B, C)$ and let a be P-new at o in u. Pick a component $X \in \{AB, BC\}$ and set $K = \{o' \in u \mid o \in \ulcorner u_{\leq o'} \urcorner,\ a$ is P-new at o' in $\gamma'(u \upharpoonright X)\}$.*

1. *For any move m in $u \upharpoonright X$, $(\exists o' \in K.\, o' \in \ulcorner u_{\leq m} \upharpoonright X \urcorner) \implies o \in \ulcorner u_{\leq m} \urcorner$.*
2. *For any m in $u \upharpoonright X$ with $a \in \nu(\ulcorner \underline{u}_{\leq m} \upharpoonright X \urcorner)$, $o \in \ulcorner u_{\leq m} \urcorner \implies \exists o' \in K.\, o' \in \ulcorner u_{\leq m} \upharpoonright X \urcorner$.*
3. *For any name b renameable for a in u at o, $\gamma'(((a\ b)_o \cdot u) \upharpoonright X) = (a\ b)_K \cdot \gamma'(u \upharpoonright X)$.*

Proof. 1 is straightforward: if there exists o' such that $o' \in \ulcorner u_{\leq m} \upharpoonright X \urcorner$ then $o' \in \ulcorner u_{\leq m} \urcorner$ (Lemma 15) and, since $o \in \ulcorner u_{\leq o'} \urcorner$, we obtain $o \in \ulcorner u_{\leq m} \urcorner$.
For 2, consider such a move m and let o' be the first O-move in $\ulcorner \gamma'(u_{\leq m} \upharpoonright X) \urcorner$ containing a. Such a move necessarily exists because of P-availability and the fact that $a \in \nu(\underline{u} \upharpoonright X) \setminus \mathsf{P}(u)$ implies $a \in \mathsf{O}(\gamma'(u \upharpoonright X))$. We also have that $o' \in \ulcorner u_{\leq m} \urcorner$. Moreover, $o \in \ulcorner u_{\leq m} \urcorner$ by hypothesis, and therefore either $o \in \ulcorner u_{\leq o'} \urcorner$ or $o' \in \ulcorner u_{\leq o} \urcorner$. Since a is P-new in u at o, it is the case that $o \in \ulcorner u_{\leq o'} \urcorner$, and hence $o' \in K$.
Now, 1 and the fact that b is appropriate for the renaming in u imply that b is appropriate for each of the renamings in $\gamma'(u \upharpoonright X)$: for all $o' \in K$ and $u' \sqsubseteq u$,

$$\begin{aligned} o' \in \gamma'(\ulcorner u' \upharpoonright X \urcorner) &\implies o' \in \ulcorner u' \urcorner \implies o \in \ulcorner u' \urcorner \\ &\implies b \notin \nu(\ulcorner u' \urcorner) \implies b \notin \nu(\gamma'(\ulcorner u' \upharpoonright X \urcorner)). \end{aligned}$$

So take some $m^{\Sigma} \in u \upharpoonright X$ with $a \in \nu(m^{\Sigma})$. If $a \notin \nu(\ulcorner \underline{u}_{\leq m} \upharpoonright X \urcorner)$ then, because of γ', in both the LHS and RHS of 3 we obtain $m^{\Sigma \setminus (a, \Sigma(a))}$. If $a \in \nu(\ulcorner \underline{u}_{\leq m} \upharpoonright X \urcorner)$ then m^{Σ} is not affected (wrt a and b) by γ' in both sides, and by 2 we have that m^{Σ} is renamed in the LHS iff it is in the RHS, as required. □

Proposition 28. *If $\sigma : A \to B$, $\tau : B \to C$ are blind strategies then so is $\sigma;\tau$.*

Proof. Let $u \in Int(A, B, C)$ with $\gamma'(u \upharpoonright AB) \in \sigma$ and $\gamma'(u \upharpoonright BC) \in \tau$, and let b be renameable for a at o in $s = \gamma(u \upharpoonright AC)$. u can be chosen in such a way that $b \notin \nu(u) \setminus \nu(s)$. We need to show that $(a\ b)_o \cdot s \in \sigma;\tau$.
We claim that b is renameable for a at o in u. Indeed, $a \in \mathsf{O}(u \upharpoonright AC)$ implies $a \notin \mathsf{P}(u)$; we have $a \in \nu(o)$; and $a \in \mathsf{O}(u \upharpoonright AC)$, $a \notin \nu(\ulcorner \gamma(u_{\leq o} \upharpoonright AC) \urcorner^{-})$ imply $a \notin \nu(\ulcorner u_{\leq o} \urcorner^{-})$ by Lemma 15. Hence, a is P-new in u at o. If $b \in \mathsf{P}(u)$ then, since $b \notin \nu(u) \setminus \nu(s)$, we have $b \in \nu(s)$ and in fact $b \in \mathsf{P}(s)$, which contradicts renameability in s. Moreover, for each $u' \sqsubseteq u$ with $o \in \ulcorner u' \urcorner$, we have $o \in \ulcorner s' \urcorner$ and therefore $b \notin \nu(\ulcorner s' \urcorner)$, where $s' = \gamma(u' \upharpoonright AC)$. We need to show $b \notin \nu(\ulcorner u' \urcorner)$. If $b \in \nu(s')$ then we must have $b \in \mathsf{O}(u' \upharpoonright AC)$, so $b \notin \nu(\ulcorner u' \urcorner)$ by Lemma 15. Note that $b \in \nu(u') \setminus \nu(s')$ is not possible as it would imply $b \in \mathsf{P}(u)$. Thus, b is appropriate for renaming in u.
Hence, it suffices to show that $(a\ b)_o \cdot \gamma(u \upharpoonright AC) = \gamma(((a\ b)_o \cdot u) \upharpoonright AC)$ and $((a\ b)_o \cdot u) \upharpoonright AB \in \sigma$, $((a\ b)_o \cdot u) \upharpoonright BC \in \tau$. The former is straightforward. From the latter two conditions we show the case of σ. By the previous lemma we have that

$$\gamma'(((a\ b)_o \cdot u) \upharpoonright AB) = (a\ b)_K \cdot \gamma'(u \upharpoonright AB)\,,$$

where K contains moves o' such that a is P-new at o' in $\gamma'(u \upharpoonright AB)$. Hence, since σ is blind and $\gamma'(u \upharpoonright AB) \in \sigma$, we have $(a\ b)_K \cdot \gamma'(u \upharpoonright AB) \in \sigma$, as required. □

Since the identity strategy is blind and blind strategies compose we obtain a ***category $\mathcal{G}$ of arenas and blind strategies***. Observe that blind strategies are uniquely determined by the underlying strict plays via renamings, i.e. two blind strategies are equal if and only if they contain the same strict plays.

4. Soundness

Henceforth, when writing $\sigma : A$ we shall mean a blind strategy on A. Following [12, 1], we show that $\mathcal{G}$ is equivalent to the Klesli category of another category $\mathcal{G}'$ equipped with a strong monad T.

Definition 29. Let $\sigma : A$. We say that σ is ***single-threaded*** if:

- for all $\iota^{\Sigma} \in P_A$ there is $\iota^{\Sigma} p^{\Sigma} \in \sigma$ where p is an answer;
- for all $\iota^{\Sigma} p^{\Sigma} s \in \sigma$,
  - for all Σ' with $\mathsf{dom}(\Sigma') = \mathsf{dom}(\Sigma)$, we have $\iota^{\Sigma'} p^{\Sigma'} s \in \sigma$,
  - there is at most one move in s justified by p.

Thus, a strategy σ is single-threaded if: it immediately answers every initial question without altering the store; its behaviour is independent of the values in the initial store; and it only contains plays consisting of one *thread*. The latter is the call-by-value adaptation of the standard notion of thread of [14, 7]. In particular, a thread of a play s is a subsequence s' of s comprising all moves

of s hereditarily justified by the same threader move. A move in s is called a *threader* move if it is justified by the answer to the initial move. All this is formalised below.

We call a play $s \in P_A$ *total* if $|s| > 1$ implies that s is of the form $\iota^{\Sigma} p^{\Sigma'} s'$ with p an answer move. Let $s \in P_A$ be a total play with $|s| > 2$. We define its *threader move*, written $\mathsf{thrr}(s)$, by:

$$\mathsf{thrr}(s) = m \iff \ulcorner \underline{s} \urcorner = \iota\, p\, m\, s' \quad \text{for some } \iota, p.$$

For $m \in s$, we also write $\mathsf{thrr}(m)$ for $\mathsf{thrr}(s_{\leq m})$, where s is understood from the context. The ***thread*** of s, written $\lceil s \rceil$, is the subsequence of s containing the first two moves of s and all moves-with-store m^{Σ} of s such that $\mathsf{thrr}(m) = \mathsf{thrr}(s)$. Put otherwise:

$$\lceil \iota^{\Sigma} p^{\Sigma'} s' \rceil = \max\{\iota^{\Sigma} p^{\Sigma'} s'' \preccurlyeq \iota^{\Sigma} p^{\Sigma'} s' \mid \forall m \in s''.\, \mathsf{thrr}(m) = \mathsf{thrr}(\iota^{\Sigma} p^{\Sigma'} s')\}$$

where $\preccurlyeq$ is the subsequence relation, and max means of maximum length. We extend the above to all total sequences by setting $\lceil s \rceil = s$ whenever $|s| \leq 2$. Note that, for all even-length $s' p^{\Sigma}$ with length greater than 2, $\mathsf{thrr}(s' p^{\Sigma}) = \mathsf{thrr}(s')$. Therefore, $\lceil s' p^{\Sigma} \rceil = \lceil s' \rceil p^{\Sigma}$ for all total sequences $s' p^{\Sigma}$ of even length.

Observe that, for total plays, $\ulcorner s \urcorner = \ulcorner \lceil s \rceil \urcorner$. Thus, for each total play s ending in a P-move p, P-availability implies that $\nu(p)$ cannot contain any name $a \in \mathsf{O}(s)$ that is not already present in $\lceil s \rceil$. We next introduce a notion of play which restricts even further the names that can appear in P-moves by stipulating that all names which are freshly introduced by a P-move in a thread must be fresh for the whole play.

Definition 30. For each single-threaded strategy $\sigma : A$, we define:

$$\sigma^{\dagger} = \{s \in P_A \mid s \text{ total},\ \forall s' p^{\Sigma} \sqsubseteq s.\ \lceil s' p^{\Sigma} \rceil \in \sigma,\ \nu(p) \cap \nu(\underline{s'}) \subseteq \nu(\lceil \underline{s'} \rceil)\}$$

Note that the last condition above is equivalent to saying that names introduced by P in different threads are always distinct. That is, the condition

$$\phi(s) \ \equiv\ s \text{ total},\ \forall s' \sqsubseteq^{\text{even}} s.\ \lceil s' \rceil \in \sigma,\ (\mathsf{thrr}(s') \neq \mathsf{thrr}(s) \implies \mathsf{P}(s') \cap \mathsf{P}(s) = \emptyset)$$

gives $\sigma^{\dagger} = \{s \in P_A \mid \phi(s)\}$.

Following [12], we can show that $\sigma^{\dagger}$ is a strategy. Moreover, we can define a lluf subcategory $\mathcal{G}'$ of $\mathcal{G}$ containing only single-threaded strategies, with composition $;'$ expressed by:

$$\sigma ;' \tau = \sigma^{\dagger}; \tau$$

The identity in $\mathcal{G}'$ is given by $\mathsf{id}_{A'} = \{\lceil s \rceil \mid s \in \mathsf{id}_A\}$.

We define the following *lifting* functor $T : \mathcal{G}' \to \mathcal{G}'$. On arenas, we have:

$$\begin{array}{ll} M_{TA} = \{*_1, *_2\} + M_A & I_{TA} = \{*_1\} \\ \lambda_{TA} = [\{(*_1, PA), (*_2, OQ)\}, \lambda_A] & \vdash_{TA} = \{(*_1, *_2), (*_2, \iota_A)\} \cup \vdash_A \end{array}$$

whereas for each $\sigma : A \to B$ we set:[7]

$$T\sigma : TA \to TB = \{\epsilon, *_1^A *_1^B\} \cup \{*_1^A *_1^B *_2^B *_2^A \, s \in P_{TA \to TB} \mid \forall \iota_A \in s.\, \gamma'(s \upharpoonright \iota_A) \in \sigma, \\ \forall \iota'_A \neq \iota_A \in s.\, \mathsf{P}(s \upharpoonright \iota_A) \cap \mathsf{P}(s \upharpoonright \iota'_A) = \emptyset\}$$

where s does not contain $*_2^A$ or $*_2^B$, and its restriction to an occurrence ι_A is given by:

$$s \upharpoonright \iota_A = \max\{s' \preccurlyeq s \mid \forall m \in s'.\, \iota_A \in \ulcorner s_{\leq m} \urcorner\}$$

We can now show the following.

Lemma 31. *The category $\mathcal{G}'$ has finite products given by $\otimes$ and the obvious projections, and unit 1. Moreover, T is a strong monad such that $\mathcal{G}'$ has T-exponentials, that is, for all A, B, C there is a bijection $\Lambda^T : \mathcal{G}'(A \otimes B, TC) \cong \mathcal{G}'(A, B \Rightarrow C)$ natural in A, C.*

Proof. Our development of single-threaded strategies followed closely [12], and thus the same arguments as *loc. cit.* apply. Our additional claim, that T is a strong monad, is a standard result for the lifting functor (cf. [35]). □

The above is to say that $\mathcal{G}'$ is λ_c-model [18], which gives a canonical interpretation of the call-by-value λ-calculus in the associated Kleisli category. The latter is equivalent to the category $\mathcal{G}$, by the obvious bijection

$$\Phi : \mathcal{G}(A, B) \cong \mathcal{G}'(A, TB)$$

which adds/removes the two opening moves of TB from plays. Let us write the strong monad T in full as $T = (T, \eta, \mu, st)$, and let ψ be the natural transformation:

$$\psi_{A,B} : TA \otimes TB \xrightarrow{\cong} TB \otimes TA \xrightarrow{st} T(TB \otimes A) \xrightarrow{T\cong} T(A \otimes TB) \xrightarrow{T(st)} TT(A \otimes B) \xrightarrow{\mu} T(A \otimes B)$$

For each $\sigma : A \to B$, $\tau : A \to C$ and $\phi : A \otimes B \to C$ (in $\mathcal{G}$), we define the following strategies.

$$\begin{aligned} \langle\!|\sigma, \tau|\!\rangle : A \to B \otimes C &= \Phi^{-1}\big(A \xrightarrow{\langle \Phi(\sigma), \Phi(\tau)\rangle} TB \otimes TC \xrightarrow{\psi} T(B \otimes C)\big) \\ \Lambda(\phi) : A \to B \Rightarrow C &= (\Lambda^T(\Phi(\phi)))^\dagger \\ \mathsf{ev}_{A,B} : (A \Rightarrow B) \otimes A \to B &= \Phi^{-1}\big((A \Rightarrow B) \otimes A \xrightarrow{\Lambda^{T\,-1}(\mathsf{id}'_{A \Rightarrow B})} TB\big) \end{aligned}$$

Moreover, $\mathcal{G}$ contains coproducts. For arenas A, B, we define $A + B$ by:

$$M_{A+B} = M_A \uplus M_B\,, \quad I_{A+B} = I_A \cup I_B\,, \quad \lambda_{A+B} = [\lambda_A, \lambda_B]\,, \quad \vdash_{A+B} = \vdash_A \cup \vdash_B\,.$$

For each $\sigma : A \to C$ and $\tau : B \to C$, $[\sigma, \tau] : A + B \to C$ is given by $\sigma \cup \tau$ (modulo indexing of moves in the disjoint union $M_A \uplus M_B$).

[7] here by $\iota_A \neq \iota'_A$ we mean that the specific occurrences of ι_A and ι'_A inside s are distinct.

To interpret the remaining constructs of Reduced ML in $\mathcal{G}$, we follow Stark by showing the existence of special morphisms, as described in Chapter 5 of [33] and adjusted in Appendix B to accommodate Ω. We list those related to reference manipulation below (as morphisms in $\mathcal{G}$ rather than in $\mathcal{G}'_T$).

$$\begin{aligned} get &: \mathbb{A} \to \mathbb{Z} = \{\epsilon\} \cup \{a^{(a,i)} i^{(a,i)} \mid a \in \mathbb{A}\} \\ set &: \mathbb{A} \otimes \mathbb{Z} \to 1 = \{\epsilon\} \cup \{(a,i)^{(a,i')} *^{(a,i)} \mid a \in \mathbb{A}\} \\ new &: 1 \to \mathbb{A} = \{\epsilon\} \cup \{* \; a^{(a,0)} \mid a \in \mathbb{A}\} \end{aligned}$$

As a consequence, we conclude that $\mathcal{G}$ is a model of Reduced ML.[8] This lets us interpret any term-in-context $\mathsf{u}, \Gamma \vdash M : \theta$ with a strategy $[\![\mathsf{u}, \Gamma \vdash M : \theta]\!] : [\![\mathsf{u}, \Gamma \vdash \theta]\!]$, denoted also as $[\![M]\!] : [\![\mathsf{u}, \Gamma \vdash \theta]\!]$. The interpretation is given explicitly below. Suppose that $|\mathsf{u}| = n$ and $\Gamma = \{x_1 : \theta_1, \cdots, x_k : \theta_k\}$. We write $[\![\mathsf{u}, \Gamma]\!]$ for the arena $[\![\mathsf{u}]\!] \otimes [\![\theta_1]\!] \otimes \cdots \otimes [\![\theta_k]\!]$.

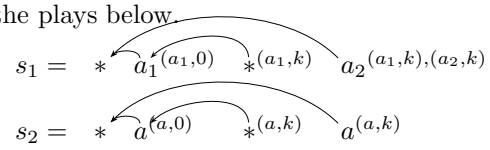
- $[\![\mathsf{u}, \Gamma \vdash () : \mathsf{unit}]\!] = [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\mathsf{t}} 1$, where $\mathsf{t} = \{\epsilon, (\bar{a}, \iota_\Gamma)*\}$.
- $[\![\mathsf{u}, \Gamma \vdash \Omega : \mathsf{unit}]\!] = [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\perp} 1$, where $\perp = \{\epsilon\}$.
- $[\![\mathsf{u}, \Gamma \vdash i : \mathsf{int}]\!] = [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\mathsf{t}} 1 \xrightarrow{\mathsf{i}} \mathbb{Z}$, where $\mathsf{i} = \{\epsilon, *\, i\}$.
- $[\![\mathsf{u}, \Gamma \vdash x_j : \theta_j]\!] = [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\pi_{n+j}} [\![\theta_j]\!]$.
- $[\![\mathsf{u}, \Gamma \vdash M_1 \oplus M_2 : \mathsf{int}]\!] = [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\langle [\![M_1]\!], [\![M_2]\!] \rangle} \mathbb{Z} \otimes \mathbb{Z} \xrightarrow{\sigma_\oplus} \mathbb{Z}$, where $\sigma_\oplus = \{\epsilon, (i_1, i_2)\,(i_1 \oplus i_2)\}$.
- $[\![\mathsf{u}, \Gamma \vdash \mathsf{if}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_2 : \theta]\!] = [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\langle\!| [\![M]\!], \mathsf{id} \rangle} \mathbb{Z} \otimes [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\mathsf{if} \otimes \mathsf{id}} (1+1) \otimes [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\cong} [\![\mathsf{u}, \Gamma]\!] + [\![\mathsf{u}, \Gamma]\!] \xrightarrow{[[\![N_1]\!], [\![N_2]\!]]} [\![\theta]\!]$, where $\mathsf{if} = \{\epsilon,\ i *_l,\ 0 *_r \mid i \neq 0\}$.
- $[\![\mathsf{u}, \Gamma \vdash \mathsf{ref}\ M : \mathsf{int\,ref}]\!] = [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\langle\!| \mathsf{t}; new, [\![M]\!] \rangle} \mathbb{A} \otimes \mathbb{Z} \xrightarrow{\langle\!| set, \pi_1 \rangle} 1 \otimes \mathbb{A} \xrightarrow{\cong} \mathbb{A}$.
- $[\![\mathsf{u}, \Gamma \vdash !M : \mathsf{int}]\!] = [\![\mathsf{u}, \Gamma]\!] \xrightarrow{[\![M]\!]} \mathbb{A} \xrightarrow{get} \mathbb{Z}$.
- $[\![\mathsf{u}, \Gamma \vdash M := N : \mathsf{unit}]\!] = [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\langle\!| [\![M]\!], [\![N]\!] \rangle} \mathbb{A} \otimes \mathbb{Z} \xrightarrow{set} 1$.
- $[\![\mathsf{u}, \Gamma \vdash MN : \theta']\!] = [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\langle\!| [\![M]\!], [\![N]\!] \rangle} ([\![\theta]\!] \Rightarrow [\![\theta']\!]) \otimes [\![\theta]\!] \xrightarrow{\mathsf{ev}} [\![\theta']\!]$.
- $[\![\mathsf{u}, \Gamma \vdash \lambda x.M : \theta \to \theta']\!] = \Lambda([\![M]\!] : [\![\mathsf{u}, \Gamma]\!] \otimes [\![\theta]\!] \to [\![\theta']\!])$.

We have used two isomorphisms above: $(1+1) \otimes [\![\mathsf{u}, \Gamma]\!] \xrightarrow{\cong} [\![\mathsf{u}, \Gamma]\!] + [\![\mathsf{u}, \Gamma]\!]$ and $1 \otimes \mathbb{A} \xrightarrow{\cong} \mathbb{A}$. In both cases, the strategies perform copying between instances

[8] Strictly speaking, following [33, Chapter 5], the categorical conditions of Appendix B are satisfied by the category $\mathcal{G}'$ and yield a model of Reduced ML in the Kleisli category $\mathcal{G}'_T$. Here instead we express the model directly in $\mathcal{G}$, using the bijection: $\mathcal{G}(A, B) \cong \mathcal{G}'(A, TB)$.

of the same arena ($[\![\mathsf{u}, \Gamma]\!]$ and A respectively). In the first case, the initial move (containing a move from $1 + 1$) will determine which copy of $[\![\mathsf{u}, \Gamma]\!]$ from $[\![\mathsf{u}, \Gamma]\!] + [\![\mathsf{u}, \Gamma]\!]$ will be active.

Example 32. The two terms from example (5) in the Introduction are interpreted (in $\mathcal{G}$) by the strategies given respectively (through even-prefix closure and orbiting) by the plays below.



The fact that the interpretations are different means that our model is not complete yet. On the other hand, by showing that the model conforms to a slightly modified variant of Stark's framework (Appendix B), one can conclude Computational Soundness and Adequacy [33, Proposition 5.12].

Proposition 33. *For all terms* $\mathsf{u}, \Gamma \vdash M, N : \theta$, $[\![M]\!] = [\![N]\!] \implies M \cong N$.

5. Definability

Recall that typing judgements $\mathsf{u}, x_1 : \theta_1, \cdots, x_k : \theta_k \vdash M : \theta$ are interpreted as strategies for the prearena $[\![\mathsf{u}]\!] \otimes [\![\theta_1]\!] \otimes \cdots \otimes [\![\theta_n]\!] \to [\![\theta]\!]$. We shall call such prearenas *denotable*.

Our definability argument will rely on the fact that atoms occurring in moves from denotable prearenas can be ordered (for instance, from left to right in moves comprising pairs of submoves). More generally, the set of names occurring in a play of a denotable arena can be (linearly) ordered according to their order of appearance and, if they were introduced in the same move, using the ordering associated with that move. We shall refer to that ordering as *canonical*. Note in particular that the canonical ordering is a nominal relation. That is, if we denote by $<_s$ the canonical ordering on a play s then, for all names a, b and permutations π, $a <_s b \iff \pi(a) <_{\pi \cdot s} \pi(b)$.

Definition 34. Let A be a denotable prearena.

- $\sigma : A$ is ***finitary*** if the set $\{[s] \mid s \in \sigma\}$ is finite.
- $\sigma : A$ is ***strongly deterministic*** if $\mathsf{P}(s) = \emptyset$ for all $s \in \sigma$.
- A strongly deterministic $\sigma : A$ is ***innocent*** if $sp^{\Sigma}, t \in \sigma$, $to^T \in P_A$ and $\ulcorner s \urcorner = \ulcorner to^T \urcorner$ imply $to^T p^{\Sigma} \in \sigma$. An innocent strategy σ is *finitarily innocent* if $\mathsf{vf}(\sigma) = \{\ulcorner s \urcorner \mid s \in \sigma\}$ is finitary, that is, the set $\{[t] \mid t \in \mathsf{vf}(\sigma)\}$ is finite.

Remark 35. Note that innocence implies blindness.

Using two factorizations we will show that any finitary blind strategy in a denotable prearena is definable. The first one eliminates violations of strong determinacy with the help of *new* (corresponding to $\mathsf{ref}\ 0$). The second one factors out non-innocence (also using *new*). Finally, we prove a direct definability result for finitarily innocent strongly deterministic strategies.

Lemma 36. *Let $A = C \to D$ be a denotable prearena and $\sigma : A$ a finitary strategy. There exists $k \in \mathbb{N}$ and a finitary strongly deterministic strategy $\overline{\sigma} : \mathbb{A}^k \otimes C \to D$ such that $(new^k \otimes \mathsf{id}_C); \overline{\sigma} = \sigma$. If σ is blind, then so is $\overline{\sigma}$.*

Proof. Because σ is finitary, there exists a bound on the number of P-names in any play, say, k. Given a play s, let $\overline{\mathsf{P}}(s)$ be the list containing elements of $\mathsf{P}(s)$ in *canonical* order. Further, given $s = m_1^{\Sigma_1} m_2^{\Sigma_2} \cdots m_l^{\Sigma_l} \in P_{C \to D}$ and $s' = (\bar{a}, q_1, \cdots, q_k, m_1)^{\Sigma'_1} m_2^{\Sigma'_2} \cdots m_l^{\Sigma'_l} \in P_{\mathbb{A}^k \otimes C \to D}$, let us write $s \triangle s'$ if

- $\overline{\mathsf{P}}(s) = a_1 \cdots a_{|\mathsf{P}(s)|}$;
- $\Sigma'_i = \Sigma_i \cup \{(a, 0) \mid a \in \mathsf{dom}(\Sigma'_i) \setminus \mathsf{dom}(\Sigma_i)\}$ for any $1 \leq i \leq l$.

Then one can take $\overline{\sigma}$ to be $\{s' \in P_{\mathbb{A}^k \otimes C \to D} \mid \exists s \in \sigma.(s \triangle s')\}$. Preservation of blindness and finitariness follow immediately from the construction. □

Next we prove that finitary strongly deterministic blind strategies can be factored through finitarily innocent ones.

Lemma 37. *Let $A = C \to D$ be a denotable prearena and $\sigma : C \to D$ a finitary strongly deterministic blind strategy. There exists a finitarily innocent strongly deterministic strategy $\dot{\sigma} : \mathbb{A} \otimes C \to D$ such that $(new \otimes \mathsf{id}_C); \dot{\sigma} = \sigma$.*

The standard way [5] of proving such results is to store the history of play using the additional $\mathbb{A}$ component. This is impossible in our case, because atoms cannot be stored. However, given a play s, we can try to work with a numerical representation of s, where atoms are represented by integers denoting their position in the canonical ordering associated with a play. Let us write $\#(s)$ for such an encoding. In particular we have $\#(s_1) = \#(s_2)$ iff $[s_1] = [s_2]$.

Unfortunately, this is not yet sufficient for a successful factorization through an innocent strategy because, given $\#(s)$ and $\ulcorner so^{\Sigma} \urcorner$, it will in general be impossible to extract so^{Σ} (or $[so^{\Sigma}]$) due to the fact that o might contain O-names occurring in s, but not in $\ulcorner so^{\Sigma} \urcorner$. Note, however, that given $\#(s)$ and $\ulcorner so^{\Sigma} \urcorner$ we can still determine $\widetilde{so^{\Sigma}}$ thanks to the absence of P-names. Furthermore, since σ is blind and strongly deterministic (in particular plays satisfy P-availability), we can uniquely identify $p^{\Sigma'}$ such that $so^{\Sigma} p^{\Sigma'} \in \sigma$ (though not necessarily the whole of $so^{\Sigma} p^{\Sigma'}$), by referring to $\widetilde{so^{\Sigma}}$ and σ.[9] Analogously, we can also deduce $\widetilde{so^{\Sigma} p^{\Sigma'}}$. Thus, the familiar factorization technique can be employed provided that, instead of $\#(s)$, the argument will rely on $\#(\widetilde{s})$, where $\#(\widetilde{s})$ stands for $\#(s')$ and s' is a strict play such that $s' \overset{r}{\sim} s$ (by previous remarks the code is independent of the exact choice of s').

[9] Recall that blind strategies are generated by their strict plays. Moreover, recall the notation $\widetilde{s} = [s']$, where s' a strict play such that $s \overset{r}{\sim} s'$.

Proof. Let us assume that $\#(\epsilon) = 0$. Given $s = m_1^{\Sigma_1} m_2^{\Sigma_2} \cdots m_k^{\Sigma_k} \in \sigma$ and $a \notin \nu(s)$, let us define $\dot{s_a} \in P_{\mathbb{A} \otimes C \to D}$ to be $(a, m_1)^{\Sigma_1'} m_2^{\Sigma_2} \cdots m_k^{\Sigma_k'}$, where

$$\Sigma_i' = \begin{cases} \Sigma_i \cup \{(a, \#(\widetilde{s_{<m_i}}))\} & i \text{ is odd,} \\ \Sigma_i \cup \{(a, \#(\widetilde{s_{\leq m_i}}))\} & i \text{ is even.} \end{cases}$$

Observe that $\sigma' = \{\dot{s_a} \,|\, s \in \sigma, a \notin \nu(s)\}$ is a finitary strongly deterministic blind strategy on $\mathbb{A} \otimes C \to D$ such that $(new \otimes \mathsf{id}_C); \sigma' = \sigma$. Next we show that the P-views of σ' define an innocent strategy: $t_1 o_1^{\Sigma_1} p_1^{\Sigma_1'}, t_2 o_2^{\Sigma_2} p_2^{\Sigma_2'} \in \sigma'$ and $\ulcorner t_1 o_1^{\Sigma_1} \urcorner = \ulcorner t_2 o_2^{\Sigma_2} \urcorner$ imply $\ulcorner t_1 o_1^{\Sigma_1} p_1^{\Sigma_1'} \urcorner = \ulcorner t_2 o_2^{\Sigma_2} p_2^{\Sigma_2'} \urcorner$.

For a start, note that $\ulcorner t_1 o_1^{\Sigma_1} \urcorner = \ulcorner t_2 o_2^{\Sigma_2} \urcorner$ implies $o_1 = o_2$ and $\Sigma_1 = \Sigma_2$, so we shall simply write o and Σ to refer to them. Because Σ_1, Σ_2 contain the codes of $\widetilde{t_1}$ and $\widetilde{t_2}$ respectively, we have $\widetilde{t_1} = \widetilde{t_2}$. By $\ulcorner t_1 o^{\Sigma} \urcorner = \ulcorner t_2 o^{\Sigma} \urcorner$, we can also conclude that $\widetilde{t_1 o^{\Sigma}} = \widetilde{t_2 o^{\Sigma}}$. Consequently, we also have $\widetilde{t_1 o^{\Sigma} p_1^{\Sigma_1'}} = \widetilde{t_2 o^{\Sigma} p_1^{\Sigma_1'}}$, because, by strong determinacy, only O-names from $\ulcorner t_1 o^{\Sigma} \urcorner = \ulcorner t_2 o^{\Sigma} \urcorner$ can occur in p_1. Thus $t_1 o^{\Sigma} p_1^{\Sigma_1'} \overset{r}{\sim} t_2 o^{\Sigma} p_1^{\Sigma_1'}$. By blindness of σ', $t_2 o^{\Sigma} p_1^{\Sigma_1'} \in \sigma'$. Because σ' is strongly deterministic and $t_2 o^{S} p_2^{\Sigma_2'} \in \sigma'$ we have $t_2 o^{S} p_1^{\Sigma_1'} = t_2 o^{\Sigma} p_2^{\Sigma_2'}$. Hence $\ulcorner t_1 o^{\Sigma} p_1^{\Sigma_1'} \urcorner = \ulcorner t_2 o^{\Sigma} p_1^{\Sigma_1'} \urcorner = \ulcorner t_2 o^{\Sigma} p_2^{\Sigma_2'} \urcorner$.

To satisfy the Lemma we can now take $\dot{\sigma}$ to be the smallest innocent strategy containing σ'. Note that $(new \otimes \mathsf{id}_C); \dot{\sigma} = \sigma$, as the interaction of σ' and *new* involves exclusively plays from $\dot{\sigma}$. □

Remark 38. The argument showing that σ' can be used to generate an innocent strategy would collapse if its definition relied on $\#(s_{\leq m_i})$ and $\#(s_{<m_i})$ respectively: the problematic plays would be those in which O reuses a name that has disappeared from P-view.

Lemma 39. *Let $\sigma : A$ be a finitarily innocent strategy on a denotable prearena. There exists a Reduced ML term $\mathsf{u}, x_1 : \theta_1, \cdots, x_k : \theta_k \vdash M_\sigma : \theta$ such that*

$$[\![\mathsf{u}, x_1 : \theta_1, \cdots, x_k : \theta_k \vdash M_\sigma : \theta]\!] = \sigma.$$

Proof. We reason by induction on the size of the longest play in σ.

If $\sigma = \{\epsilon\}$, we can take $M_\sigma \equiv \Omega_\theta$ (see Remark 3). Otherwise suppose $(\bar{a}, q_1, \cdots, q_k)^{\Sigma_O} p^{\Sigma_P} \in \sigma$, where $q_i \in \{*\} \cup \mathbb{A} \cup \mathbb{Z}$. Note that, because σ is finitarily innocent, the number of nominal equivalence classes capturing the above plays is finite and each is determined by the values of q_i (where $q_i \in \mathbb{Z}$), by which names in $(\bar{a}, q_1, \cdots, q_k)$ are different/the same and by the values stored in Σ_O. Observe that each possible such scenario can be identified using $=$, ! and constants and, using nested conditionals, one can construct a conditional statement on $x_1, \cdots, x_k$ in which each but one branch corresponds to $(\bar{a}, q_1, \cdots, q_k)^{\Sigma_O}$ such that $(\bar{a}, q_1, \cdots, q_k)^{\Sigma_O}$ is extendable to a play from σ and the last branch is reached in all other scenarios. We shall place Ω_θ in that branch, but in the other cases we are going to construct special terms $M_{(\bar{a}, q_1, \cdots, q_k)^{\Sigma_O}}$ (to be defined soon) which are to be placed in the corresponding branches.

Given a store $\Sigma = \{(b_i, v_i) \,|\, 1 \leq i \leq j\}$, let $\overline{\Sigma} \equiv (\hat{b}_1 := v_1); \cdots ; (\hat{b}_j := v_j)$, where, for each name b, $\hat{b}$ is the leftmost element of $\bar{a}, x_1, \cdots, x_k$ such that the corresponding element of $\bar{a}, q_1, \cdots, q_k$ is b. We write Γ for $x_1 : \theta_1, \cdots, x_k : \theta_k$.

1. Suppose p is an answer. Then p must be an initial move of $[\![\theta]\!]$.

- If θ is a ground type, we set

$$M_{(\bar{a},q_1,\cdots,q_k)^{\Sigma_O}} \equiv \begin{cases} \overline{\Sigma_P}; () & \theta \equiv \mathsf{unit}; \\ \overline{\Sigma_P}; i & \theta \equiv \mathsf{int}, p = i; \\ \overline{\Sigma_P}; x_i & \theta \equiv \mathsf{int\,ref}, p = q_i. \end{cases}$$

- If $\theta \equiv \theta' \to \theta''$ then $p = *$. Let $A = [\![\mathsf{u}, \theta_1, \cdots, \theta_k, \theta' \vdash \theta'']\!]$. Consider the finitarily innocent strategy $\sigma' : A$ defined by: for all $(\bar{a}, q_1, \cdots, q_k, m)^{\Sigma} s \in P_A$,

$$(\bar{a}, q_1, \cdots, q_k, m)^{\Sigma} s \in \mathsf{vf}(\sigma') \iff (\bar{a}, q_1, \cdots, q_k)^{\Sigma_O} p^{\Sigma_P} m^{\Sigma} s \in \mathsf{vf}(\sigma).$$

  By IH we obtain a term $\mathsf{u}, \Gamma, x : \theta' \vdash M_{\sigma'} : \theta''$. We then take

$$M_{(\bar{a},q_1,\cdots,q_k)^{\Sigma_O}} \equiv \overline{\Sigma_P}; (\lambda x^{\theta'} . M_{\sigma'}).$$

2. Suppose p is a question. Then p is a move in $[\![\theta_i]\!]$ and $\theta_i \equiv \theta' \to \theta''$.

- If θ' is a ground type, then $p \in \{*\} \cup \mathbb{Z} \cup \mathbb{A}$. Let $A = [\![\mathsf{u}, \theta_1, \cdots, \theta_k, \theta'' \vdash \theta]\!]$. Consider the finitarily innocent strategy $\sigma' : A$ defined by: for all $(\bar{a}, q_1, \cdots, q_k, m)^{\Sigma} s \in P_A$,

$$(\bar{a}, q_1, \cdots, q_k, m)^{\Sigma} s \in \mathsf{vf}(\sigma') \iff (\bar{a}, q_1, \cdots, q_k)^{\Sigma_O} p^{\Sigma_P} m^{\Sigma} s \in \mathsf{vf}(\sigma).$$

  By IH we obtain a term $\mathsf{u}, \Gamma, x : \theta'' \vdash M_{\sigma'} : \theta$. Then we can take $M_{(\bar{a},q_1,\cdots,q_k)^{\Sigma_O}} \equiv \overline{\Sigma_P}; ((\lambda x^{\theta''} . M_{\sigma'})(x_i M_p))$, where

$$M_p \equiv \begin{cases} () & \theta' \equiv \mathsf{unit}; \\ p & \theta' \equiv \mathsf{int}; \\ \hat{p} & \theta' \equiv \mathsf{int\,ref}. \end{cases}$$

- If $\theta' \equiv \theta'_1 \to \theta'_2$ we proceed in two steps.

  1. Let $A_1 = [\![\mathsf{u}, \theta_1, \cdots, \theta_k, \theta'_1 \vdash \theta'_2]\!]$ and consider the finitarily innocent $\sigma'_1 : A_1$ defined by: for all $(\bar{a}, q_1, \cdots, q_k, m)^{\Sigma} s \in P_{A_1}$,

     $$(\bar{a}, q_1, \cdots, q_k, m)^{\Sigma} s \in \mathsf{vf}(\sigma'_1) \iff (\bar{a}, q_1, \cdots, q_k)^{\Sigma_O} p^{\Sigma_P} m^{\Sigma} s \in \mathsf{vf}(\sigma).$$

     By IH we obtain $\Gamma, x : \theta'_1 \vdash M_{\sigma'_1} : \theta'_2$.
  2. Let $A_2 = [\![\mathsf{u}, \theta_1, \cdots, \theta_k, \theta'' \vdash \theta]\!]$ and consider the finitarily innocent $\sigma'_2 : A_2$ defined by: for all $(\bar{a}, q_1, \cdots, q_k, m)^{\Sigma} s \in P_{A_2}$,

     $$(\bar{a}, q_1, \cdots, q_k, m)^{\Sigma} s \in \mathsf{vf}(\sigma'_2) \iff (\bar{a}, q_1, \cdots, q_k)^{\Sigma_O} p^{\Sigma_P} m^{\Sigma} s \in \mathsf{vf}(\sigma).$$

     By IH we obtain $\Gamma, x : \theta'' \vdash M_{\sigma'_2} : \theta$.

Now we can take $M_{(\bar{a},q_1,\cdots,q_k)^{\Sigma_O}} \equiv \overline{\Sigma_P}; ((\lambda x^{\theta''}.M_{\sigma'_2})(x_i(\lambda x^{\theta'_1}.M_{\sigma'_1})))$.

□

Note that the above proof actually shows that M_σ can be taken to be ref-free. Putting the three lemmas together we obtain the following.

Proposition 40. *Let $\sigma : A$ be a finitary strategy over a denotable prearena. Then there exists a Reduced ML term $\mathsf{u}, \Gamma \vdash M_\sigma : \theta$ such that $[\![\mathsf{u}, \Gamma \vdash M_\sigma : \theta]\!] = \sigma$.*

Thanks to the Proposition, we can now define a fully abstract model of Reduced ML in the usual way by quotienting $\mathcal{G}$ by the induced *intrinsic preorder* defined below.

Definition 41. Suppose $\sigma_1, \sigma_2 : 1 \to A$. We define $\sigma_1 \leq \sigma_2$ to hold iff, for any $\rho : A \to 1$, $\sigma_1; \rho \neq \{\epsilon\}$ implies $\sigma_2; \rho \neq \{\epsilon\}$.

It is common to view the above preorder as *testing σ_i with ρ*, where $\sigma_i; \rho \neq \{\epsilon\}$ is regarded as a successful outcome.

Theorem 42. *Given Reduced ML terms $\vdash M_1 : \theta$ and $\vdash M_2 : \theta$, we have $\vdash M_1 \sqsubseteq_\sim M_2$ iff $[\![\vdash M_1]\!] \leq [\![\vdash M_2]\!]$.*

6. Program equivalence explicitly

Let σ_1, σ_2, ρ be as in the definition of $\leq$. Note that during composition of σ_i with ρ there is a full symmetry between O-names and P-names, i.e. names which are O-names in σ_i are viewed as P-names in ρ, and vice versa. This can be contrasted with the general case of composition, where both strategies may regard a name as an O-name during composition, though not a P-name. This symmetry of roles means that, because plays of ρ satisfy P-availability, a successful outcome can only be reached by interaction with a play of σ_i that satisfies the dual condition, for all $a \in \mathbb{A}$:

- for all $s'o^\Sigma \sqsubseteq^{\text{odd}} s$, if $a \in \nu(o) \cap \nu(s')$ then $a \in \mathsf{Av}_\mathsf{O}(s')$ (*O-Availability*).

Similarly, whenever the play s engages with ρ successfully, the following condition holds, for all $a \in \mathbb{A}$:

- for all $s'p^\Sigma o^{\Sigma'} \sqsubseteq s$, if $a \in \mathsf{P}(s'p^\Sigma) \setminus \mathsf{Av}_\mathsf{O}(s'p^\Sigma)$ then $\Sigma(a) = \Sigma'(a)$ (*O-Passivity*).

This time this is due to the definition of composition, which stipulates that the part of store irrelevant to one of the strategies must be copied. This means that the plays of σ_i that "matter" must necessarily meet the above condition. Finally, whenever $\sigma_i; \rho \neq \{\epsilon\}$, the play witnessing this is ***complete***, i.e. all of its questions are answered.

Definition 43. A play is ***relevant*** if it is complete, satisfies O-availability and O-passivity. We write $\mathsf{rel}(\sigma)$ for the set of relevant plays of σ.

We can represent *relevant* plays more succinctly by restricting the associated stores to mutually available names (both O- and P-available). The outcome is not a play any more, though it remains a legal justified sequence.

Definition 44. Given a prearena A and a sequence $s \in L_A$, s is called a ***protoplay*** if it satisfies the following conditions, for all $a \in \mathbb{A}$.

- For all $s'p^{\Sigma} \sqsubseteq^{\text{even}} s$, if $a \in \nu(p) \cap \nu(s')$ then $a \in \mathsf{Av}_{\mathrm{P}}(s')$ $\;(P\text{-}Availability)$.
- For all $s'o^{\Sigma} \sqsubseteq^{\text{odd}} s$, if $a \in \nu(o) \cap \nu(s')$ then $a \in \mathsf{Av}_{\mathrm{O}}(s')$ $\;(O\text{-}Availability)$.
- For all $s'm^{\Sigma} \sqsubseteq s$, $\mathsf{dom}(\Sigma) = \mathsf{Av}_{\mathrm{P}}(s'm^{\Sigma}) \cap \mathsf{Av}_{\mathrm{O}}(s'm^{\Sigma})$ $\;(PO\text{-}Storage)$.

We write P_A^{pro} for the set of protoplays on a prearena A.

Let γ'' be the obvious operation on justified sequences that simply erases the O-unavailable names in stores. Although some information about σ is seemingly lost by applying γ'' to $\mathsf{rel}(\sigma)$, the missing values turn out to be inessential for testing. By O-passivity, the lost values of O-unavailable names can be uniquely retrieved in O-moves, by copying values from the preceding P-moves. However, more surprisingly, it does not matter what values such names have in P-moves either. This is because the names are then P-unavailable for ρ and, during composition, are dealt with *uniformly* by propagation as long as they remain unavailable.

We take advantage of the fact that the test ρ is a blind strategy. Recall that blind strategies are uniquely determined by their strict plays, i.e. plays in which O-names fresh in the P-view must be genuinely fresh at the point of introduction. Consequently, if we want to check if σ_i passes the ρ test, we can take advantage of the fact that any contribution from ρ will originate from a strict play. Let $s'' = \gamma''(s')$ $(s' \in \mathsf{rel}(\sigma))$ be a protoplay generated by σ_i. To test whether s'' represents a renaming of a strict play from ρ, it suffices to "refresh" P-names in s'' and try to match it with that the strict play. The desired refreshing operation (for P-names using O-views) is entirely dual to renamings introduced in Definition 22, albeit defined on protoplays.

Definition 45. Given a prearena A, $s \in P_A^{\mathsf{pro}}$, $a \in \mathsf{P}(s)$ and a P-move p in s,

- we say that a is ***O-new at*** p ***in*** s if $a \in \nu(p)$ and $a \notin \nu(\llcorner s_{\leq p} \lrcorner^{-})$;
- in such a case, for any $b \in \mathbb{A}$, we say that a is ***renameable for*** b ***at*** p ***in*** s provided $b \notin \mathsf{O}(s)$ and, for all $s' \sqsubseteq s$, if p occurs in $\llcorner s' \lrcorner$ then $b \notin \nu(\llcorner s' \lrcorner)$;
- under the assumptions above, we define the ***dual renaming*** $(a\ b)^p \cdot s$ of s by induction on the subsequences of s:

$$(a\ b)^p \cdot \epsilon = \epsilon \qquad (a\ b)^p \cdot (tm^{\Sigma}) = \begin{cases} ((a\ b)^p \cdot t)\,((a\ b) \cdot m^{\Sigma}) & p \in \llcorner tm^{\Sigma} \lrcorner \\ ((a\ b)^p \cdot t)\, m^{\Sigma} & p \notin \llcorner tm^{\Sigma} \lrcorner \end{cases}$$

We write $s \underset{r}{\sim} s'$ if s' can be obtained from s through a sequence of dual renamings. We say that a protoplay s is ***dually strict*** if, whenever a name a is O-new at a move p in s, then $a \notin \nu(s_{<p})$.

Now, for any $\sigma : A$, let us define:

$$\widehat{\sigma} = \{s \in P_A^{\mathsf{pro}} \mid \ s \text{ dually strict}, \ \exists s'.\ s' \in \mathsf{rel}(\sigma), \ s \underset{r}{\sim} \gamma''(s')\}\,.$$

Lemma 46. *For any* $\sigma_1, \sigma_2 : 1 \to A$, $\widehat{\sigma_1} \subseteq \widehat{\sigma_2} \implies \sigma_1 \leq \sigma_2$.

Proof. Assume $\widehat{\sigma_1} \subseteq \widehat{\sigma_2}$ and take some $\rho : A \to 1$ such that $\sigma_1; \rho = \{**\}$, the latter obtained by some interaction $* \, u_1 \, *^{\Sigma_1} \in \mathit{Int}(1, A, 1)$. Let $\gamma'(* \, u_1) = * \, s_1$ and $\gamma'(u_1 *^{\Sigma_1}) = t_1 *^{\Sigma_1'}$, so $* s_1 \in \sigma_1$ and $t_1 *^{\Sigma_1'} \in \rho$.
Note that, since all name-introductions happen in A, $\mathsf{P}(* s_1) = \mathsf{O}(t_1)$ and $\mathsf{O}(* s_1) = \mathsf{P}(t_1)$. Moreover, P-views in $* s_1$ are O-views in t_1 and viceversa. Hence, the fact that t_1 satisfies P-availability and $* u_1 *^{\Sigma_1}$ is an interaction implies that $* s_1$ satisfies O-availability and O-passivity; dually, $t_1 *^{\Sigma_1'}$ satisfies O-availability and O-passivity. Thus, $* s_1 \in \mathsf{rel}(\sigma_1)$. Because $\widehat{\sigma_1} \subseteq \widehat{\sigma_2}$, there is some $* s_2 \in \mathsf{rel}(\sigma_2)$ such that $\gamma''(* s_1) \underset{r}{\sim} \gamma''(* s_2)$. Now,

$$\gamma''(* s_1) = \gamma''(\gamma'(* u_1)) \overset{(*)}{=} \gamma'(\gamma''(* u_1)) \overset{(**)}{=} \gamma'(* \, t_1)\,,$$

where $(*)$ holds because γ' and γ'' remove from the stores O-names and P-names respectively and are therefore independent, and $(**)$ holds because γ'' on $1 \to A$ is γ' on $A \to 1$. Now, observing that O-novelty in $1 \to A$ translates to P-novelty in $A \to 1$, we have that, for any series of dual renamings $\overline{(a \ b)_m}\cdot$,

$$\gamma''(* s_2) = \overline{(a \ b)^m}\cdot\gamma''(* s_1) = \overline{(a \ b)^m}\cdot\gamma'(* \, t_1) = \gamma'(*(\overline{(a \ b)_m}\cdot t_1)) = \gamma'(* \, t_2)\,, \quad (7)$$

where we take $t_2 *^{\Sigma_2'}$ such that $t_2 *^{\Sigma_2'} = \overline{(a \ b)_m}\cdot(t_1 *^{\Sigma_1'})$, so $t_2 *^{\Sigma_2'} \in \rho$. The above implies that $* \underline{s}_2 = * \, \underline{t}_2$, say $* \underline{s}_2 = * v$. We now construct a legal interaction $* \, u_2 \, *^{\Sigma_2}$. We first construct $* \, u_2$ by adding stores to $* v$ as follows. Suppose $* u'$ has already been constructed and let m be the next move to be examined in $* v$. That move corresponds to a move-with-store m^{Σ} in $* s_2$ and to another $m^{\Sigma'}$ in $* \, t_2$. Add then to $* u'$ the move m with store $\Sigma \cup \Sigma'$. The latter is a legal store because of (7): if $a \in \mathsf{dom}(\Sigma) \cap \mathsf{dom}(\Sigma')$ then a is P-available both at m^{Σ} and at $m^{\Sigma'}$, therefore it is also O-available at m^{Σ} and so it appears in (7). Moreover, $\mathsf{dom}\,(\Sigma \cup \Sigma')$ contains all names $a \in \nu(* \, \underline{u'} m)$: each such a is either a P-name in $1 \to A$ (hence in $\mathsf{dom}(\Sigma)$) or a P-name in $A \to 1$ (hence in $\mathsf{dom}(\Sigma)'$). It contains no more names, because of frugality of its projections. Thus,

$$\mathsf{dom}(\Sigma \cup \Sigma') = \nu(* \underline{u'} m) = \mathsf{P}(* u' m^{\Sigma \cup \Sigma'} \upharpoonright 1, A) \cup \mathsf{P}(* u' m^{\Sigma \cup \Sigma'} \upharpoonright A, 1)\,.$$

Finally, we form $* \, u_2 \, *^{\Sigma_2}$ by taking Σ_2 to be the last store in u_2, updated by Σ_2'. We now have $* \, u_2 \, *^{\Sigma_2} \in \mathit{Int}(1, A, 1)$ as it also satisfies the store-change conditions; the latter follows from the fact that $* s_2 = \gamma'(* \, u_2)$ and $t_2 *^{\Sigma_2'} = \gamma'(u_2 *^{\Sigma_2})$ satisfy P-storage. We therefore have $** = \gamma(**^{\Sigma_2}) \in \sigma_2; \rho$, as required. □

In order to prove the implication in the opposite direction we need to define the following class of strategies.

Definition 47. Let s be a play of prearena A. Define the least strategy on A containing s by:

$$\rho_s = \{\ \pi \cdot s'' \mid \pi \in \textsc{Perm},\ \exists s' \sqsubseteq^{\text{even}} s.\ s' \overset{r}{\sim} s''\ \}\,.$$

In the following we denote for brevity moves-with-stores by $x, y, etc.$

Lemma 48. *ρ_s defined as above is a (blind) strategy.*

Proof. Blindness is obvious by construction, so we need only show that ρ_s is indeed a strategy, that is, it satisfies determinacy. So let $sxy_1, sxy_2 \in \rho_s$, say $sxy_i \overset{r}{\sim} \pi_i \cdot s'x'y'$, $s'x'y' \sqsubseteq^{\text{even}} s$, for $i = 1, 2$. Let us set $s_i x_i \hat{y}_i = \pi_i^{-1} \cdot sxy_i$ and $\pi = \pi_1^{-1} \circ \pi_2$, so $s_1x_1 = \pi \cdot s_2x_2$. We then have that $s_1x_1\hat{y}_1 \overset{r}{\sim} s_2x_2\hat{y}_2$, say $s_1x_1\hat{y}_1 = (a_1\ b_1)_{m_1} \cdot \ldots (a_n\ b_n)_{m_n} \cdot s_2x_2\hat{y}_2$. Let $i_1 ... i_k$ be the subsequence of $1...n$ containing all the indices of the renamings affecting $\hat{y}_2$, i.e. all i's such that $m_i \in \ulcorner (a_{i+1}\ b_{i+1})_{m_{i+1}} \cdot \ldots (a_n\ b_n)_{m_n} \cdot s_2x_2\hat{y}_2 \urcorner$. We have:

$$\ulcorner s_1x_1\hat{y}_1 \urcorner = (a_{i_1}\ b_{i_1}) \cdot \ldots (a_{i_k}\ b_{i_k}) \cdot \ulcorner s_2x_2\hat{y}_2 \urcorner\,.$$

Set $\pi' = (a_{i_1}\ b_{i_1}) \circ \cdots \circ (a_{i_k}\ b_{i_k})$. Now let $\overline{a}_i$ be the ordering of the set $\mathsf{P}(s_ix_i)$, for $i = 1, 2$, which is obtained from the canonical ordering on $\nu(s_ix_i)$. Since renamings do not affect P-names and $s_1x_1 \overset{r}{\sim} s_2x_2$, we have $\overline{a}_1 = \overline{a}_2$, say $\overline{a}_i = \overline{a}$. Clearly, $\pi' \cdot \overline{a} = \overline{a}$; moreover, $s_1x_1 = \pi \cdot s_2x_2$ implies $\pi \cdot \overline{a} = \overline{a}$. Thus, we have:

$$[(\overline{a}_1, \ulcorner s_1x_1 \urcorner), s_1x_1] = [(\overline{a}_2, \ulcorner s_2x_2 \urcorner), s_2x_2]\,,\ \ [(\overline{a}_1, \ulcorner s_1x_1 \urcorner), \hat{y}_1] = [(\overline{a}_2, \ulcorner s_2x_2 \urcorner), \hat{y}_2]\,,$$

and $\nu(\hat{y}_i) \cap \nu(s_ix_i) \subseteq \nu(\overline{a}_i, \ulcorner s_ix_i \urcorner)$, for $i = 1, 2$, by P-availability. Hence, by Lemma 55 we obtain $[s_1x_1\hat{y}_1] = [s_2x_2\hat{y}_2]$, and therefore $[sxy_1] = [sxy_2]$. □

Lemma 49. *For any $\sigma_1, \sigma_2 : 1 \to A$, $\sigma_1 \leq \sigma_2 \implies \widehat{\sigma_1} \subseteq \widehat{\sigma_2}$.*

Proof. Suppose $\sigma_1 \leq \sigma_2$ and take some $\hat{s}_1 \in \widehat{\sigma_1}$, so there is some $s_1 \in \sigma_1$ such that $\hat{s}_1 \underset{r}{\sim} \gamma''(s_1)$. s_1 is of the form $*s_1'$. Then, $*s_1'* \in \mathit{Int}(1, A, 1)$ apart from the fact that some names are missing from its stores. Take then $*u_1*^{\Sigma_1}$ to be the result of filling those stores with the missing names valued 0; we have $*u_1*^{\Sigma_1} \in \mathit{Int}(1, A, 1)$. Let $t = \gamma'(u_1*^{\Sigma_1})$ and form the strategy $\rho_t : A \to 1$. We have that $*u_1*^{\Sigma_1}$ is an interaction of σ_1 and ρ_t, so $\sigma_1; \rho_t = \{**\}$ and therefore $\sigma_2; \rho_t = \{**\}$. So let $*u_2*^{\Sigma_2} \in \mathit{Int}(1, A, 1)$ with $\gamma'(*u_2) \in \sigma_2$ and $\gamma'(u_2*^{\Sigma_2}) \in \rho_t$. Because of the way ρ_t is constructed, we may choose u_2 in such a way that $\gamma'(u_2*^{\Sigma_2}) \overset{r}{\sim} t = \gamma'(u_1*^{\Sigma_1})$. Because of duality of renamings with dual renamings and of γ' with γ'', the latter implies that $\gamma''(*u_2) \underset{r}{\sim} \gamma''(*u_1)$, and therefore

$$\gamma'(\gamma''(*u_2)) \underset{r}{\sim} \gamma'(\gamma''(*u_1))\,,\ \ \therefore \gamma''(\gamma'(*u_2)) \underset{r}{\sim} \gamma''(\gamma'(*u_1)) = \gamma''(s_1) = \hat{s}_1\,.$$

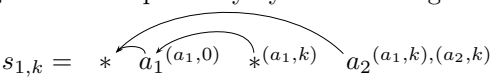
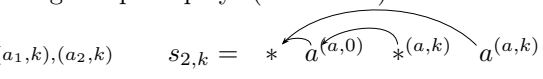
Now, $\gamma'(*u_2) \in \sigma_2$ and the fact that $\hat{s}_1$ is dually strict imply that $\hat{s}_1 \in \widehat{\sigma_2}$, as required. □

Observe that $\widehat{\sigma}$, like σ, is saturated under renamings (extended to act on protoplays). This makes it possible to simplify the above result along the following lines. We call a protoplay ***mutually strict*** if it is both strict and dually strict. Note that by using $\overset{r}{\sim}$ and $\underset{r}{\sim}$ (in any order) we can convert a protoplay to a mutually strict protoplay, unique up to name permutations. Given $\sigma : A$, let $\widehat{\widehat{\sigma}}$ be $\{s \in P_A^{\mathsf{pro}} \mid s \text{ is mutually strict}, \exists s' \in \mathsf{rel}(\sigma).\, s\, (\overset{r}{\sim}; \underset{r}{\sim})\, \gamma''(s')\}$.

Theorem 50. *Given $\sigma_1, \sigma_2 : 1 \to A$, $\sigma_1 \leq \sigma_2$ iff $\widehat{\widehat{\sigma_1}} \subseteq \widehat{\widehat{\sigma_2}}$.*

It follows that terms of Reduced ML are equivalent if and only if they induce the same mutually strict complete protoplays. In particular, this explicit characterisation means that the intrinsic quotient of $\mathcal{G}$ by $\leq$ is effectively presentable [30]. Note that effective presentability concerns compact elements only. Hence, there is no contradiction between our result and the undecidability result for contextual equivalence of finitary Reduced ML [19], because the latter uses terms whose denotations are not compact.

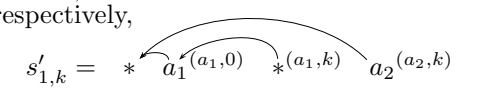
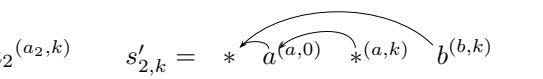
Example 51. We revisit the two terms from example (5) in the Introduction. We previously saw that their translation is given by the strategies σ_1 and σ_2 generated respectively by the following complete plays (for each k).

$s_{1,k} =$  $s_{2,k} =$ 

The former play is not a protoplay as it breaks PO-storage. On the other hand, the latter one is a protoplay but is not dually strict. We compute, for $i = 1, 2$:

$$\begin{aligned} \widehat{\sigma_i} &= \{s \in P_A^{\mathsf{pro}} \mid s \text{ dually strict},\ s \underset{r}{\sim} \gamma''(s_{i,k})\} \\ \gamma''(s_{1,k}) &= * \, a_1^{(a_1,0)} \, *^{(a_1,k)} \, a_2^{(a_2,k)} \\ \gamma''(s_{2,k}) &= s_{2,k} \ \underset{r}{\sim} \ s'_{2,k} = * \, a^{(a,0)} \, *^{(a,k)} \, b^{(b,k)} \end{aligned}$$

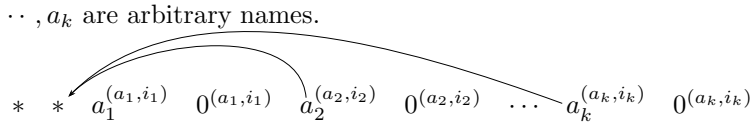
where $a \neq b$ and $s'_{2,k} = (a\ b)^a \cdot s_{2,k}$, with the P-move a being the last move of $s_{2,k}$. Let us set $s'_{1,k} = \gamma''(s_{1,k})$. Note that $s'_{1,k} \underset{r}{\sim} s'_{1,k}$ and observe that both $s'_{i,k}$ are dually strict. Thus, $\widehat{\sigma_1}$ and $\widehat{\sigma_2}$ comprise the orbits of $s'_{1,k}$ and $s'_{2,k}$ respectively,

$s'_{1,k} =$  $s'_{2,k} =$ 

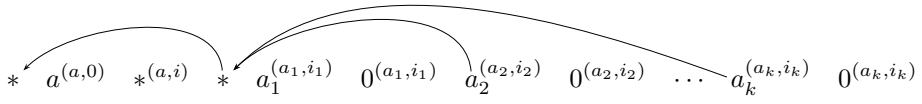
for all k, and hence $\widehat{\sigma_1} = \widehat{\sigma_2}$. Thus, the initial terms are equivalent.

Example 52. The mutually strict complete protoplays for equivalences (2), (3) and (4) of the Introduction are given as follows. We draw pointers only if they do not point at the preceding moves.

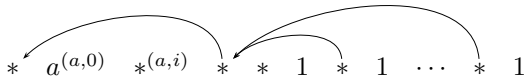
(2): $a_1, \cdots, a_k$ are arbitrary names.



(3): $a_1, \cdots, a_k$ are arbitrary names different from a.



(4): a is an arbitrary name.



7. Conclusion

We presented a fully abstract model for Reduced ML, a call-by-value higher-order language with integer references. Along with [16, 1, 10, 11, 12, 20, 34, 35], the paper is part of a series of works that address the bad-variable problem present in early game models. From that perspective, the present paper eliminates the shortcomings of [4]. In a follow-up paper [24], we proposed a fully abstract and effectively presentable game model for a higher-order language with general references, which can be viewed as a bad-variable-free refinement of [2].

Another direction we pursued concerns algorithmic representations of our game model, from which we derived a procedure for deciding program equivalence for a low-order fragment of Reduced ML (with iteration and a bounded integer type), via an appropriate translation into automata over infinite alphabets [23]. We have also carried out a similar analysis for a language with full ground storage (i.e. references to integers, and references to references to integers, etc.), providing a full characterisation of the decidable cases [25].

Appendix A. Correspondence with the full-store framework

We briefly recall how full strategies compose.

Definition 53. A justified sequence on $A \to B \to C$ is a *full interaction sequence* of A, B, C if $\gamma(u \upharpoonright AB) \in P^F_{A\to B}$, $\gamma(u \upharpoonright BC) \in P^F_{B\to C}$ and the following conditions hold.

- $\mathsf{O}(u \upharpoonright AC) \cap (\mathsf{P}(u \upharpoonright AB) \cup \mathsf{P}(u \upharpoonright BC)) = \emptyset$;
- $\mathsf{P}(u \upharpoonright AB) \cap \mathsf{P}(u \upharpoonright BC) = \emptyset$;
- for all $u' \sqsubseteq u$ ending in $m^{\Sigma} m'^{\Sigma'}$ and $a \in \mathsf{dom}(\Sigma')$ if
  - m' is a P-move in $A \to B$ and $a \notin \nu(\underline{u'} \upharpoonright AB)$,
  - or m' is a P-move in $B \to C$ and $a \notin \nu(\underline{u'} \upharpoonright BC)$,
  - or m' is an O-move in $A \to C$ and $a \notin \nu(\underline{u'} \upharpoonright AC)$,

  then $\Sigma'(a) = \Sigma(a)$;

- for all $u' \sqsubseteq u$ ending in m^{Σ}, $\mathsf{dom}(\Sigma) = \nu(\underline{u'})$.

The set of all full interaction sequences of A, B, C is denoted by $Int_F(A, B, C)$.

Reasoning as with ordinary plays we have that $\gamma(u \upharpoonright AC) \in P^F_{A \to C}$ (here one does not need to verify P-availability and P-storage). Moreover, for each $u \in Int(A, B, C)$ define the set of full justified sequences u^F by $\epsilon^F = \{\epsilon\}$ and:

$$(uo^{\Sigma})^F = \{u' o^{\Sigma \cup \Sigma'} \mid u' \in u^F,\ \mathsf{dom}(\Sigma \cup \Sigma') = \nu(\underline{u}o)\} \tag{A.1}$$

$$(up^{\Sigma})^F = \{u' p^{\Sigma \cup \Sigma'} \mid u' \in u^F,\ \mathsf{dom}(\Sigma \cup \Sigma') = \nu(\underline{u}p),\ \forall a \in \mathsf{dom}(\Sigma').\, \Sigma'(a) = u'(a)\}$$

where o is an O-move in $A \to C$, p a generalised P-move, and $u'(a)$ is the value of a in the last store of u'.

Full strategies $\sigma :_F A \to B$ and $\tau :_F B \to C$ are then composed as follows.

$$\sigma ;_F \tau = \{\gamma(u \upharpoonright AC) \mid u \in Int_F(A, B, C),\ \gamma(u \upharpoonright AB) \in \sigma,\ \gamma(u \upharpoonright BC) \in \tau\}$$

The above is shown to be well-defined in [12] (making also implicitly use of Lemma 55).

Lemma 20. Suppose $\sigma : A \to B$ and $\tau : B \to C$. Then $(\sigma ; \tau)^F = \sigma^F ;_F \tau^F$.

Proof. ($\subseteq$) Let $s \in (\sigma ; \tau)^F$. There exists $u \in Int(A, B, C)$ such that $\gamma(u \upharpoonright AC) = \gamma'(s) \in \sigma ; \tau$, $\gamma'(u \upharpoonright AB) \in \sigma$ and $\gamma'(u \upharpoonright BC) \in \tau$. By making appropriate choices of Σ' in (A.1) we can obtain $u_F \in u^F$ such that $s = \gamma(u_F \upharpoonright AC)$. We claim that $u_F \in Int_F(A, B, C)$. By construction, $\gamma(u_F \upharpoonright AB) \in P^F_{A \to B}$, $\gamma(u_F \upharpoonright BC) \in P^F_{B \to C}$ and $\mathsf{dom}(\Sigma) = \nu(\underline{t}m)$ for each $tm^{\Sigma} \sqsubseteq u_F$. Let us now take $t = t' m^{\Sigma'} m'^{T'} \sqsubseteq u_F$ with $\gamma'(t)$ ending in $m^{\Sigma} m'^{T}$.

- Suppose m' is a P-move in $X \in \{AB, BC\}$ and $a \in \mathsf{dom}(T') \setminus \nu(\underline{t} \upharpoonright X)$. If $a \in \mathsf{dom}(T') \setminus \mathsf{dom}(T)$ then $T'(a) = \Sigma'(a)$ by construction of u_F. If $a \in \mathsf{dom}(T) \setminus \nu(\underline{t} \upharpoonright X)$, then $T'(a) = \Sigma'(a)$, because $u \in Int(A, B, C)$.

- Suppose m' is a O-move in AC and $a \in \mathsf{dom}(T') \setminus \nu(\underline{t} \upharpoonright AC)$. Observe that by Lemma 13, $a \in \mathsf{P}(t \upharpoonright AB) \cup \mathsf{P}(t \upharpoonright BC)$, so $a \in \mathsf{dom}(T) \setminus \nu(\underline{t} \upharpoonright AC)$ and $T'(a) = \Sigma'(a)$ follows, because $u \in Int(A, B, C)$.

It remains to show that $\gamma(u_F \upharpoonright AB) \in \sigma^F$ and $\gamma(u_F \upharpoonright BC) \in \tau^F$. Indeed, we have $\gamma'(\gamma(u_F \upharpoonright AB)) = \gamma'(u_F \upharpoonright AB) = \gamma'(u \upharpoonright AB) \in \sigma$, so $(\gamma'(\gamma(u_F \upharpoonright AB)))^F \subseteq \sigma^F$. To finish the argument, we show $\gamma(u_F \upharpoonright AB) \in (\gamma'(\gamma(u_F \upharpoonright AB)))^F$. For the latter, it suffices to show that P-moves in $\gamma(u_F \upharpoonright AB)$ copy values of unavailable names from the store of their preceding move.
So take $t \sqsubseteq \gamma(u_F \upharpoonright AB)$ such that the last move of t is a P-move and $a \in \mathsf{O}(t) \setminus \nu(\ulcorner \underline{t} \urcorner)$. Then $t = t' m^{\Sigma''} m'^{T''}$ and, by switching, m and m' are consecutive in u with stores, say Σ, T respectively, and in u_F with stores, say Σ', T'. We want to show $\Sigma''(a) = T''(a)$.

- If $a \notin \mathsf{dom}(T)$ then $T''(a) = \Sigma''(n)$ follows from the construction of u_F, since $T'(a) = \Sigma'(a)$.

- If $a \in \mathsf{dom}(T)$, we appeal to the fact that $u \in Int(A, B, C)$.

The case of $\gamma(u_F \upharpoonright BC) \in \tau^F$ is analogous. Hence $s \in \sigma^F ;_F \tau^F$.

($\supseteq$) Let $s \in \sigma^F ;_F \tau^F$, i.e. there exists $u_F \in Int_F(A, B, C)$ such that $s = \gamma(u_F \upharpoonright AC)$, $\gamma(u_F \upharpoonright AB) \in \sigma^F$ and $\gamma(u_F \upharpoonright BC) \in \tau^F$. Let u be the obvious restriction of u_F (only domains of stores are restricted) to fit the definition of $Int(A, B, C)$. We show that $u \in Int(A, B, C)$. Let $t = t' m^{\Sigma} m'^{T} \sqsubseteq u$ by due to some $t'' m^{\Sigma'} m'^{T'} \sqsubseteq u_F$.

- Suppose m' is a P-move in $X \in \{AB, BC\}$ and $a \in \mathsf{dom}(T) \setminus \mathsf{Av}_{\mathrm{P}}(\underline{t} \upharpoonright X)$. If $a \notin \nu(\underline{t} \upharpoonright X)$ then $T(a) = \Sigma(a)$ follows from the fact that $u_F \in Int_F(A, B, C)$ and u is its restriction. Otherwise, $a \in (\mathsf{dom}(T) \cap \nu(\underline{t} \upharpoonright X)) \setminus \mathsf{Av}_{\mathrm{P}}(\underline{t} \upharpoonright X)$ and hence $a \in \mathsf{O}(\underline{t} \upharpoonright X) \setminus \nu(\ulcorner \underline{t} \upharpoonright X \urcorner)$. Since $\gamma(u_F \upharpoonright X) \in \sigma^F$ or $\gamma(u_F \upharpoonright X) \in \tau^F$, $T(a) = \Sigma(a)$ follows from definition of $(\_)^F$.

- Suppose m' is an O-move in AC and $a \in \mathsf{dom}(T) \setminus \mathsf{Av}_{\mathrm{P}}(\underline{t} \upharpoonright AC)$. Then $a \notin \nu(\ulcorner \underline{t} \upharpoonright AC \urcorner)$ so $a \in \mathsf{P}(\underline{t} \upharpoonright AB) \cup \mathsf{P}(\underline{t} \upharpoonright BC)$ and $a \notin \mathsf{O}(\underline{t} \upharpoonright AC)$. The latter implies $a \notin \nu(\underline{t} \upharpoonright AC)$ so $T(a) = \Sigma(a)$ follows from $u_F \in Int_F(A, B, C)$.

Because $\gamma'(u \upharpoonright AB) = \gamma'(u_F \upharpoonright AB) \in \sigma$ and $\gamma'(u \upharpoonright BC) = \gamma'(u_F \upharpoonright BC) \in \tau$, we obtain $\gamma(u \upharpoonright AC) \in \sigma ; \tau$. To complete the argument, we show that $s = \gamma(u_F \upharpoonright AC) \in (\gamma(u \upharpoonright AC))^F$. For the latter, it suffices to show that P-moves in s copy values of unavailable names from the store of their preceding move.
So take $t = t' o^{\Sigma} p^{\Sigma'} \sqsubseteq s$ and $a \in \mathsf{O}(\underline{t}) \setminus \nu(\ulcorner \underline{t} \urcorner)$. Let $v = v' o^{\Sigma_0} m_1^{\Sigma_1} \cdots m_j^{\Sigma_j} p^{\Sigma_{j+1}} \sqsubseteq u$ be such that $\gamma(v \upharpoonright AC) = t$. Since $a \in \mathsf{O}(\underline{t})$, for any $X \in \{AB, BC\}$ we have $a \notin \nu(\underline{v} \upharpoonright X)$ or $a \in \mathsf{O}(\underline{v} \upharpoonright X)$. In the latter case, because $a \notin \nu(\ulcorner \underline{t} \urcorner)$, Lemma 54 implies $a \notin \nu(\overline{v})$ and thus, by Lemma 14, $a \notin \nu(\ulcorner \underline{v} \upharpoonright X \urcorner), \nu(\ulcorner \underline{v}_{\leq m_i} \upharpoonright X \urcorner)$ for $i = 1, \cdots, j$. Consequently, $\Sigma_i(a) = \Sigma_{i+1}(a)$ for $i = 0, \cdots, j$, either by the fact that $u \in Int(A, B, C)$ or by $\gamma'(u_F \upharpoonright X) \in \sigma \cup \tau$. $\Sigma(a) = \Sigma'(a)$ follows. □

Lemma 54. *Let $u \in Int_F(A, B, C)$ be such that $\gamma'(u \upharpoonright AB) \in P_{A \to B}$ and $\gamma'(u \upharpoonright BC) \in P_{B \to C}$. Then, for any name $a \in \mathsf{O}(u \upharpoonright AC)$, if $a \in \nu(\overline{u})$ then $a \in \nu(\ulcorner \underline{u} \upharpoonright AC \urcorner)$.*

Proof. Identical to the proof of Lemma 15. □

Lemma 55 ([35, Lemma 2.11])**.** *Let X be a strong nominal set. For all $x_1, x_2, y_1, y_2, z_1, z_2 \in X$ and finite $\mathsf{u} \subseteq \mathbb{A}$ such that $\mathsf{u} \subseteq \nu(y_i) \cap \nu(z_i) \subseteq \nu(x_i)$, for $i = 1, 2$, if there exist permutations π_y, π_z satisfying*

$$\pi_y \cdot (x_1, y_1) = (x_2, y_2)\,, \quad \pi_z \cdot (x_1, z_1) = (x_2, z_2)\,, \quad \forall a \in \mathsf{u}.\, \pi_y(a) = \pi_z(a) = a$$

then there exists a permutation π such that $\pi \cdot (x_1, y_1, z_1) = (x_2, y_2, z_2)$ and $\forall a \in \mathsf{u}.\, \pi(a) = a$.

Appendix B. Computational Soundness and Adequacy

Here we discuss a modified variant of Stark's categorical framework (described in Chapter 5 of [33]), needed to deliver Proposition 33. The adjustment is necessary so as to incorporate the constant Ω. Since most of the framework remains the same, we focus only on the details that have to be added or modified.

Recall the computational metalanguage for Reduced ML (without Ω) from [33, Section 5.6]. We add to it a constant $\bot$ and the associated typing and derivation rules:

$$\frac{}{\Gamma \vdash \bot : T\,Unit} \qquad \frac{\Gamma, y : Unit \vdash e : T\,Unit}{\Gamma \vdash let\, y \Leftarrow \bot\, in\, e \;=\; \bot} \qquad \frac{}{\Gamma; [()] = \bot \vdash\; tt = ff} \tag{B.1}$$

The translation into the metalanguage is extended with $[\![\Omega]\!] = \bot$. Moreover, since observational equivalence is now defined in terms of equiconvergence, we replace the (MONO$^+$) rule by a new one.

First, we define a grammar for *store terms* by:

$$t \;::=\; () \mid let\, x \Leftarrow new\, in\, t \mid let\, x \Leftarrow set(n,i)\, in\, t$$

where $i \in \mathbb{Z}$. Using the above we set:

$$(\mathrm{MONO}_\bot)\;\; \frac{\Gamma \vdash t, t' : T\,Unit \qquad \vdash e, e' : T\,Unit}{\Gamma;\; let\, x \Leftarrow t\, in\, e \;=\; let\, x \Leftarrow t'\, in\, e' \;\vdash\; e = e'}$$

where t, t' are store terms. Thus, the new rule stipulates that store terms cannot lead to divergence.

Since our games setting provides a model for the extended metalanguage, $\Gamma \vdash [()] = \bot$ is not provable. This in turn implies that $\Gamma \vdash let\, x \Leftarrow t\, in\, [()] \;=\; let\, x \Leftarrow t'\, in \bot$ is not provable for any store terms t, t'. Hence, using the fact that every Reduced ML term reduces either to a value or such a reduction is prevented by Ω, we can prove that the translation into the metalanguage is adequate (as in [33, Proposition 5.10]).

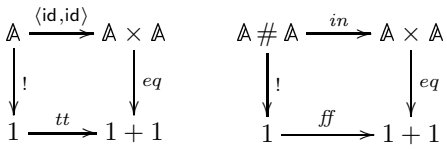
Next we list a complete list of abstract categorical conditions needed to model the extended metalanguage, adapting the setting of [33, Section 5.8]. We require a category $\mathcal{C}$ satisfying the following conditions.

- It is cartesian with product $\times$ and terminal object 1.
- It has a strong monad $T = (T, \eta, \mu, st)$, where all η_A are monic, and T-exponentials $A \Rightarrow TB$. We let ψ, ψ' be the natural transformations:

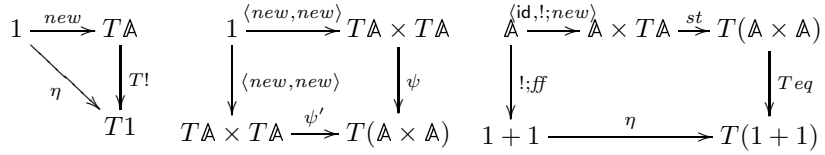
$$\psi_{A,B} :\; TA \times TB \xrightarrow{st'} T(A \times TB) \xrightarrow{T(st)} TT(A \times B) \xrightarrow{\mu} T(A \times B)$$
$$\psi'_{A,B} :\; TA \times TB \xrightarrow{\cong} TB \times TA \xrightarrow{\psi_{B,A}} T(B \times A) \xrightarrow{T\cong} T(A \times B)$$

  where $st'_{A,B} : TA \times B \xrightarrow{\cong} B \times TA \xrightarrow{st_{B,A}} T(B \times A) \xrightarrow{T\cong} T(A \times B)$.
- It has an initial object 0 and disjoint coproduct $1 + 1$ (cf. [33, page 52]).

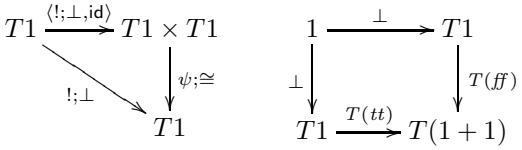
- There is an object $\mathbb{Z}$ for integers and associated morphisms $\oplus : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ (cf. [33, page 116]).
- There is a distinguished object $\mathbb{A}$ and accompanying morphisms $new : 1 \to T\mathbb{A}$ and $eq : \mathbb{A} \times \mathbb{A} \to 1 + 1$ such that:
  - The following diagrams are pullbacks,



    where tt and ff the left and right injections respectively, and $\mathbb{A} \# \mathbb{A}$ and in are defined by the pullback property.
  - The following diagrams commute.

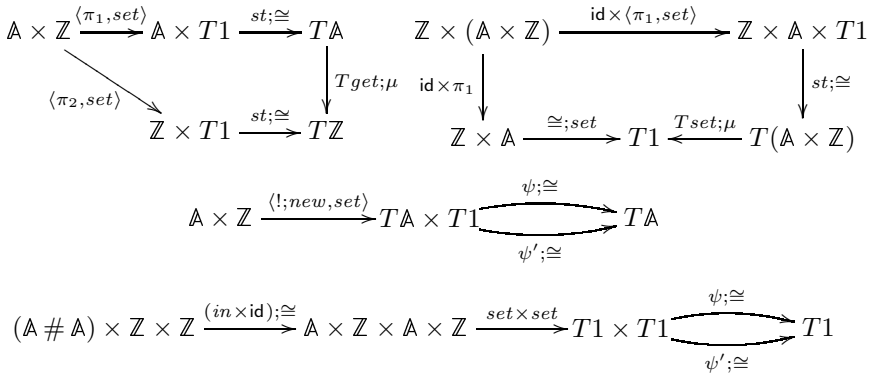


    The above correspond to Stark's (DROP), (SWAP) and (FRESH) rules [33, Figure 3.3].
- There is a distinguished morphism $\bot : 1 \to T1$ making the following diagrams commute.



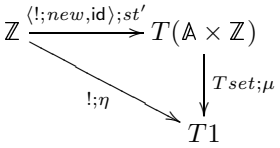
    The diagrams correspond to the inference rules for $\bot$ we introduced in (B.1). The first one ensures that combining two computations where the first is diverging yields divergence, while the second one renders equating termination and divergence inconsistent.
- There are distinguished morphisms $get : \mathbb{A} \to T\mathbb{Z}$ and $set : \mathbb{A} \times \mathbb{Z} \to T1$

such that the following diagrams commute.



The diagrams in the first line correspond to Stark's (READ) and (WRITE) rules respectively from [33, Figure 5.6], while the ones in the second and third line correspond to (SWAP′) and (SWAP″) respectively.

- In order to validate the (MONO$_\perp$) rule, we also require that the diagram below commutes,



(B.2)

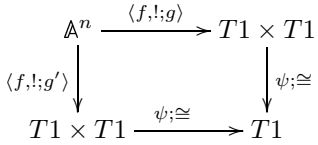
and, additionally, that a particular mono requirement be satisfied. We define the set of *setting morphisms*:

$$\mathit{Sting} = \{\mathbb{A}^n \xrightarrow{\langle \mathsf{id},!;h\rangle;\cong} (\mathbb{A}\times\mathbb{Z})^n \xrightarrow{set^n} (T1)^n \xrightarrow{\psi_n} T1 \mid h : 1 \to \mathbb{Z}^n, n \geq 0\}$$

where $\psi_n : (T1)^n \to T1$ is given inductively by $\psi_0 = \eta$ and:

$$\psi_{i+1} : T1 \times (T1)^i \xrightarrow{\mathsf{id}\times\psi_i} T1 \times T1 \xrightarrow{\psi} T1$$

We stipulate that, for all $f \in \mathit{Sting}$ and all $g, g' : 1 \to T1$, if the following diagram commutes



then $g = g'$.

Following [33, Section 5.8] one can show that any category $\mathcal{C}$ as above is a model for the metalanguage and thus a computationally sound model of Reduced ML. Moreover, when $\mathcal{C}$ is non-degenerate, i.e. $0 \ncong 1$, the model is also computationally adequate.

The rule which we need to additionally demonstrate is (MONO$_\perp$). First, note that using (B.2) and the (DROP), (SWAP), (SWAP$'$), (SWAP$''$) and (WRITE) rules we can derive

$$\frac{\Gamma \vdash t, t' : T\,Unit \qquad \vdash e, e' : T\,Unit}{\Gamma\,;\, let\, x \Leftarrow t\, in\, e \;=\; let\, x \Leftarrow t'\, in\, e' \;\vdash\; let\, x \Leftarrow t_1\, in\, e \;=\; let\, x \Leftarrow t'_1\, in\, e'}$$

for all store terms t, t'; with t_1 obtained from t by dropping all its *new* constructs and the corresponding *set*'s, and similarly for t'_1. Hence, t_1 and t'_1 are mapped into setting morphisms in $\mathcal{C}$ and, by our mono requirement, we obtain $e = e'$.

References

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

[2] S. Abramsky, K. Honda, and G. McCusker. Fully abstract game semantics for general references. In *Proceedings of LICS*, pages 334–344. IEEE Computer Society Press, 1998.

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

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

[5] S. Abramsky and G. McCusker. Linearity, sharing and state: a fully abstract game semantics for Idealized Algol with active expressions. In P. W. O'Hearn and R. D. Tennent, editors, *Algol-like languages*, pages 297–329. Birkhaüser, 1997.

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

[7] R. Harmer. *Games and full abstraction for non-deterministic languages.* PhD thesis, University of London, 1999.

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

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

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

[11] J. Laird. Game Semantics for Higher-Order Concurrency. In *Proceedings of FSTTCS*, volume 4337 of *Lecture Notes in Computer Science*, pages 417–428. Springer-Verlag, 2006.

[12] J. Laird. A game semantics of names and pointers. *Annals of Pure and Applied Logic*, 151:151–169, 2008.

[13] P. B. Levy. Global state considered helpful. In *Proceedings of MFPS*, volume 218 of *Electronic Notes in Theoretical Computer Science*, pages 241–259. Elsevier, 2008.

[14] G. McCusker. *Games for recursive types*. BCS Distinguished Dissertation. Cambridge University Press, 1998. *See also [15].*

[15] G. McCusker. Games and full abstraction for FPC. *Information and Computation*, 160(1-2):1–61, 2000.

[16] G. McCusker. On the semantics of Idealized Algol without the bad-variable constructor. In *Proceedings of MFPS*, volume 83 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2003.

[17] R. Milner, M. Tofte, and R. Harper. The Definition of Standard ML. The MIT Press, 1990.

[18] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[19] A. S. Murawski. Functions with local state: regularity and undecidability. *Theoretical Computer Science*, 338(1/3), 315–349, 2005.

[20] A. S. Murawski. Bad variables under control. In *Proceedings of CSL*, volume 4646 of *Lecture Notes in Computer Science*, pages 558–572. Springer, 2007.

[21] A. S. Murawski and N. Tzevelekos. Full abstraction for Reduced ML. In *Proceedings of FOSSACS*, volume 5504 of *Lecture Notes in Computer Science*, pages 32–47. Springer, 2009.

[22] A. S. Murawski and N. Tzevelekos. Block structure vs. scope extrusion: between innocence and omniscience. In *Proceedings of FOSSACS*, volume 6014 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2010.

[23] A. S. Murawski and N. Tzevelekos. Algorithmic Nominal Game Semantics. In *Proceedings of ESOP*, volume 6602 of *Lecture Notes in Computer Science*, pages 419–438. Springer, 2011.

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

[25] A. S. Murawski and N. Tzevelekos. Algorithmic Games for Full Ground References. In *Proceedings of ICALP*, volume 7392 of *Lecture Notes in Computer Science*, pages 312–324. Springer, 2012.

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

[27] C.-H. Luke Ong. Observational equivalence of 3rd-order Idealized Algol is decidable. In *Proceedings of LICS*, pages 245–256. IEEE Computer Society Press, 2002.

[28] A. M. Pitts and I. Stark. On the observable properties of higher order functions that dynamically create local names, or: What's new? In *Proc. 18th Int. Symp. on Math. Foundations of Computer Science*, pages 122–141. Springer-Verlag, 1993. LNCS Vol. 711.

[29] A. M. Pitts and I. D. B. Stark. Operational reasoning for functions with local state. In A. D. Gordon and A. M. Pitts, editors, *Higher-Order Operational Techniques in Semantics*, pages 227–273. Cambridge University Press, 1998.

[30] G. Plotkin. Domains. University of Edinburgh, 1983.

[31] J. C. Reynolds. The essence of Algol. In J. W. de Bakker and J.C. van Vliet, editors, *Algorithmic Languages*, pages 345–372. North Holland, 1978.

[32] U. Scöpp. Names and binding in type theory. PhD thesis, University of Edinburgh, 2006.

[33] I. D. B. Stark. *Names and higher-order functions*. PhD thesis, University of Cambridge Computing Laboratory, 1995. Technical Report No. 363.

[34] N. Tzevelekos. Full abstraction for nominal general references. *Logical Methods in Computer Science*, 5(3:8), 2009.

[35] N. Tzevelekos. Nominal game semantics. PhD thesis, Oxford University Computing Laboratory, 2008.