Full Abstraction for Reduced ML

Andrzej S. Murawski[⋆] and Nikos Tzevelekos

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD, UK

Abstract. We present the first effectively presentable fully abstract model for Stark's Reduced ML, the paradigmatic higher-order programming language combining call-by-value evaluation and 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. This leads to an explicit characterization of program equivalence.

1 Introduction

Reduced ML is a programming language introduced by Stark [22] as part of his investigations into generative aspects of programming languages. It combines higher-order functions with integer references and is defined simply by extending the call-by-value λ-calculus with primitives for integer and reference manipulation. Despite its economy, Reduced ML manages to embody several important paradigms (imperative programming, functional programming, call-by-value evaluation, scope extrusion), which makes it an attractive object for theoretical study. On the other hand, research into it offers wide scope for applicability, as Reduced ML is intimately related to Standard ML [15] and, in fact, has been designed with faithfulness to the latter in mind.

The first steps in the semantic analysis of Reduced ML were taken by Stark, who has identified a matching categorical framework and considered example categories, albeit without a general full abstraction result[1]. Further progress was possible with the arrival of game semantics [4,9,19]. Although the first papers concerned call-by-name computation, attention soon turned to the call-by-value framework [8,6]. In particular, Abramsky and McCusker presented a fully abstract model for a language called RML [6], 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 [21]. Thus, mkvar allows one to define terms of reference type that need not correspond to actual memory locations. Unfortunately, this affects

⋆ Supported by an EPSRC Advanced Research Fellowship (EP/C539753/1).

1 A denotational model of a programming language is *fully abstract* iff equality of denotations coincides with program equivalence. Programs are *equivalent* iff they can be used interchangeably without observable differences.

the induced notion of program equivalence, so the full abstraction result of [6] does not apply to Reduced ML. More precisely, it can fail at types containing occurrences of $\mathsf{int\,ref}$. Typical counterexamples are the failures of 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 [14,18], 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* [10,2,23]. 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]. $\lambda\nu!$ turns out 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. This can be illustrated by the terms[2]

$$f : \mathsf{int\,ref} \to \mathsf{unit} \vdash \mathsf{let}\, n_1 = \mathsf{ref}\,0\,\mathsf{in}\,\mathsf{let}\, n_2 = \mathsf{ref}\,0\,\mathsf{in}\,((f n_1); (n_2 := !n_1); n_2) : \mathsf{int\,ref}.$$

and $f : \mathsf{int\,ref} \to \mathsf{unit} \vdash \mathsf{let}\, n = \mathsf{ref}\,0\,\mathsf{in}\,(f n); n : \mathsf{int\,ref}$, which are equivalent in Reduced ML, but inequivalent[3] in $\lambda\nu!$. This is because a $\lambda\nu!$-context can detect the difference between n_1 from n_2 by storing the names and subsequently comparing them. In contrast, as our results confirm, the same effect cannot be achieved by a context belonging to Reduced ML.

Previous 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 [3,23]. The first of these models suffers from the "bad-variable" problem outlined above. The second one, while adaptable to Reduced ML, leans rather too heavily 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 is obtained in the present paper.

Our point of departure is the observation that, although a Reduced ML program will in general not be able to keep track of all the names it encounters during the course of interaction with another program, at any given execution point there is a subset of such names that the program may have access to. In game

[2] $\mathsf{let}\, x = M\,\mathsf{in}\, N$ stands for the Reduced ML term $(\lambda x.N)M$.

[3] Strictly speaking, the terms are not in $\lambda\nu!$, but the scenario can be easily recast in $\lambda\nu!$ by replacing $\mathsf{ref}\,0$ with $\nu n.n$.

semantics, using the notion of P-view, we can describe this set conservatively as 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 result. As our example demonstrates, a Reduced ML 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 iff 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 accessible 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 obtain an explicit characterization of equivalence: terms of Reduced ML are equivalent iff they induce the same *mutually strict complete protoplays* (complete plays where O plays only O-available names and in which stores are restricted to *mutually available* names).

For example, *each* of the two terms introduced above generates the following such plays

$$\begin{tikzcd}[column sep=small] * & n_1^{(n_1,0)} \arrow[l] & *^{(n_1,k)} \arrow[l, dash] & n_2^{(n_2,k)} \arrow[lll, bend right] \end{tikzcd},$$

where k ranges over the set of integers. Hence, the terms are indeed equivalent.

Notes. A long version with proofs is available from the authors' webpages. We are grateful to Jim Laird for email discussions.

2 Reduced ML

Reduced ML is the call-by-value λ-calculus over the ground types $\mathsf{unit}, \mathsf{int}, \mathsf{int\,ref}$ augmented with basic commands (termination, divergence), primitives for inte-

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

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

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

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

Fig. 1. Syntax of Reduced ML

ger arithmetic (constants, zero-test, binary integer functions) and reference manipulation (locations, dereferencing, assignment, 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 some extent immaterial: for the full abstraction argument to hold it suffices to be able to compare integer variables with integer constants and act on the result. The same can be said about the lack of recursion, which can be added without affecting our results. Note that we did not include reference equality testing, because it is expressible [20]. For instance, one can define $\mathsf{eq}:\mathsf{int\,ref}\to\mathsf{int\,ref}\to\mathsf{int}$ as

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

In the above and in what follows, we write $M;N$ for the term $(\lambda z^{\theta}.N)M$, where z does not occur in N and θ matches the type of M.

To define the operational semantics of Reduced ML, we need to introduce a notion of store. A *store* 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* iff 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), (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 range over locations. 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

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

$$\frac{V \text{ is canonical}}{s, V \Downarrow s, V} \qquad \frac{M \Downarrow 0 \quad N_0 \Downarrow V}{\mathsf{if}\, M\, \mathsf{then}\, N_1\, \mathsf{else}\, N_0 \Downarrow V} \qquad \frac{i \neq 0 \quad M \Downarrow i \quad N_1 \Downarrow V}{\mathsf{if}\, M\, \mathsf{then}\, N_1\, \mathsf{else}\, N_0 \Downarrow V}$$

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

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

Fig. 2. Big-step operational semantics of Reduced ML

In particular, this means that the ordering of the hypotheses is significant. We shall write $\Gamma \vdash M : \theta$ iff $\emptyset, \Gamma \vdash M : \theta$ 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 $M \Downarrow$ iff $\emptyset, M \Downarrow s, ()$ for some store s.

Definition 1. *We say that the term-in-context $\Gamma \vdash M_1 : \theta$* **approximates** *$\Gamma \vdash M_2 : \theta$ (written $\Gamma \vdash M_1 \precsim M_2$) 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 $\Gamma \vdash M_1 \cong M_2$).*

The only difference between our definition of Reduced ML and Stark's is the presence of Ω, the divergent constant without a reduction rule. Thanks to it, we can define $\precsim$ and reason about $\cong$ in a more concise way. At the same time, program equivalence of Ω-free Reduced ML terms remains unaffected, 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 Nominal Game Semantics

We begin this section with a brief review of the fundamentals of nominal game semantics [2,11,24]. Let us fix a countably infinite set $\mathbb{A}$, the set of *atoms*, the elements of which we denote by a, b, c, n and variants. In nominal game semantics two participants play a game by exchanging moves that might involve atoms. 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 name-renamings. Fortunately, this overhead can be dealt with robustly using the language of nominal set theory [7].

Definition 2. *Let us write* 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* PERM($\mathbb{A}$)[4]. *Moreover, each $x \in X$ must have* finite support*, that is, there exists a finite set $S \subseteq \mathbb{A}$ such that, for all permutations π, $(\forall a \in S.\, \pi(a) = a) \implies \pi \cdot x = x$.*

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 $a \notin \nu(x)$. Clearly, $\mathbb{A}$ is a nominal set by taking $\pi \cdot a = \pi(a)$, for each π and a. More interestingly, so is the set $\mathbb{A}^*$ of finite lists of atoms with permutations acting elementwise. If X and Y are nominal sets then so is their cartesian product $X \times Y$, with permutations acting componentwise, and their disjoint union $X \uplus Y$. Moreover, $X' \subseteq X$ is a *nominal subset* of X if X' is closed under permutation actions, these acting as on X. Then we can define $R \subseteq X \times Y$ to be a *nominal relation* iff R a nominal subset of $X \times Y$. A *nominal function* is a function which is also a nominal relation.

In game semantics a particular strengthening of the notion of support, called *strong support*, has turned out necessary to guarantee correct behaviour under strategy composition (see [24] for motivation and a detailed explanation of its significance). Here we consider an even stronger notion of support, one in which the support of each element can be linearly ordered in a canonical, nominal manner. A nominal set X is called a ***linear nominal set*** if, for each element x of X, there exists a linear order $<_x$ on $\nu(x)$ such that, for all $a, b \in \mathbb{A}$, $a <_x b$ implies $\pi(a) <_{\pi \cdot x} \pi(b)$ for any permutation π [5]. It is easy to check that all elements in a linear nominal set have strong support. For example, the nominal set $\mathbb{A}^*$ is linear, whereas $\mathcal{P}_{fin}(\mathbb{A})$ is not. A nominal subset of a linear nominal set is itself linear. Moreover, by straightforward manipulations of the orderings available for linear X, Y we can render the nominal sets $X \times Y$ and $X \uplus Y$ linear.

Finally, in nominal sets we can *define* atom-abstractions. The form of abstraction we will be using is that of complete support abstraction, that is, for a nominal set X and $x \in X$, we define $[x]$ to be $\{y \in X \mid \exists \pi.\, y = \pi \cdot x\}$.

3.1 Nominal Arenas

Here we present nominal arenas (and prearenas), which are essentially the call-by-value arenas of Honda and Yoshida [8] cast inside the theory of nominal sets.

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

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

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

[5] Equivalently, the relation $\{(a, b, x) \mid a, b \in \mathbb{A},\ x \in X,\ a <_x b\}$ is nominal.

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*. The simplest arena is $0 = (\emptyset, \emptyset, \emptyset, \emptyset)$. Other "flat" arenas are 1, $\mathbb{Z}$ and $\mathbb{A}$, defined by $M_1 = I_1 = \{*\}$, $M_{\mathbb{Z}} = I_{\mathbb{Z}} = \mathbb{Z}$, $M_{\mathbb{A}} = I_{\mathbb{A}} = \mathbb{A}$.

We take advantage 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; i_A and i_B range over initial moves in the respective arenas.

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

$$\begin{aligned}
M_{A\Rightarrow B} &= \{*\} \uplus I_A \uplus \overline{I}_A \uplus M_B \\
I_{A\Rightarrow B} &= \{*\} \\
\lambda_{A\Rightarrow B} &= [(*, PA), (i_A, OQ), \bar{\lambda}_A \upharpoonright \overline{I}_A, \lambda_B] \\
\vdash_{A\Rightarrow B} &= \{(*, i_A), (i_A, i_B)\} \cup \vdash_A \cup \vdash_B
\end{aligned}$$

The types of Reduced ML will be interpreted by arenas in the following way: $[\![\mathsf{unit}]\!] = 1$, $[\![\mathsf{int}]\!] = \mathbb{Z}$, $[\![\mathsf{int\,ref}]\!] = \mathbb{A}$ and $[\![\theta_1 \to \theta_2]\!] = [\![\theta_1]\!] \Rightarrow [\![\theta_2]\!]$. Although types are interpreted by arenas, the actual games will be played in ***prearenas***, which are defined in the same way as arenas with the exception that initial moves are O-questions. For given arenas A, B we can construct a prearena $A \to B$ by

$$\begin{aligned}
M_{A\to B} &= M_A \uplus M_B & \lambda_{A\to B} &= [(i_A, OQ) \cup (\bar{\lambda}_A \upharpoonright \overline{I}_A)\ ,\ \lambda_B] \\
I_{A\to B} &= I_A & \vdash_{A\to B} &= \{(i_A, i_B)\} \cup \vdash_A \cup \vdash_B\ .
\end{aligned}$$

Typing judgements $\Gamma \vdash \theta$, where $\Gamma = \{x_1 : \theta_1, \cdots, x_n : \theta_n\}$, will eventually be interpreted by strategies for the prearena $[\![\theta_1]\!] \otimes \cdots \otimes [\![\theta_n]\!] \to [\![\theta]\!]$ (if $n = 0$ we take the left-hand side to be 1), which we shall denote by $[\![\Gamma \vdash \theta]\!]$.

3.2 Plays

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

A ***basic justified sequence*** in a prearena A is a finite sequence s of moves of A satisfying the following conditions: the first move must be initial, but all other moves m must be equipped with a pointer to an earlier occurrence of another 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^S. 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 linear) 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.

$$\begin{aligned} \llcorner \epsilon \lrcorner &= \epsilon & \ulcorner \epsilon \urcorner &= \epsilon \\ \llcorner s\, o^S \lrcorner &= \llcorner s \lrcorner\, o^S & \ulcorner s\, p^S \urcorner &= \ulcorner s \urcorner\, p^S \\ \llcorner s\, o^S \overset{\frown}{t}\, p^{S'} \lrcorner &= \llcorner s \lrcorner\, o^S p^{S'} & \llcorner s\, p^S \overset{\frown}{t}\, o^{S'} \lrcorner &= \llcorner s \lrcorner\, p^S o^{S'} \end{aligned}$$

A name in s is said to be ***introduced by player*** X ($X \in \{O, P\}$) iff 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 4. *A justified sequence is* legal *iff it satisfies the following conditions.*

Alternation. *No two adjacent moves belong to the same player.*

Bracketing. *The justifier of each answer is the most recent unanswered question.*

Visibility. *For any $tm^S \sqsubseteq s$, the justifier of m is in $\llcorner tm^S \lrcorner$ if m is an O-move and in $\ulcorner tm^S \urcorner$ otherwise.*

Frugality. *For any $tm^S \sqsubseteq s$, $\mathsf{dom}\, S \subseteq \nu(\underline{tm^S})$.*

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

Note that legal sequences contain those of [11]. Because of frugality, the support of a legal sequence is that of its underlying basic sequence, and therefore L_A is a linear nominal set. Our model will be based on still more restrictive plays.

Definition 5. *A legal sequence s is a* **play** *iff it satisfies the following two conditions.*

P-availability. *For each $s'p^S \sqsubseteq^{even} s$ and any $a \in \mathbb{A}$, if $a \in \nu(p) \cap \nu(s')$ then $a \in \mathsf{Av}_{\mathrm{P}}(s')$.*

P-storage. *For any $s'm^S \sqsubseteq s$, $\mathsf{dom}\, S = \mathsf{Av}_{\mathrm{P}}(s'm^S)$.*

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

Note that the two conditions are biased towards P. Equivalently, P-availability can be restated as: for any $s'p^S \sqsubseteq^{\text{even}} s$ and any $a \in \nu(p)$, 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 $\mathsf{Av}_{\mathrm{P}}(s) = \nu(\ulcorner s \urcorner)$.

3.3 Strategies

Definition 6. *A* **strategy** *σ on a prearena A, written $\sigma : A$, is a set of equivalence classes $[s]$ of even-length* plays *of A satisfying*

Even-prefix closure. *If $[so^S p^{S'}] \in \sigma$ then $[s] \in \sigma$,*
Determinacy. *If $[sp_1^{S_1}], [s'p_2^{S_2}] \in \sigma$ and $[s] = [s']$ then $[sp_1^{S_1}] = [s'p_2^{S_2}]$.*

Next we show how strategies can be composed. First, following [11], let us define an endofunction γ on justified sequences that restricts a given justified sequence to a frugal one by removing from the stores the atoms violating it. Now, let γ' be an analogous *partial* function enforcing P-storage, i.e. γ' will remove O-names violating P-storage (it is undefined when the domain of any of the stores involved contains an insufficient supply of atoms, i.e. some of the P-available names required are missing).

Now we turn to defining a suitable notion of interaction between plays. Given arenas A, B, C, let u be a sequence $m_1^{S_1} \cdots m_k^{S_k}$ of moves from $M_A + M_B + M_C$ with store, equipped with pointers such that no C-move has a pointer to an A-move and vice versa. We define $u \upharpoonright A, B$ to be u in which all C-moves are suppressed along with associated pointers. $u \upharpoonright B, C$ is defined analogously. $u \upharpoonright A, C$ 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. By $u_{\leq m_i}$ we mean the initial segment of u ending in $m_i^{S_i}$.

Definition 7. *u is an interaction sequence of A, B, C iff $\gamma'(u \upharpoonright A, B) \in P_{A \to B}$, $\gamma'(u \upharpoonright B, C) \in P_{B \to C}$ and the following conditions hold:*

- $\mathsf{P}(u \upharpoonright A, B) \cap \mathsf{P}(u \upharpoonright B, C) = \emptyset$;
- $\mathsf{O}(u \upharpoonright A, C) \cap (\mathsf{P}(u \upharpoonright A, B) \cup \mathsf{P}(u \upharpoonright B, C)) = \emptyset$;
- *for each $u' \sqsubseteq u$ ending in a move-with-store m^S,*
  $\mathsf{dom}\, S = (\mathsf{O}(u' \upharpoonright A, C) \cap \nu(\ulcorner \underline{u'} \upharpoonright A, C \urcorner)) \cup \mathsf{P}(u' \upharpoonright A, B) \cup \mathsf{P}(u' \upharpoonright B, C)$;
- *for each $u' \sqsubseteq u$ ending in $m^S m'^{S'}$, if m' is:*
  - *a P-move in $A \to B$ then $S'(a) = S(a)$ for all $a \in \mathsf{dom}\, S' \setminus \mathsf{Av}_\mathrm{P}(u' \upharpoonright A, B)$;*
  - *a P-move in $B \to C$ then $S'(a) = S(a)$ for all $a \in \mathsf{dom}\, S' \setminus \mathsf{Av}_\mathrm{P}(u' \upharpoonright B, C)$;*
  - *an O-move in $A \to C$ then $S'(a) = S(a)$ for all $a \in \mathsf{dom}\, S' \setminus \mathsf{Av}_\mathrm{P}(u' \upharpoonright A, C)$.*

The set of all interaction sequences of A, B, C will be denoted by $Int(A, B, C)$.

The first two conditions ensure that name-privacy is not broken under composition; the third one imposes an extended notion of P-availability for sequences; and the fourth set of conditions ensures that participants do not change parts of the store inaccessible to them. It can be shown that, if $u \in Int(A, B, C)$ then $\gamma(u \upharpoonright A, C) \in P_{A \to C}$. Two strategies $\sigma : A \to B$ and $\tau : B \to C$ can now be composed as follows

$$\sigma;\tau = \{[\gamma(u \upharpoonright A, C)] \mid u \in Int(A, B, C),\ [\gamma'(u \upharpoonright A, B)] \in \sigma,\ [\gamma'(u \upharpoonright B, C)] \in \tau\}.$$

Associativity of composition can be established using similar arguments to those in [11][6]. Using the standard definition of the identity strategy one can then obtain a category of arenas and games. Next we shall define its lluf subcategory that will be used to prove the full abstraction result.

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 by renamings of names that O introduces in the P-view.

Definition 8

- *Given a prearena A, $s \in P_A$, $a \in \mathsf{O}(s)$ and an O-move o in s, we say that a is* **P-new at** o **in** s *iff $a \in \nu(o)$ and $a \notin \nu(\ulcorner s_{\leq o} \urcorner^-)$.*
- *Given A, s, a, o as above and $b \in \mathbb{A}$, we say that a is* **renameable for** b **at** o **in** s *provided $b \notin \mathsf{P}(s)$ and, for any $s' \sqsubseteq s$, 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*[7]*:*

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

where $(a\ b)$ is the permutation swapping a with b. We write $s \overset{r}{\sim} s'$ iff 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 $a \in \nu(m)$). 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***.

Example 9. Let $A = \mathbb{A} \rightarrow (\mathbb{A} \Rightarrow 1)$,

$$s_1 = n_1^{(n_1,0)}\ {*}^{(n_1,1)}\ n_2^{(n_1,2),(n_2,3)}\ {*}^{(n_1,4),(n_2,5)}\ n_3^{(n_1,6),(n_3,7)}\ {*}^{(n_1,8),(n_3,9)}$$

and

$$s_2 = n_1^{(n_1,0)}\ {*}^{(n_1,1)}\ n_3^{(n_1,2),(n_3,3)}\ {*}^{(n_1,4),(n_3,5)}\ n_3^{(n_1,6),(n_3,7)}\ {*}^{(n_1,8),(n_3,9)}\,.$$

Then n_2 is P-new at the third move (also n_2) in s_1, n_2 is renameable for n_3 at that move and $(n_2\ n_3)_{n_2} \cdot s_1 = s_2$. Note also that $(n_3\ n_2)_{n_3} \cdot s_2 = s_1$, where the subscript n_3 stands for the third move of s_2, and that s_1 is strict, whereas s_2 is not.

In general it can be shown that renamings are reversible, so $\overset{r}{\sim}$ is an equivalence relation. Observe that, for any play s, there exists a *strict* play s' (determined uniquely up to atom-abstraction) such that $s \overset{r}{\sim} s'$. We write $\widetilde{s}$ for $[s']$.

Definition 10. *A strategy $\sigma : A$ is* **blind** *iff $[s] \in \sigma$ and $s \overset{r}{\sim} s'$ imply $[s'] \in \sigma$.*

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 positions (via renamings).

[6] In fact, strategies in our framework can be regarded as compact representations of a class of strategies from [11], namely those independent of P-unavailable names.

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

4 Properties of $\mathcal{G}$

Henceforth, when writing $\sigma : A$ we shall mean a blind strategy on A. Following [2] and [11], $\mathcal{G}$ can be shown equivalent to the Klesli category of another category $\mathcal{G}'$ equipped with a strong monad T. More precisely, $\mathcal{G}'$ is a lluf subcategory of $\mathcal{G}$ consisting of total single-threaded strategies [11] such that store values introduced in the first move cannot be modified in the next two moves. The strong monad T takes an arena A to $A_\perp$ given by

$$\begin{aligned} M_{A_\perp} &= \{*_1, *_2\} + M_A & I_{A_\perp} &= \{*_1\} \\ \lambda_{A_\perp} &= [\{(*_1, PA), (*_2, OQ)\}, \lambda_A] & \vdash_{A_\perp} &= \{(*_1, *_2), (*_2, i_A)\} \cup \vdash_A \ . \end{aligned}$$

Moreover, one can show that $\mathcal{G}'$ has finite products and T-exponentials, i.e., for any arena A, there is a natural bijection between $\mathcal{G}'(A \otimes B, TC)$ and $\mathcal{G}'(A, B \Rightarrow C)$. That is to say, $\mathcal{G}'$ is λ_c-model [16], which gives a canonical interpretation of the call-by-value λ-calculus in the associated Kleisli category, i.e., equivalently, the category $\mathcal{G}$. 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 [22]. We list those related to reference manipulation below (as morphisms in $\mathcal{G}$ rather than in $\mathcal{G}'_T$).

$$\mathit{get} = \{[\epsilon], [n^{(n,i)} i^{(n,i)}]\} : \mathbb{A} \to \mathbb{Z} \qquad \mathit{set} = \{[\epsilon], [(n,i)^{(n,i')} *^{(n,i)}]\} : \mathbb{A} \otimes \mathbb{Z} \to 1$$

Memory allocation is interpreted using the strategies $\mathit{new}_i = \{[\epsilon], [* n^{(n,i)}]\} : 1 \to \mathbb{A}$. As a consequence, we conclude that $\mathcal{G}$ is a model of Reduced ML in the sense of Stark[8]. This lets us interpret any term-in-context $\Gamma \vdash M : \theta$ with a strategy $[\![\Gamma \vdash M : \theta]\!] : [\![\Gamma \vdash \theta]\!]$.

Example 11. The two terms from the introduction are interpreted (in $\mathcal{G}$) by the strategies given respectively (through even-prefix closure) by the plays below.

$$\begin{tikzcd} * & n_1^{(n_1,0)} \arrow[l] \arrow[r, dash] & *^{(n_1,k)} & n_2^{(n_1,k),(n_2,k)} \arrow[lll, bend right] & & * & n^{(n,0)} \arrow[l] \arrow[r, dash] & *^{(n,k)} & n^{(n,k)} \arrow[lll, bend right] \end{tikzcd}$$

Conformance to Stark's framework guarantees Computational Soundness and Adequacy [22].

Definition 12

- *$\sigma : A$ is* **finitary** *iff it is finite.*
- *$\sigma : A$ is* **strongly deterministic** *iff, for any $s \in P_A$ such that $[s] \in \sigma$, we have $\mathsf{P}(s) = \emptyset$.*
- *A strategy $\sigma : A$ is* **innocent** *iff, $[sp], [t] \in \sigma$, $to \in P_A$, $\ulcorner s \urcorner = \ulcorner to \urcorner$ implies the existence of $[top'] \in \sigma$ such that $[\ulcorner sp \urcorner] = [\ulcorner top' \urcorner]$ (note that innocence implies blindness). An innocent strategy $\sigma : A$ is* finitarily innocent *iff $\mathsf{vf}(\sigma) = \{[\ulcorner s \urcorner] \mid s \in P_A, [s] \in \sigma\} : A$ is finite.*

[8] Strictly speaking, the cartesian closure requirement from [22] is not satisfied, but it turns out too strong: T-exponentials suffice for the author's subsequent results [2].

Using two factorizations we can show that any finitary blind strategy in a denotable[9] prearena is definable. The first one eliminates violations of strong determinism with the help of new_0 (corresponding to $\mathsf{ref}\,0$). The second one factors out non-innocence (also using new_0). Finally, we prove a direct definability result for finitarily innocent strongly deterministic strategies. We discuss the innocent factorization in more detail below, as it involves the key novelties of our framework.

Lemma 13. *Let $\sigma : 1 \to A$ be a finitary strongly deterministic blind strategy. There exists a finitarily innocent strongly deterministic strategy $\dot{\sigma} : \mathbb{A} \to A$ such that $new_0; \dot{\sigma} = \sigma$.*

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 store a numerical representation of $[s]$ instead. Recall that the set of moves of an arena is a linear nominal set, i.e. there is a *canonical* ordering of atoms in any move. Hence, in any legal sequence, atoms can also be ordered in a *canonical* way according to their order of appearance and, if they were introduced in the same move, using the canonical ordering associated with that move. Consequently, we can represent $[s]$ as an integer by representing atoms in s with integers that correspond to their position in the associated canonical order and by encoding such a sequence as an integer. 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 $\ulcorner so^S \urcorner$ and $\#(s)$, it will in general be impossible to extract so^S (or $[so^S]$) due to the fact that o might contain O-names occurring in s, but not in $\ulcorner so^S \urcorner$. As a result, given $\#(s)$ and $\ulcorner so^S \urcorner$, we may then be unable to relate some names in o with those of s, which will prevent us from reconstructing $[so^S]$. Note, however, that given $\#(s)$ and $\ulcorner so^S \urcorner$ we can still determine $\widetilde{so^S}$ in absence of P-names. Furthermore, since σ is blind and strongly deterministic (in particular plays satisfy P-availability), we can uniquely identify $p^{S'}$ such that $so^S p^{S'} \in \sigma$ (though not necessarily the whole of $so^S p^{S'}$), by referring[10] to $\widetilde{so^S}$ and σ. Analogously, we can also deduce $\widetilde{so^S p^{S'}}$. 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').

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

Definition 14. *Suppose $\sigma_1, \sigma_2 : 1 \to A$. We define $\sigma_1 \le \sigma_2$ to hold iff, for any $\rho : A \to 1$, $\sigma_1; \rho \neq \{[\epsilon]\}$ implies $\sigma_2; \rho \neq \{[\epsilon]\}$.*

[9] $1 \to [\![\theta]\!]$, where θ is a Reduced ML type.

[10] Recall that blind strategies are generated by their strict plays.

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

Theorem 15. *Given Reduced ML terms $\vdash M_1 : \theta$, $\vdash M_2 : \theta$, we have $\vdash M_1 \mathrel{\underset{\sim}{\sqsubset}} M_2$ iff $[\![\vdash M_1]\!] \leq [\![\vdash M_2]\!]$.*

5 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 of ***O-availability***: for each $s'o^S \sqsubseteq^{\text{odd}} s$ and any $a \in \mathbb{A}$, if $a \in \nu(o) \cap \nu(s')$ then $a \in \mathsf{Av}_{\mathsf{O}}(s')$.

Similarly, whenever the play s engages with ρ successfully, the ***O-passivity*** condition holds: for each $s'p^S o^{S'} \sqsubseteq s$ and any $a \in \mathbb{A}$, if $a \in \mathsf{P}(s'p^S) \setminus \mathsf{Av}_{\mathsf{O}}(s'p^S)$ then $S(a) = S'(a)$. 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 16. *A play is* **relevant** *iff 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. We call such sequences ***protoplays*** and 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 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.

Finally, 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 one wants 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 8, though it needs to be defined on protoplays to be correct. For two protoplays s, s', we write $s \underset{r}{\sim} s'$ iff s can be obtained from s' by a series of dual renamings. A protoplay is ***dually strict*** iff any P-name fresh in the O-view is fresh at the point of introduction. Given $\sigma : A$, let $\widehat{\sigma}$ be the following set of equivalence classes of protoplays $\widehat{\sigma} = \{[s] \,|\, s \text{ is dually strict}, s' \in \mathsf{rel}(\sigma), s \underset{r}{\sim} \gamma''(s')\}$. We can then show the following result.

Lemma 17. *Given $\sigma_1, \sigma_2 : 1 \to A$, $\sigma_1 \le \sigma_2$ iff $\widehat{\sigma_1} \subseteq \widehat{\sigma_2}$.*

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*** iff 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 atom-abstraction. Given $\sigma : A$, let $\widehat{\widehat{\sigma}}$ be $\{[s] \mid s \text{ is mutually strict}, s' \in \mathsf{rel}(\sigma), s\,(\overset{r}{\sim};\underset{r}{\sim})\,\gamma''(s')\}$.

Theorem 18. *Given $\sigma_1, \sigma_2 : 1 \to A$, $\sigma_1 \le \sigma_2$ iff $\widehat{\widehat{\sigma_1}} \subseteq \widehat{\widehat{\sigma_2}}$.*

It follows that terms of Reduced ML are equivalent iff they induce the same mutually strict protoplays.

Thus we have shown that program equivalence and approximation in Reduced ML can be captured explicitly, which is the first result of this kind for Reduced ML. The characterization immediately implies that equivalence is decidable for finitary strategies and that the fully abstract model of Reduced ML is effectively presentable.

Our results identify mutually strict protoplays as an appealing object for future study and, indeed, our fully abstract model can be presented in a more direct way by founding the games on them. It has to be said, though, that composition of such plays is quite intricate, because they cannot be combined by parallel composition with hiding: although this kind of interaction is sufficient to test plays, composition in general lacks the convenient duality between O-names and P-names. Consequently, in order to compose mutually strict protoplays, one has to allow for renamings before synchronization and follow with dual renamings afterwards. We intend to present an account of this procedure in the full version of the paper.

In this submission however we have chosen to present the model gradually: starting from the intuitive framework in which full information about the store is available we successively imposed a series of restrictions. We believe this leads to a more informative presentation and decomposes the difficulties involved in dealing with mutually strict protoplays into smaller arguments. For instance, the correctness proof of compositionality of mutually strict protoplays (for the algorithm sketched above) draws on insights obtained from all of our compositionality proofs (P-availability, P-storage, blindness) as well as the argument behind the explicit characterization.

We hope to bring our results to bear on the on-going research into algorithmic aspects of game models [1] and to contribute new methods of reasoning about program equivalence in Reduced ML. This direction has been pursued using logical relations in [20]. However, there are limits to what can be achieved, as program equivalence of finitary Reduced ML (finite types) is already undecidable at second order, due to a similar result for Reduced ML with mkvar in [17].

References

1. Abramsky, S., Ghica, D.R., Murawski, A.S., Ong, C.-H.L.: Applying game semantics to compositional software modeling and verification. In: Jensen, K., Podelski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 421–435. Springer, Heidelberg (2004)
2. Abramsky, S., Ghica, D.R., Murawski, A.S., Ong, C.-H.L., Stark, I.D.B.: Nominal games and full abstraction for the nu-calculus. In: Proc. of LICS, pp. 150–159 (2004)
3. Abramsky, S., Honda, K., McCusker, G.: Fully abstract game semantics for general references. In: Proc. of LICS, pp. 334–344 (1998)
4. Abramsky, S., Jagadeesan, R., Malacaria, P.: Full abstraction for PCF. Information and Computation 163, 409–470 (2000)
5. Abramsky, S., McCusker, G.: Linearity, sharing and state: a fully abstract game semantics for Idealized Algol with active expressions. In: Algol-like languages, Birkhaüser, pp. 297–329 (1997)
6. Abramsky, S., McCusker, G.: Call-by-value games. In: Nielsen, M. (ed.) CSL 1997. LNCS, vol. 1414, pp. 1–17. Springer, Heidelberg (1998)
7. Gabbay, M.J., Pitts, A.M.: A new approach to abstract syntax with variable binding. Formal Aspects of Computing 13, 341–363 (2002)
8. Honda, K., Yoshida, N.: Game-theoretic analysis of call-by-value computation. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.) ICALP 1997. LNCS, vol. 1256, pp. 225–236. Springer, Heidelberg (1997)
9. Hyland, J.M.E., Ong, C.-H.L.: On Full Abstraction for PCF. Information and Computation 163(2), 285–408 (2000)
10. Laird, J.: A game semantics of local names and good variables. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 289–303. Springer, Heidelberg (2004)
11. Laird, J.: A game semantics of names and pointers. Annals of Pure and Applied Logic 151, 151–169 (2008)
12. McCusker, G.: Games for recursive types. BCS Distinguished Dissertation. Cambridge University Press, Cambridge (1998)
13. McCusker, G.: Games and full abstraction for FPC. Information and Computation 160(1-2), 1–61 (2000)
14. McCusker, G.: On the semantics of Idealized Algol without the bad-variable constructor. In: Proc. of MFPS. ENTCS, vol. 83 (2003)
15. Milner, R., Tofte, M., Harper, R.: The Definition of Standard ML. MIT Press, Cambridge (1990)
16. Moggi, E.: Notions of computation and monads. Information and Computation 93, 55–92 (1991)
17. Murawski, A.S.: Functions with local state: regularity and undecidability. Theoretical Computer Science 338(1/3), 315–349 (2005)

18. Murawski, A.S.: Bad variables under control. In: Duparc, J., Henzinger, T.A. (eds.) CSL 2007. LNCS, vol. 4646, pp. 558–572. Springer, Heidelberg (2007)
19. Nickau, H.: Hereditarily sequential functionals. In: Matiyasevich, Y.V., Nerode, A. (eds.) LFCS 1994. LNCS, vol. 813, pp. 253–264. Springer, Heidelberg (1994)
20. Pitts, A.M., Stark, I.D.B.: Operational reasoning for functions with local state. In: Higher-Order Operational Techniques in Semantics, pp. 227–273. CUP (1998)
21. Reynolds, J.C.: The essence of Algol. In: de Bakker, J.W., van Vliet, J. (eds.) Algorithmic Languages, pp. 345–372. North Holland, Amsterdam (1978)
22. Stark, I.D.B.: Names and Higher-Order Functions. PhD thesis, University of Cambridge (1995)
23. Tzevelekos, N.: Full abstraction for nominal general references. In: Proc. of LICS, pp. 399–410 (2007)
24. Tzevelekos, N.: Nominal game semantics. D.Phil. thesis, University of Oxford (2008)