Game semantics for good general references

Andrzej S. Murawski
University of Leicester, UK

Nikos Tzevelekos
University of Oxford, UK

***Abstract*—We present a new fully abstract and effectively presentable denotational model for RefML, a paradigmatic higher-order programming language combining call-by-value evaluation and general references in the style of ML. Our model is built using game semantics. In contrast to the previous model by Abramsky, Honda and McCusker [3], it provides a faithful account of reference types, and the full abstraction result does not rely on the availability of spurious constructs of reference type (bad variables). This is the first denotational model of this kind, preceded only by the trace model recently proposed by Laird [16].**

I. Introduction

Nearly all modern programming languages implement the concept of a mutable reference in some form, with varying constraints as to the kind of values permitted in the store. The storing of ground-type values such as integers is the most simple embodiment of the idea, underpinning basic imperative programming. However, languages such as C, Java or ML offer much more flexibility with regard to storage through the availability of function pointers, object references and general references respectively. In fact, in the last two cases values of any type are storable. In this paper we focus on ML-style general references, which can support the storage of ground values, functions as well as references themselves. Even without polymorphism or recursive types, the paradigm is remarkably expressive: thanks to inherent sharing and the ability to create cyclic structures in the store references can account for recursion and many high-level programming abstractions such as objects [8] and aspects [27]. In recent years they have been an extremely popular object of research involving a panoply of techniques: environmental bisimulations [14, 26], game semantics [3, 30, 18], Hoare-style logics [31, 28], logical relations [7, 5, 10], possible-world semantics [20], realizability [6], and traces [16].

In this paper we advance the state of the art in fully abstract modelling of general references with game semantics. The first game model of a language involving general references was constructed by Abramsky, Honda and McCusker [3]. The model interprets references according to Reynolds' recipe [25]: the reference type $\mathsf{ref}\ \theta$ (for storing values of type θ) is viewed as a product of the reading $(1 \to \theta)$ and writing $(\theta \to 1)$ types respectively. The associated full abstraction[1] argument for reference types requires one to populate the product space corresponding to references with non-native reference objects. These so-called "bad references" are arbitrary pairs consisting of (possibly unrelated) reading and writing methods. As a result, the model fails to validate many fundamental equivalences associated with storage such as the ones below[2]:

- $x : \mathsf{ref}\ \theta \vdash x := !x \cong () : \mathsf{unit}$
- $x : \mathsf{ref}\ \theta \vdash (x := V; x := W) \cong x := W : \mathsf{unit}$
- $x : \mathsf{ref}\ \theta \vdash (x := V; !x) \cong (x := V; V) : \theta$

because the interpretation does not relate reads and writes before the reference is bound to a reference cell. Moreover, the process of reading or writing values is not viewed as atomic and treated as a compound operation that might produce side-effects potentially affecting all other memory locations.

In recent years nominal game semantics [2, 17, 30] has emerged as an alternative approach to modelling storage in game semantics, which can overcome the shortcomings caused by bad variables. Reference types are no longer treated as syntactic sugar for certain product types, but interpreted through a countable set of *names*, intuitively corresponding to names of reference cells. Several fully abstract and effectively presentable models for languages involving storage have been obtained by following this route, without relying on the availability of bad variables.

- Laird [17] showed how to capture an extension of the ν-calculus [24] with storage for (untyped) names. A reference cell may contain itself in this setting.
- We showed how to account for ground-type storage in [23], thus mending the original model of Reduced ML due to Abramsky and McCusker [4].

Both of the above results consist in superimposing store information on the standard notion of play. In the former case the full store is included with each move, whereas in the latter the store needs to be carefully restricted. It should be noted that general references do *not* extend either of the above languages conservatively.

In this paper we refine the model of general references presented in [3]. Our plays do not feature information about the full content of the store, as this would reveal functional values and jeopardize full abstraction. Instead, interactions with the functional store, which partially reveal the properties of stored values, are weaved into traditional play. To that end we introduce a novel notion of play in which justification pointers from moves need not point at other moves but, alternatively, they might point inside the stores that other moves carry, more precisely to their functional parts. Another new technical

Supported by an EPSRC Advanced Research Fellowship (EP/C539753/1).
Supported by EPSRC (EP/F067607/1).

[1] A model is (equationally) *fully abstract* if equality of interpretations coincides with the notion of program equivalence.

[2] V, W are assumed to be values.

ingredient in our work is the notion of composition. Here the main challenge is to identify conditions ensuring that higher-order values not accessible to one of the strategies will not be covertly modified during composition.

On the structural level, our proof of full abstraction follows the well-established pattern of proving such results. Soundness (Section V) is obtained by showing conformance with a categorical framework [30], already known to guarantee soundness. Completeness (Section VI) follows from a definability result, which is interesting in its own right, as the new structure of plays enables one to perform rather unexpected transformations on plays to reduce the problem to simpler and smaller instances. Altogether we obtain a model in which program approximation (contextual preorder) corresponds to inclusion of the induced complete[3] plays. This immediately implies effective presentability, i.e. a decidable presentation of the compact elements of the model.

Related and future work. As already described, our model rectifies problems present in a previous game model due to Abramsky, Honda and McCusker [3]. The structure of their model was subsequently studied by Levy [21] and Melliès [22] with the aim of understanding its structure in more abstract terms. Otherwise the most closely related work is Laird's fully abstract trace semantics of essentially the same language [16]. Our model can be viewed as a game-semantic counterpart of his work: traces are derived from terms through an operational semantics, whereas our strategies are defined in a compositional and syntax-free manner. This illustrates a recent convergence of complementary results in the two fields (cf. [19] and [18]) that promises to lead, in the long run, to an operational account of game semantics, which will ultimately make it possible to move smoothly between (syntax-directed, non-compositional) labelled transition system semantics and (syntax-independent, compositional) game semantics. Laird [15] has also presented a fully abstract model for a fragment of Concurrent ML with higher-order communication channels. Although it does not handle higher-order references, it does embody a notion of play where typed channel names may justify plays of the corresponding types.

A different fully abstract game model for the language considered in this paper has already been presented by one of us [30]. Grounded in monadic semantics for store, it did not however offer an explicit characterization of program equivalence due to reliance on innocent strategies (which had to be quotiented for full abstraction). The present work can thus also be seen as a refinement of that work towards a model that captures the behaviour of the environment more faithfully.

In the wide spectrum of methodologies for references our work offers a new foundation for compositional analysis of general references. Modular verification of programs with general references is a topical problem, which was already attacked through a variety of approaches, e.g. separation logic [28]. In future, we hope to apply our model to model-checking and control-flow analysis in the spirit of algorithmic game semantics [12, 1]. Although higher-order references are an expressive paradigm, quickly resulting in undecidability, decidable properties can sometimes be identified [9] and we will be in a good position to approach such from a new perspective. We would also like to make an impact on the automated or machine-checkable verification of program equivalences and understand the relationship between game semantics and other methods used to the same end, such as step-indexing [5] and bisimulation-based techniques [26]. On the semantic front, as a next step, we would like to extend our work to polymorphism, so as to eliminate the bad-variable problem in [18].

[3]A play is *complete* if any question occurring in it has been answered.

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

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

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

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

$$\frac{u,\Gamma\vdash M:\theta}{u,\Gamma\vdash\mathsf{ref}_\theta(M):\mathsf{ref}\,\theta}\qquad\frac{u,\Gamma\vdash M:\mathsf{ref}\,\theta\quad u,\Gamma\vdash N:\mathsf{ref}\,\theta}{u,\Gamma\vdash M=N:\mathsf{int}}$$

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

Fig. 1. Syntax of RefML.

II. The language RefML

We shall work with types defined by the grammar below.

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

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

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

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

Notes: $i \neq 0, a \neq b, a' \notin \mathsf{dom}(S)$.

Fig. 2. Small-step operational semantics of RefML.

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

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

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

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

Example 1: Let circ be the circular reference defined by:

$$\vdash \mathsf{let}\, x = \mathsf{ref}_{\mathsf{unit}\to\mathsf{unit}}(\lambda y^{\mathsf{unit}}.y)\ \mathsf{in}\\ x := (\lambda y^{\mathsf{unit}}.(!x)y);\ x : \mathsf{ref}\,(\mathsf{unit} \to \mathsf{unit})$$

We shall write Ω_{unit} for the divergent term $(!\mathsf{circ})\,()$. Using Ω_{unit} it is easy to define analogous divergent terms Ω_θ at any type. Also, for any type θ, we define the terms new_θ by:

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

where θ'' is one of $\mathsf{unit}, \mathsf{int}$ or a function type. These terms create new names and initialise them with default values. We shall write $\mathsf{new}\, x\, \mathsf{in}\, M$ for $\mathsf{let}\, x = \mathsf{new}_\theta\, \mathsf{in}\, M$.

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

Example 3:

- Let Inc be the term

$$\vdash \mathsf{let}\, n = \mathsf{ref}_{\mathsf{int}}(0)\ \mathsf{in}\ \lambda x^{\mathsf{unit}}.n := !n + 1; !n : \mathsf{unit} \to \mathsf{int}$$

that reports the number of times the function has been invoked. Let $\Gamma = \{x : \mathsf{ref}\,(\mathsf{unit} \to \mathsf{int}), y : \mathsf{ref}\,(\mathsf{unit} \to \mathsf{int})\}$. The first two terms listed below turn out to be equivalent, but the third one can be distinguished from the two, because 2 is returned.

$$\Gamma \vdash x := \mathsf{Inc}; y := !x; ((!x)() + (!y)()) : \mathsf{int}$$
$$\Gamma \vdash x := \mathsf{Inc}; y := !x; 3 : \mathsf{int}$$
$$\Gamma \vdash x := \mathsf{Inc}; y := \mathsf{Inc}; ((!x)() + (!y)()) : \mathsf{int}$$

The equivalence is borrowed from [31] and, like the equivalences listed in the introduction, cannot be confirmed in the game model of [3] due to the bad variable problem.

- RefML is not a conservative extension of Reduced ML [29], in which the only reference type available is ref int. Let $\Gamma = \{f : \mathsf{unit} \to \mathsf{unit}\}$. The terms

$$\Gamma \vdash \mathsf{let}\, n = \mathsf{ref}_{\mathsf{int}}(0)\ \mathsf{in}\ \lambda y^{\mathsf{unit}}.\mathsf{if}\ !n\ \mathsf{then}\ ()\ \mathsf{else}\ (n := 1; f())$$
$$\Gamma \vdash \mathsf{let}\, n = \mathsf{ref}_{\mathsf{int}}(0)\ \mathsf{in}\ \lambda y^{\mathsf{unit}}.\mathsf{if}\ !n\ \mathsf{then}\ ()\ \mathsf{else}\ (f(); n := 1)$$

are equivalent in Reduced ML[4], but inequivalent when tested with RefML contexts. For instance, take $C[-]$ to be the context below.

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

III. Game semantics

Our game model will be constructed using mathematical objects (moves, plays, strategies) that feature names drawn from the set $\mathbb{A} = \biguplus_\theta \mathbb{A}_\theta$. We set $\mathbb{A}_\phi = \biguplus_{\theta,\theta'} \mathbb{A}_{\theta\to\theta'}$, these are the names of functional type. Although names underpin various elements of our model, we do not want to delve into the precise nature of the sets containing them. Hence, all of our definitions preserve name-invariance, i.e. our objects are (strong) *nominal sets* [11, 30]. Note that we do not need the full power of the theory but mainly the basic notion of name-permutation. Here permutations are bijections $\pi : \mathbb{A} \to \mathbb{A}$ with finite support which respect the indexing of name-sets. For an element x belonging to a (nominal) set X we write $\nu(x)$ for its name-support, which is the set of names occurring in x. Moreover, for any $x, y \in X$, we write $x \sim y$ if there is a permutation π such that $x = \pi \cdot y$. Our model is couched in the Honda-Yoshida style of modelling call-by-value computation [13]. Before we define what it means to play our games, we introduce the auxiliary concept of an arena.

Definition 4: An *arena* $A = \langle M_A, I_A, \lambda_A, \vdash_A \rangle$ is given by:

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

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

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

[4]This can be established by mapping them to the corresponding strategies in [4] or [23].

We range over moves by m, n and use $\mathrm{i}, \mathrm{q}, \mathrm{a}$ to refer to initial moves, question-moves and answer-moves respectively. We also use o and p to stress ownership of moves. Let $\overline{\lambda}_A$ be the OP-complement of λ_A. Note that if $\mathrm{i} \vdash_A m$ then $\lambda_A(m) = (O, Q)$. We call such moves m the *initial questions* of the arena A. Given arenas A, B, the arenas $A \otimes B$ and $A \Rightarrow B$ are constructed as follows, where $\bar{I}_A = M_A \backslash I_A$, $\bar{\vdash}_A = (\vdash_A \restriction \bar{I}_A{}^2)$ (and similarly for B).

$$
\begin{aligned}
M_{A\otimes B} &= (I_A \times I_B) \uplus \bar{I}_A \uplus \bar{I}_B \quad I_{A\otimes B} = I_A \times I_B \\
\lambda_{A\otimes B} &= [(\mathrm{i}_A, \mathrm{i}_B) \mapsto PA,\ \lambda_A \restriction \bar{I}_A,\ \lambda_B \restriction \bar{I}_B] \\
\vdash_{A\otimes B} &= \{((\mathrm{i}_A, \mathrm{i}_B), m) \mid \mathrm{i}_A \vdash_A m \vee \mathrm{i}_B \vdash_B m\} \cup \bar{\vdash}_A \cup \bar{\vdash}_B
\end{aligned}
$$

$$
\begin{aligned}
M_{A\Rightarrow B} &= \{\star\} \uplus M_A \uplus M_B \quad I_{A\Rightarrow B} = \{\star\} \\
\lambda_{A\Rightarrow B} &= [\star \mapsto PA,\ \overline{\lambda}_A[\mathrm{i}_A \mapsto OQ],\ \lambda_B] \\
\vdash_{A\Rightarrow B} &= \{(\star, \mathrm{i}_A)\} \cup \{(\mathrm{i}_A, \mathrm{i}_B)\} \cup \vdash_A \cup \vdash_B
\end{aligned}
$$

Now for each type θ we define the corresponding arena $[\![\theta]\!]$.

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

We write 1 for $[\![\mathsf{unit}]\!]$, $\mathbb{Z}$ for $[\![\mathsf{int}]\!]$, and $\mathbb{A}_\theta$ for $[\![\mathsf{ref}\,\theta]\!]$. Moreover, we set $M_\phi = \biguplus_{\theta,\theta'} M_{[\![\theta\to\theta']\!]}$. Although types are interpreted by arenas, the actual games will be played in *prearenas*, which are defined in the same way as arenas with the exception that initial moves are O-questions. Given arenas A, B we define the prearena $A \to B$ as follows.

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

We write Val_θ for the set $I_{[\![\theta]\!]}$, that is,

$$
\mathsf{Val}_{\mathsf{unit}} = \mathsf{Val}_{\theta\to\theta'} = \star, \quad \mathsf{Val}_{\mathsf{int}} = \mathbb{Z}, \quad \mathsf{Val}_{\mathsf{ref}\,\theta} = \mathbb{A}_\theta.
$$

Let $\mathsf{Val} = \biguplus_\theta \mathsf{Val}_\theta$. A store Σ is a type-preserving finite partial function from $\mathbb{A}$ to Val, that is, $\Sigma : \mathbb{A} \rightharpoonup \mathsf{Val}$ and

$$
|\Sigma| \text{ finite} \wedge (a \in \mathsf{dom}(\Sigma) \cap \mathbb{A}_\theta \implies \Sigma(a) \in \mathsf{Val}_\theta).
$$

We write Sto for the set of all stores. A move-with-store on a (pre)arena A is a pair m^Σ with $m \in M_A$ and $\Sigma \in \mathsf{Sto}$.

Definition 5: A *justified sequence* on a prearena A is a sequence of moves-with-store from $M_A \uplus M_\phi$ such that, apart from the first move which must be of the form i^Σ with $\mathrm{i} \in I_A$, every move in s is equipped with a pointer to an earlier move, or to a name inside the store of an earlier move. These pointers are called *justification pointers* and are subject to the following constraints.

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

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

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

For each $S \subseteq \mathbb{A}$ and Σ we define:

$$
\Sigma^0(S) = S, \quad \Sigma^{i+1}(S) = \Sigma(\Sigma^i(S)) \cap \mathbb{A}, \quad \Sigma^*(S) = \bigcup_i \Sigma^i(S).
$$

The set of *available names* of a justified sequence is defined inductively by $\mathsf{Av}(\epsilon) = \emptyset$ and

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

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

Definition 6: Let A be a prearena. A justified sequence s on A is called a ***legal sequence***, written $s \in L_A$, if it satisfies the conditions below.

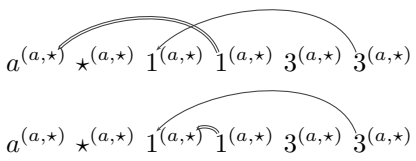
- No adjacent moves belong to the same player, and no move points to a move (or the store of a move) of the same player (*Alternation*).
- The justifier of each answer is the most recent unanswered question (*Bracketing*).

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

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

We write P_A for the set of plays on A.

Example 7: Here are two plays on $[\![\mathsf{ref}\,(\mathsf{int} \to \mathsf{int})]\!] \to [\![\mathsf{int} \to \mathsf{int}]\!]$ (for the sake of clarity, we omit pointers that would just point at preceding moves). We use double-line pointers to highlight the justification pointers pointing at stores.



The plays will be among those used to interpret the terms

$$
\begin{aligned}
x &: \mathsf{ref}\,(\mathsf{int} \to \mathsf{int}) \vdash\ !x : \mathsf{int} \to \mathsf{int} \\
x &: \mathsf{ref}\,(\mathsf{int} \to \mathsf{int}) \vdash \lambda h^{\mathsf{int}}.(!x)h : \mathsf{int} \to \mathsf{int}
\end{aligned}
$$

respectively. Note that these terms can be distinguished by the context

$$
\mathsf{let}\, x = \mathsf{new}_{\mathsf{int}\to\mathsf{int}}\, \mathsf{in}\, (\lambda f^{\mathsf{int}\to\mathsf{int}}.f(x := \lambda h^{\mathsf{int}}.0\,;\,0))\,[\_].
$$

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

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

Definition 8: A ***strategy*** σ on a prearena A, written $\sigma : A$, is a set of even-length plays of A satisfying:

- If $so^\Sigma p^{\Sigma'} \in \sigma$ then $s \in \sigma$ (*Even-prefix closure*).
- If $s \in \sigma$ and $s \sim t$ then $t \in \sigma$ (*Equivariance*).
- If $s_1 p_1^{\Sigma_1}, s_2 p_2^{\Sigma_2} \in \sigma$ and $s_1 \sim s_2$ then $s_1 p_1^{\Sigma_1} \sim s_2 p_2^{\Sigma_2}$ (*Nominal determinacy*).

Example 9: For each arena A there is an *identity strategy*, $\mathsf{id}_A : A \to A$, defined by

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

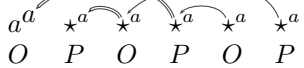
where the indices l, r distinguish the two copies of A, and $s' \restriction A_x$ is the subsequence of s' containing only moves from the x-copy, along with all ϕ-moves having a-ancestors from the x-copy (for some a).

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

Definition 10: Let s be an alternating justified sequence in A, $s' \sqsubseteq s$ be ending in $m^\Sigma m'^{\Sigma'}$ and let $a \in \mathsf{dom}(\Sigma) \cap \mathsf{dom}(\Sigma') \cap \mathbb{A}_\phi$ such that $m'^{\Sigma'}$ is not a-justified by m^Σ. We say that (s, s', a) is a ***copycat triple*** if, for all ϕ-moves n^T in s which have m^Σ or $m'^{\Sigma'}$ as an a-ancestor,

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

Example 11: We will be economical when writing stores and, in particular, components of the form $(a, \star)$ will often be written simply as a. The copycat behaviour is exemplified in the strategy $\sigma : \mathbb{A}_{\mathsf{unit} \to \mathsf{unit}} \to 1 = \{\epsilon, a^{(a,\star)} \star^{(a,\star)} s\}$ where $(a^{(a,\star)} \star^{(a,\star)} s, a^{(a,\star)} \star^{(a,\star)}, a)$ are copycat triples. For example, σ contains the play:



σ will turn out to denote $x : \mathsf{ref}\,(\mathsf{unit} \to \mathsf{unit}) \vdash () : \mathsf{unit}$.

We now turn to defining a suitable notion of interaction between plays. Given arenas A, B, C, we define the prearena $A \to B \to C$ by setting $M_{A \to B \to C} = M_{A \to B} \uplus M_C$, $I_{A \to B \to C} = I_A$ and:

$$\lambda_{A \to B \to C} = [\lambda_{A \to B}[\mathrm{i}_B \mapsto PQ], \overline{\lambda}_C]$$
$$\vdash_{A \to B \to C} = \vdash_{A \to B} \cup \{(\mathrm{i}_B, \mathrm{i}_C)\} \cup \vdash_C$$

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

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

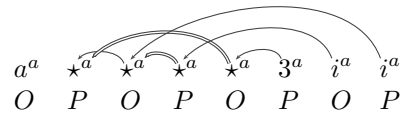
  then $\Sigma(a) = \Sigma'(a)$ and, moreover, if $a \in \mathbb{A}_\phi$ then $(u \restriction X, u' \restriction X, a)$ are a copycat triple, where X is the respective element of $\{AB, BC, AC\}$.

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

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

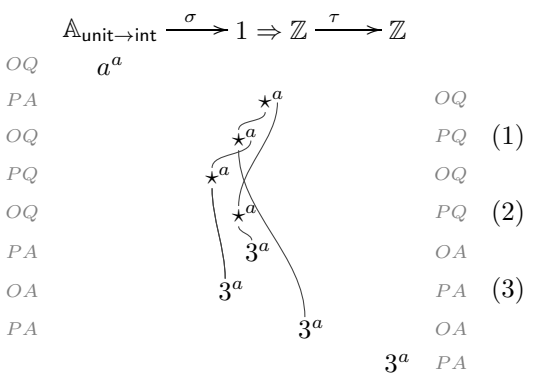
We shall be referring to the last condition in the definition as the *copycat condition*. According to it, during an interaction the players cannot change the parts of the store which regard names that are not available to them. Moreover, in the case these names are of functional type, the players are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

Example 13: Consider the strategy $\sigma : \mathbb{A}_{\mathsf{unit} \to \mathsf{int}} \to 1 \Rightarrow \mathbb{Z}$ given by the set of all even prefixes of plays of the form



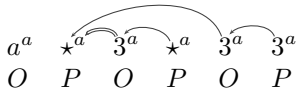
for all $i \in \mathbb{Z}$, and let $\tau : 1 \Rightarrow \mathbb{Z} \to \mathbb{Z}$ contain all even prefixes of $\star\star j\, j$ for all $j \in \mathbb{Z}$. Their interaction is depicted below.

We mark polarities for σ on the left of the diagram, and for τ on the right.

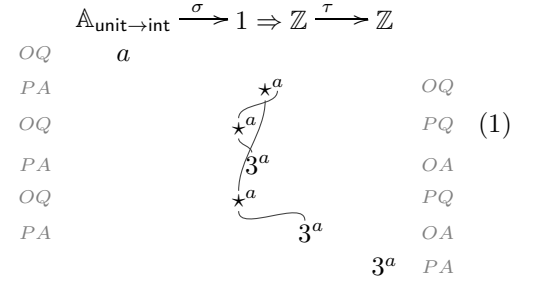


Consider point (1) in the interaction. In τ, P plays $\star^a$ but a is not available in the composite play at that point, hence P must copycat from that point on at a-descendants of (that occurrence of) $\star^a$. This is precisely what happens in points (2) and (3).

Let us now consider the same τ and the strategy $\sigma' : \mathbb{A}_{\mathsf{unit}\to\mathsf{int}} \to 1 \Rightarrow \mathbb{Z}$ given by the even prefixes of:



The following could seem a possible interaction. Note that the projection on the right is the play $\star\star 3\,3 \in \tau$, as the move played at (1) and its successor are removed by γ.



However, there is a violation of the copycat condition at (1). Again, a is not available at that point in τ and therefore P plays $\star^a$ and starts a copycat triple on a. But $\star^a$ is justified by a, thus violating the conditions for a copycat triple. Put simply, τ interrogates the value of a name which is not available to it.

Next we go on to show that the definition of interaction sequences is a sound basis for composing strategies. First we examine the switching discipline implied by it.

Lemma 14: If $u \in Int(A, B, C)$ then for all $u' m^{\Sigma} n^{T} \sqsubseteq u$ the following *switching conditions* are satisfied.

- n is a P-move in AB (in BC) iff m is an O-move in AB (resp. BC).
- n is an O-move in AC iff m is a P-move in AC.

Remark 15: Note that the switching conditions allow for m being a P-move in AC from A and its successor, n, being an O-move in AC from C. These are perfectly legal interactions.

Below by saying that m survives in $u \restriction_\gamma AB$ we mean that the application of γ on $u \restriction AB$ does not completely remove m^{Σ} (and similarly for the other cases). We say that m introduces a in $u' = u \restriction_\gamma AB$ if m survives in u' as, say, $m^{\Sigma'}$ and the latter introduces a in u' (and similarly for the other cases).

Lemma 16: Let $u \in Int(A, B, C)$ and suppose m^{Σ} introduces the name a in u. If m is a P-move in AB (a P-move in BC, an O-move in AC) then m introduces a in $u \restriction_\gamma AB$ (resp. $u \restriction_\gamma BC$, $u \restriction_\gamma AC$).

Corollary 17: For all $u \in Int(A, B, C)$ we have $\nu(u) = P(u \restriction_\gamma AB) \uplus P(u \restriction_\gamma BC) \uplus O(u \restriction_\gamma AC)$.

Lemma 18: If $u \in Int(A, B, C)$ then $(u \restriction_\gamma AB) \in P_{A\to B}$, $(u \restriction_\gamma BC) \in P_{B\to C}$ and $(u \restriction_\gamma AC) \in P_{A\to C}$.

Definition 19: Given strategies $\sigma : A \to B$ and $\tau : B \to C$ we define the composite strategy $\sigma;\tau : A \to C$ to be

$$\sigma;\tau = \{\, s \in P_{A\to C} \mid \exists u \in \sigma \| \tau.\ s = u \restriction_\gamma AC \,\}.$$

Proposition 20: Strategy composition is well-defined, i.e. for all $\sigma : A \to B$ and $\tau : B \to C$, $\sigma;\tau$ is a strategy on $A \to C$.

Proposition 21: Composition of strategies is associative.

Definition 22: $\mathcal{G}$ is the category of arenas and strategies, in which strategies in the prearena $A \to B$ are morphisms between A and B.

IV. Model

In this section we show that $\mathcal{G}$ possesses enough structure to model RefML. Specifically, we shall show that it is (equivalent to) a *$\nu\rho$-model* in the sense of [30, Definition 3.12].

We start off by demonstrating some specific strategies which constitute the main ingredients of a $\nu\rho$-model.

Example 23: For each type θ we define a strategy $\mathsf{eq}_\theta : \mathbb{A}_\theta \otimes \mathbb{A}_\theta \to \mathbb{Z}$ by:

$$\mathsf{eq}_\theta = \{\epsilon\} \cup \{(a, a')^{\Sigma} i^{\Sigma} s \mid i \in \{0, 1\},\ a = a' \iff i = 1\}$$

where $((a, a')^{\Sigma} i^{\Sigma} s, (a, a')^{\Sigma} i^{\Sigma}, b)$ is a copycat triple, for any $b \in \mathbb{A}$. That is to say, i^{Σ} leaves the functional values of the initial Σ unchanged.

Example 24: For each name a there is a unique (up to name-permutation) store Σ_a such that $\mathsf{dom}(\Sigma) = \Sigma^*(\{a\})$. More precisely:

$$\begin{array}{ll} \Sigma_a = \{(a, v_\theta)\} & \text{if } a \in \mathbb{A}_\theta, \theta \in \{\mathsf{unit}, \mathsf{int}, \theta' \to \theta''\} \\ \Sigma_a = \{(a, b)\} \cup \Sigma_b & \text{if } a \in \mathbb{A}_{\mathsf{ref}\,\theta}, b \in \mathbb{A}_\theta \end{array}$$

where $v_\theta = 0$ if $\theta = \mathsf{int}$, and $v_A = \star$ otherwise. Then, the following strategy

$$\mathtt{nu}_\theta : 1 \to \mathbb{A}_\theta = \{\epsilon, \star a^{\Sigma_a} \mid a \in \mathbb{A}_\theta\}$$

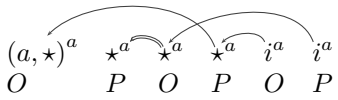
corresponds to the term new_θ (fresh reference creation) from Example 1.

A play $\mathrm{i}_A^\Sigma \mathrm{a}^{\Sigma'} s \in P_A$ is called a *thread* if there is at most one O-move in s which is justified, or a-justified for some name a, by $\mathrm{a}^{\Sigma'}$. Below we introduce strategies $\mathtt{upd}_\theta : \mathbb{A}_\theta \otimes [\![\theta]\!] \to 1$ and $\mathtt{drf}_\theta : \mathbb{A}_\theta \to [\![\theta]\!]$ for updating and dereferencing respectively, by representing only their threads. The full strategies are then obtained by interleaving their threads in such a way that one thread does not depend on another (i.e. in a *thread-independent* way, see below).

Example 25: We define

$$\begin{aligned}\mathtt{upd}_\theta = \{&(a,\mathrm{i}_\theta)^\Sigma \star^\Sigma s \mid \theta = \theta' \to \theta'',\ \mathrm{i}_\theta^\Sigma \star^\Sigma s \text{ an } a\text{-copycat}\}\\ &\cup \{\epsilon, (a,\mathrm{i}_\theta)^\Sigma \star^{\Sigma'} s' \mid \Sigma' = \Sigma[a \mapsto \mathrm{i}_\theta],\ a' \neq a,\\ &\qquad ((a,\mathrm{i}_\theta)^\Sigma \star^{\Sigma'} s', (a,\mathrm{i}_\theta)^\Sigma \star^{\Sigma'}, a') \text{ a copycat triple}\}\end{aligned}$$

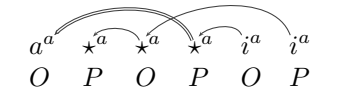
where by $\mathrm{i}_\theta^\Sigma \star^\Sigma s$ being an a-copycat we mean that P copycats between moves which are a-descendants of $\star^\Sigma$ and moves which are descendants of $(a,\mathrm{i}_\theta)^\Sigma$. Moreover, the first move of s (s'), if any, is a-justified by $\star^\Sigma$ (resp. a'-justified by $\star^{\Sigma'}$). For example, in the case of $\theta = \mathsf{int}$ we get $\mathtt{upd}_{\mathsf{int}} = \{\epsilon, (a,i)^{(a,j)} \star^{(a,i)}\}$. If θ is a function type then the update is not as explicit: the higher-order value has to be interrogated by O in order to be revealed. E.g. $\mathtt{upd}_{\mathsf{unit}\to\mathsf{int}} : \mathbb{A}_{\mathsf{unit}\to\mathsf{int}} \otimes (1 \Rightarrow \mathbb{Z}) \to 1$ has threads of the following form.



Example 26: We define

$$\begin{aligned}\mathtt{drf}_\theta = \{&a^\Sigma \star^\Sigma s \mid \theta = \theta' \to \theta'',\ a^\Sigma \star^\Sigma s \text{ an } a\text{-copycat}\}\\ &\cup \{\epsilon, a^\Sigma \mathrm{i}_\theta^\Sigma\ s' \mid \mathrm{i}_\theta = \Sigma(a),\ a' \in \mathbb{A},\\ &\qquad (a^\Sigma \mathrm{i}_\theta^\Sigma\ s', a^\Sigma \mathrm{i}_\theta^\Sigma, a') \text{ a copycat triple}\}\end{aligned}$$

where by $a^\Sigma \star^\Sigma s$ being an a-copycat we mean that P copycats between moves which are descendants of $\star^\Sigma$ and moves which are a-descendants of a^Σ. Moreover, the first move of s (s'), if any, is justified by $\star^\Sigma$ (resp. a'-justified by i_θ^Σ). For example, $\mathtt{drf}_{\mathsf{int}} = \{\epsilon, a^{(a,i)} i^{(a,i)}\}$. On the other hand, $\mathtt{drf}_{\mathsf{unit}\to\mathsf{int}} : \mathbb{A}_{\mathsf{unit}\to\mathsf{int}} \to 1 \Rightarrow \mathbb{Z}$ has threads of the following form.



Observe the duality with the play of the previous example. Updates and dereferencings are dual: the value in the initial move of upd is copied to the store of its second move, whereas the value in the second move of drf is copied from the store of its initial move.

Following [2] and [17], $\mathcal{G}$ can be shown equivalent to the Kleisli category of another category $\mathcal{G}_{\mathsf{sst}}$ equipped with a strong monad T. More precisely, $\mathcal{G}_{\mathsf{sst}}$ is the lluf subcategory of total single-threaded strategies [3, 17] such that the store values of the first move are not taken into account. Single-threaded strategies are composed by use of their thread-independent closures [17], only that in our case the notion of thread-independence has to be extended so that values of functional names coming from different threads induce copycat triples.

We say that a strategy $\sigma : A$ is *total* if for every $\mathrm{i}^\Sigma \in P_A$ there is $\mathrm{i}^\Sigma \mathrm{a}^\Sigma \in \sigma$ and, moreover, for every $a \in \mathsf{dom}(\Sigma)$ and $\mathrm{i}^\Sigma \mathrm{a}^\Sigma s \in \sigma$, $(\mathrm{i}^\Sigma \mathrm{a}^\Sigma s, \mathrm{i}^\Sigma \mathrm{a}^\Sigma, a)$ is a copycat triple. Put otherwise, a total strategy always answers the initial question at once, without introducing any new names nor updating the store. We will also consider an even stronger notion of strategy, which allows for cartesian products.

Definition 27: A total strategy $\sigma : A$ is *single-threaded* if for each $\mathrm{i}^\Sigma \mathrm{a}^\Sigma s \in \sigma$ there is at most one move which is (a-)justified by a^Σ. If, moreover,

- for all $\mathrm{i}^\Sigma \mathrm{a}^\Sigma m^{\Sigma'} s \in \sigma$, we have $\Sigma \subseteq \Sigma'$,
- for all $\mathrm{i}^\Sigma \mathrm{a}^\Sigma, \mathrm{i}^{\Sigma'} \mathrm{a}'^{\Sigma'} \in \sigma$, we have $\mathrm{a} = \mathrm{a}'$,

then we say that σ is *strongly single-threaded*.

We aim to construct a lluf subcategory of $\mathcal{G}$ containing only strongly single-threaded strategies. We start off with the following notions. For each play of the form $\mathrm{i}^\Sigma \mathrm{a}^\Sigma s$ we define its *current thread* inductively as follows.

$$\begin{aligned}&\mathsf{thr}(\mathrm{i}^\Sigma \mathrm{a}^\Sigma s m^T) = \mathrm{i}^{\Sigma'} \mathrm{a}^{\Sigma'} m^T && \text{if } m^T \text{ } (a\text{-})\text{justified by } \mathrm{a}^\Sigma\\ &\mathsf{thr}(\mathrm{i}^\Sigma \mathrm{a}^\Sigma s m^T) = \mathsf{thr}(\mathrm{i}^\Sigma \mathrm{a}^\Sigma s) m^T && \text{if } m^T \text{ } (a\text{-})\text{justified by } \mathrm{i}^\Sigma\\ &\mathsf{thr}(\mathrm{i}^\Sigma \mathrm{a}^\Sigma s m^T) = \mathsf{thr}(\mathrm{i}^\Sigma \mathrm{a}^\Sigma s') m^T && \text{if } m^T \text{ } (a\text{-})\text{justified in } s\end{aligned}$$

where s' is the longest prefix of s such that the move which (a-)justifies m^T is in $\mathsf{thr}(\mathrm{i}^\Sigma \mathrm{a}^\Sigma s')$, and $\Sigma' = T \restriction \nu(\mathrm{i}\,\mathrm{a})$. We say that $\mathrm{i}^\Sigma \mathrm{a}^\Sigma s \in P_A$ is *thread-independent*, written $s \in P_A^{\mathsf{ti}}$, if for all $s' p^\Sigma \sqsubseteq s$:

- $\mathsf{thr}(s' p^\Sigma) = \mathsf{thr}(s') p^\Sigma$,
- $\nu(\gamma(\mathsf{thr}(s' p^\Sigma))) \cap \nu(s') \subseteq \nu(\gamma(\mathsf{thr}(s')))$,
- if s' ends in o^T and $a \in \mathsf{dom}(T) \backslash \nu(\mathsf{thr}(s'))$ then $\Sigma(a) = T(a)$, and if $a \in \mathbb{A}_\phi$ then $(s, s' p^\Sigma, a)$ are a copycat triple.

For each strongly single-threaded strategy $\sigma : A$ we define:

$$\sigma^\dagger = \{s \in \sigma \mid |s| \leq 2\} \cup \{s \in P_A^{\mathsf{ti}} \mid \forall s' \sqsubseteq^e s.\ \gamma(\mathsf{thr}(s')) \in \sigma\}$$

Following [17], it can be shown that $\sigma^\dagger$ is a well-defined strategy. Moreover, if $\sigma : A \to B, \tau : B \to C$ are strongly single-threaded strategies then so is $\sigma^\dagger; \tau : A \to C$.

Definition 28: Let $\mathcal{G}_{\mathsf{sst}}$ be the lluf subcategory of $\mathcal{G}$ where morphisms are strongly single-threaded strategies and morphism composition is defined as above (i.e. via $\sigma^\dagger; \tau$).

The restriction to $\mathcal{G}_{\mathsf{sst}}$ allows us to form finite products. The one-element arena 1 is the terminal object, while binary products are given by the following construction. Given strongly thread-independent $\sigma : A \to B$ and $\tau : A \to C$, define $\langle \sigma, \tau \rangle : A \to B \otimes C$ to be the strategy which replies to each initial i_A^Σ with $(\mathrm{i}_B, \mathrm{i}_C)^\Sigma$, if $\mathrm{i}_A^\Sigma \mathrm{i}_B^\Sigma \in \sigma$ and $\mathrm{i}_A^\Sigma \mathrm{i}_C^\Sigma \in \tau$, and from that point on it plays either as σ or as τ depending on whether the current thread is one in AB or in AC respectively. Moreover, the lifting operator [2] yields a strong monad T in $\mathcal{G}_{\mathsf{sst}}$ with exponentials, with the associated Kleisli category $\mathcal{G}_{\mathsf{sst}}^T$ being equivalent to $\mathcal{G}$.

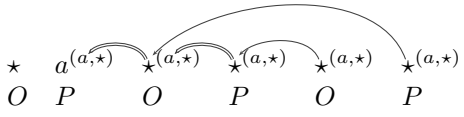
To interpret the remaining constructs of RefML in $\mathcal{G}$, we follow [30] by showing that $\mathcal{G}_{\mathsf{sst}}$ is a *$\nu\rho$-model* in the sense of [30, Definition 3.12]. The main constructs of the model are the morphisms $\mathtt{eq}_\theta, \mathtt{nu}_\theta, \mathtt{upd}_\theta, \mathtt{drf}_\theta$ presented above (as morphisms in $\mathcal{G}$ rather than in $\mathcal{G}^T_{\mathsf{sst}}$). Moreover, for each finite $u = \{a_1, \cdots, a_n\} \subseteq \mathbb{A}$ we set $[\![u]\!]$ to be the arena with moves of the form $(a'_1, \cdots, a'_n) \sim (a_1, \cdots, a_n)$, all of them initial. The functors $[\![u]\!] \otimes \_$ yield the initial state comonads of [30]. Checking that the diagrams of [30, Definition 3.12] commute is routine. It follows that $\mathcal{G}$ is a model of RefML.

Definition 29: Given RefML types $\theta_1, \cdots, \theta_n, \theta$ and finite $u \subseteq \mathbb{A}$ let us write $[\![u, \theta_1, \cdots, \theta_n \vdash \theta]\!]$ for the prearena $([\![u]\!] \otimes [\![\theta_1]\!] \otimes \cdots \otimes [\![\theta_n]\!]) \to [\![\theta]\!]$. Any RefML term-in-context $u, x_1 : \theta_1, \cdots, x_n : \theta_n \vdash M : \theta$ can then be interpreted in a canonical way as a strategy in $[\![u, \theta_1, \cdots, \theta_n \vdash \theta]\!]$, which we shall denote by $[\![u, x_1 : \theta_1, \cdots, x_n : \theta_n \vdash M : \theta]\!]$.

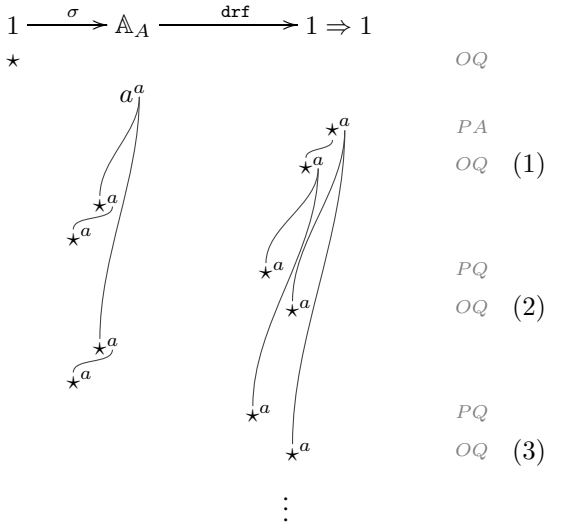
Example 30: We revisit the circular reference term $\vdash \mathsf{circ} : \mathsf{ref}\,(\mathsf{unit} \to \mathsf{unit})$ from Example 1. $[\![\vdash !\mathsf{circ}]\!]$ is given by the composition of

$$\sigma = [\![\mathsf{new}\, x \,\mathsf{in}\, x := \lambda y^{\mathsf{unit}}.(!x)y;\, x]\!] : 1 \to \mathbb{A}_{1 \Rightarrow 1}$$

with $\mathtt{drf}_{\mathsf{unit}\to\mathsf{unit}} : \mathbb{A}_{1\Rightarrow 1} \to (1 \Rightarrow 1)$. The former contains plays of the form:



The composition is depicted below; we mark the polarities of moves in the composite play.



Consider point (1) in the interaction. O plays $\star^a$ but a is not available in the composite play at that point, hence O must copycat from that point on for moves a-justified by $\star^a$. This is precisely what happens in points (2) and (3). Thus, observe that $[\![\vdash !\mathsf{circ}]\!]$ never answers the question played in (1), that is, the strategy $[\![\vdash (!\mathsf{circ})()]\!]$ diverges (it equals the empty strategy).

V. Soundness

The purpose of this section is to show that complete-trace inclusion is sound for contextual approximation (Proposition 33). We follow the traditional route by establishing Computational Soundness and Adequacy. The former is a direct consequence of working with a *$\nu\rho$-model*. We restate the relevant result [30, Proposition 3.17] below. For any term M, state S and finite set of names u, let the terms $(S)^\bullet$ and $\mathsf{new}\, u \,\mathsf{in}\, M$ be inductively defined by:

$$(\emptyset)^\bullet = () \qquad (\{(a, V)\} \uplus S)^\bullet = a := V; (S)^\bullet$$
$$\mathsf{new}\,\emptyset\,\mathsf{in}\,M = M \quad \mathsf{new}\,(\{a\} \uplus u)\,\mathsf{in}\,M = \mathsf{new}\,a\,\mathsf{in}\,(\mathsf{new}\,u\,\mathsf{in}\,M)$$

where $\mathsf{new}\, a \,\mathsf{in}\, M$ abbreviates $\mathsf{let}\, x = \mathsf{new}_\theta \,\mathsf{in}\, M[x/a]$ (if $a \in \mathbb{A}_\theta$). Note that an implicit ordering of S and u needs to be assumed but any such ordering would give (semantically and operationally) equivalent terms.

Proposition 31: If $(S, M) \twoheadrightarrow (S', M')$ then, taking u, u' to be respectively $\mathsf{dom}(S), \mathsf{dom}(S')$, we have $[\![u \vdash (S)^\bullet; M]\!] = [\![u \vdash \mathsf{new}\,(u' \setminus u)\,\mathsf{in}\,(S')^\bullet; M]\!]$.

For Adequacy, we need to show that, for any divergent closed term $\vdash M : \mathsf{unit}$ (i.e. if $M \Downarrow$), we have $[\![\vdash M]\!] = \{\epsilon\}$. Note that $M \Downarrow$ if, and only if, $(\emptyset, M)$ induces an infinite reduction sequence. Recall the reduction rule for dereferencing:

$$\textsc{Drf} \quad (S, !a) \to (S, S(a)).$$

Observe that by removing it from the reduction relation we are left with an extension of the ν-calculus [29] with dummy state that can be updated but not read. Because the ν-calculus is strongly normalising, so is the restriction of our reduction relation with the Drf rule removed (in the restricted language an attempt to evaluate $!a$ is regarded as termination). Thus, if M diverges then $(\emptyset, M)$ induces a reduction sequence with infinitely many Drf reduction steps. We will exploit this fact by adding a counter to M that increases its value just before a Drf rule can be applied. Since M diverges, the value of the counter will have to be unbounded.

Proposition 32: For all $\vdash M : \mathsf{unit}$, if $M \not\Downarrow$ then $[\![M]\!] = \{\epsilon\}$.

Proof: Suppose, for the sake of contradiction, that $M \not\Downarrow$ and $[\![\vdash M]\!] = \{\epsilon, \star\star\}$. For any term $u, \Gamma \vdash N : \theta$ and $a \in \mathbb{A}_{\mathsf{int}} \setminus u$ construct $a, u, \Gamma \vdash N_a$ by recursively replacing each subterm of N of the shape $!N'$ with $a := (!a{+}1); !N'$. Observe that each $s \in [\![u, \Gamma \vdash N]\!]$ induces some $s' \in [\![a, u, \Gamma \vdash N_a]\!]$ such that a appears in s' only in stores (and in a single place in the initial move) and O never changes the value of a. Then, for each $i \in \mathbb{Z}$ take N_i to be the term $\mathsf{new}\, a \,\mathsf{in}\, (a := i; N_a; !a)$. Because $\star\star \in [\![\vdash M]\!]$, we shall have $\star j \in [\![\vdash M_0]\!]$ for some $j \in \mathbb{Z}$.

On the other hand, each play corresponding to N_i is obtained from a play s' for N_a such that the initial value of a is i, a appears in s' only in stores (and in a single place of the initial move) and O never changes the value of a. Moreover, P never decreases the value of a in s'. Thus, if sj^Σ is a complete play of $[\![N_i]\!]$, then $j \geq i$. We shall find a term contradicting this by considering the infinite reduction sequence of $(\emptyset, M)$. It must

have infinitely many DRF steps, so suppose $(\emptyset, M) \twoheadrightarrow (S, M')$ in $j+1$ such steps. Then we obtain $(\emptyset, M_0) \twoheadrightarrow (\{(a, j+1)\} \uplus S_a, (M')_a; !a)$. By Proposition 31 we have that $\star j \in [\![\vdash \mathsf{new}\, a, u \,\mathsf{in}\, (a := j+1; (S_a)^\bullet; (M')_a; !a)]\!]$, where $u = \mathsf{dom}(S)$. Since $[\![\vdash \mathsf{new}\, a, u \,\mathsf{in}\, (a := j+1; (S_a)^\bullet; (M')_a; !a)]\!]$ is the same as $[\![\vdash (\mathsf{new}\, u \,\mathsf{in}\, (S)^\bullet; M')_{j+1}]\!]$ (assignment $a := j+1$ commutes with the creation of u), we obtain $\star j \in [\![\vdash (\mathsf{new}\, u \,\mathsf{in}\, (S)^\bullet; M')_{j+1}]\!]$, a contradiction. ∎

Recall that a play is *complete* if each question occurring in it justifies an answer. Given a set of plays X, let us write $\mathsf{comp}(X)$ for the set of complete plays in X. By Proposition 32 and monotonicity of composition we can conclude the following.

Proposition 33: Let $\Gamma \vdash M_1, M_2 : \theta$ be terms of RefML. $\mathsf{comp}([\![\Gamma \vdash M_1 : \theta]\!]) \subseteq \mathsf{comp}([\![\Gamma \vdash M_2 : \theta]\!])$ implies $\Gamma \vdash M_1 \sqsubseteq\!\!\!\approx M_2$.

VI. COMPLETENESS

Here we prove the converse of Proposition 33, which follows from the definability result below.

Lemma 34: Let s be a complete play in $[\![\Theta \vdash \theta]\!]$, where $\Theta = \{\theta_1, \cdots, \theta_n\}$, and $\mathsf{pref}(s)$ be the set of its even-length prefixes. There exist $\Gamma = \{x_1 : \theta_1, \cdots, x_n : \theta_n\}$ and $\Gamma \vdash M_s : \theta$ such that $\mathsf{comp}([\![\Gamma \vdash M_s : \theta]\!]) = \mathsf{comp}(\mathsf{pref}(s))$.

The proof of the lemma consists of two stages. First we show that a number of simplifying assumptions can be made about the shape of s. Subsequently, for plays satisfying the additional constraints, we are going to reduce the problem to an instance involving a play of strictly smaller length.

Let s be non-empty. We show that it suffices to prove the lemma for plays s subject to the following three conditions. Below we give constructions that eliminate violations of the respective conditions without altering the length of plays.

1. Stores associated with the first two moves have the same domains.

Suppose $s = \mathrm{i}^{\Sigma_\mathrm{i}}\, m^{\Sigma_m}\, s_1$ is a play in $[\![\Theta \vdash \theta]\!]$. Let $a \in \mathsf{dom}(\Sigma_m) \setminus \mathsf{dom}(\Sigma_\mathrm{i})$ with $a \in \mathbb{A}_{\theta_a}$. Consider the play $s' = (\mathrm{i}, a)^{\Sigma'_\mathrm{i}}\, m^{\Sigma'_m}\, s'_1$ in $[\![\Theta, \mathsf{ref}\, \theta_a \vdash \theta]\!]$, where $\Sigma'_\mathrm{i} = \Sigma_\mathrm{i} \uplus \Sigma_a$ (see Example 24 for Σ_a) and $m^{\Sigma'_m}\, s'_1$ is equal to $m^{\Sigma_m}\, s_1$ in which the stores have been extended with $\Sigma_a \restriction (\mathsf{dom}(\Sigma_a) \setminus \{a\})$. Given $\Gamma, z : \mathsf{ref}\, \theta_a \vdash M_{s'} : \theta$, we can take M_s to be $\mathsf{new}\, z \,\mathsf{in}\, M_{s'}$.

2. No move points at the store of the initial move.

Let $s = \mathrm{i}^{\Sigma_\mathrm{i}} \cdots m^{\Sigma_m} \cdots$, and suppose that m is justified by $a \in \mathsf{dom}(\Sigma_\mathrm{i})$ such that $a \in \mathbb{A}_{\theta'_1 \to \theta'_2}$. Then there exist $k, l \in \mathbb{N}$ such that $\Sigma_\mathrm{i}^k(m_l) = a$, where $\mathrm{i} = (m_1, \cdots, m_n)$.

Consider $s' = (\mathrm{i}, \star)^{\Sigma_\mathrm{i}} \cdots m^{\Sigma_m} \cdots$, which is a play in $\Theta, \theta'_1 \to \theta'_2 \vdash \theta$. Suppose $\Gamma, f : \theta'_1 \to \theta'_2 \vdash M_{s'} : \theta$ satisfies the Lemma. Then we can take M_s to be $\mathsf{let}\, f = !^k x_l \,\mathsf{in}\, M_{s'}$.

3. The second move is an answer that does not justify any other moves.

Suppose the second move of s is a question. Thanks to 2 we can assume it is justified by the initial move (rather than its store). Consequently, the presence of the question indicates that the play proceeds in $[\![\Theta, \theta'_1 \to \theta'_2 \vdash \theta]\!]$, where the question and the corresponding answer originate from θ'_1 and θ'_2 respectively. Thus,

$$s = \mathrm{i}^{\Sigma_\mathrm{i}}\, \mathrm{q}_{\theta'_1}^{\Sigma_\mathrm{q}}\, s_1\, \mathrm{a}_{\theta'_2}^{\Sigma_\mathrm{a}}\, s_2\, \mathrm{a}_\theta^{\Sigma}\, s_3.$$

Now consider the play

$$s' = (\mathrm{i}, a)^{\Sigma_\mathrm{i}, a}\, \mathrm{a}_{\theta'_1}^{\Sigma_\mathrm{q}, a}\, s'_1\, \mathrm{q}_{\theta'_2}^{\Sigma_\mathrm{a}, a}\, s'_2\, \mathrm{a}_\theta^{\Sigma, a}\, s'_3$$

in $[\![\Theta, \theta'_1 \to \theta'_2, \mathsf{ref}\, (\theta'_2 \to \theta) \vdash \theta'_1]\!]$, where s'_i $(i = 1, 2, 3)$ stands for s_i in which $(a, \star)$ was added to each store. Note that $\mathrm{a}_{\theta'_1}$ is the same move as $\mathrm{q}_{\theta'_1}$ only in answer position, and dually for $\mathrm{q}_{\theta'_2}, \mathrm{a}_{\theta'_2}$. Given $\Gamma, f : \theta'_1 \to \theta'_2, z : \mathsf{ref}\, (\theta'_2 \to \theta) \vdash M_{s'} : \theta'_1$, one can take M_s to be

$$\mathsf{new}\, z \,\mathsf{in}\, \mathsf{let}\, v = M_{s'} \,\mathsf{in}\, (!z)(fv).$$

So the validity of the lemma in general can be derived from that for plays in which the second move is an answer. Further, let us assume that the answer justifies a move and $s = \mathrm{i}^{\Sigma_\mathrm{i}} \star^{\Sigma_\star} s_1\, m^{\Sigma_m}\, s_2$ is a play from $[\![\Theta \vdash \theta'_1 \to \theta'_2]\!]$. Consider the complete play $s' = (\mathrm{i}, a)^{\Sigma_\mathrm{i}, a} \star^{\Sigma_\star, a} s'_1\, m^{\Sigma_m, a}\, s'_2$ in $[\![\Theta, \mathsf{ref}\, (\theta'_1 \to \theta'_2) \vdash \mathsf{unit}]\!]$, where s'_i $(i = 1, 2)$ is the same as s_i except that $(a, \star)$ was added to the store of each move. Moreover, all moves in s_1, s_2 pointing to $\star^{\Sigma_\star}$ now point at its store, just like m. Given $\Gamma, z : \mathsf{ref}\, (\theta'_1 \to \theta'_2) \vdash M_{s'} : \mathsf{unit}$, one can take M_s to be $\mathsf{new}\, z \,\mathsf{in}\, (M_{s'}; !z)$.

Consequently, in the rest of the proof we can assume that the transformations used in the proof above have been consecutively applied to the given play.

For the second stage of the proof let us assume that $s = \mathrm{i}^{\Sigma_\mathrm{i}}\, \mathrm{a}^{\Sigma_\mathrm{a}}\, \mathrm{q}^{\Sigma_\mathrm{q}}\, m^{\Sigma_m}\, s_1$ is a play in $[\![\Theta \vdash \theta]\!]$ such that no justification pointer points at Σ_i or a, and $\mathsf{dom}(\Sigma_\mathrm{i}) = \mathsf{dom}(\Sigma_\mathrm{a})$. Hence q must point at some $b \in \mathsf{dom}(\Sigma_\mathrm{a})$. Suppose $b \in \mathbb{A}_{\theta'_1 \to \theta'_2}$ and let Σ^fn be the restriction of Σ to names associated with function types. Consider the play

$$s' = (\mathrm{i}, \mathrm{q}, \mathsf{dom}(\overline{\Sigma^\mathrm{fn}_\mathrm{a}}), X)^{\Sigma_\mathrm{q} + \overline{\Sigma^\mathrm{fn}_\mathrm{a}}}\, m^{\Sigma_m + \overline{\Sigma^\mathrm{fn}_\mathrm{a}}}\, s'_1,$$

where:

- s'_1 is the same as s_1 except that each store was augmented with a fresh copy of $\Sigma^\mathrm{fn}_\mathrm{a}$, referred to as $\overline{\Sigma^\mathrm{fn}_\mathrm{a}}$;
- pointers to Σ_a from s_1 have been redirected to $\overline{\Sigma^\mathrm{fn}_\mathrm{a}}$ associated with m, and pointers to i and q now point at the new initial move.
- X is a list of names chosen so as to make s' satisfy the frugality condition, i.e. $X = \mathsf{dom}(\Sigma_\mathrm{q}) \setminus \Sigma^*_\mathrm{q}(\nu(\mathrm{i}) \cup \nu(\mathrm{q}))$. Because $\mathsf{dom}(\Sigma_\mathrm{i}) = \mathsf{dom}(\Sigma_\mathrm{a})$, we have $X \subseteq \mathsf{dom}(\Sigma_\mathrm{i})$.

Let $\mathsf{ty}(a)$ stand for $\mathsf{ref}\, \theta$ provided $a \in \mathbb{A}_\theta$. Using this notation, s' is now a play in $[\![\Theta, \theta'_1, [\mathsf{ty}(a)]_{a \in \mathsf{dom}(\Sigma^\mathrm{fn}_\mathrm{a})}, [\mathsf{ty}(x)]_{x \in X} \vdash \theta'_2]\!]$. Crucially, s' is shorter than s. Suppose

$$\Gamma, y : \theta'_1, [\overline{z_a} : \mathsf{ty}(a)], [w_x : \mathsf{ty}(x)] \vdash M_{s'} : \theta'_2$$

satisfies Lemma 34. In order to define M_s we need to be able to copy the store $\overline{\Sigma^\mathrm{fn}_\mathrm{a}}$ at m to a. To that end we introduce

auxiliary variables $\overline{\overline{z_a}}$ in which the higher-order state will be recorded immediately before m is played and add them to $M_{s'}$ to obtain

$$\Gamma, y : \theta_1', [\overline{z_a} : \mathsf{ty}(a)], [w_x : \mathsf{ty}(x)], [\overline{\overline{z_a}} : \mathsf{ty}(a)] \vdash M_{s''} : \theta_2'$$

as follows.

- If m is an answer, we define $M_{s''}$ to be
$$\mathsf{let}\, v = M_{s'}\, \mathsf{in}\, [\overline{\overline{z_a}} := !\overline{z_a}]; v\,.$$
- If m is a question, we can assume that $M_{s'}$ comes from Step 3 discussed above, i.e. has the form
$$\mathsf{new}\, z\, \mathsf{in}\, \mathsf{let}\, v = N\, \mathsf{in}\, (!z)(fv).$$
Then we can define $M_{s''}$ to be
$$\mathsf{new}\, z\, \mathsf{in}\, \mathsf{let}\, v = N\, \mathsf{in}\, [\overline{\overline{z_a}} := !\overline{z_a}]; (!z)(fv).$$

To define M_s, we need some more notation. Let $\mathrm{i} = (m_1, \cdots, m_n)$. Given $a \in \mathsf{dom}(\Sigma_\mathrm{i})$, we write a_Γ for $!^k x_l$ provided $\Sigma_\mathrm{i}^k(m_l) = a$. Note that k and l may not be determined uniquely. This will not be problematic in what follows, because the purpose of the notation is to give us a way of accessing a through the context. We can now define M_s to be:

$$\begin{array}{l}\mathsf{new}\, \overline{z_a}, \overline{\overline{z_a}}\, \mathsf{in}\, \mathsf{let}\, [w_x = x_\Gamma]_{x\in X}\, \mathsf{in}\\ \quad \mathsf{assert}(\mathrm{i}^{\Sigma_\mathrm{i}});\\ \quad [a_\Gamma := \Sigma_\mathrm{a}(a)]_{a\in\mathsf{dom}(\Sigma_\mathrm{a}^{\mathsf{int}})}; [a_\Gamma := \Sigma_\mathrm{a}(a)_\Gamma]_{a\in\mathsf{dom}(\Sigma_\mathrm{a}^{\mathsf{ref}})};\\ \quad [a_\Gamma := \lambda h.(!\overline{\overline{z_a}})h]_{a\in\mathsf{dom}(\Sigma_\mathrm{a}^{\mathsf{fn}}), a\neq b};\\ \quad b_\Gamma := M_b;\ \mathsf{play}(\mathrm{a})\end{array}$$

where

- $\mathsf{assert}(\mathrm{i}^{\Sigma_\mathrm{i}})$ has the shape $\mathsf{if}\ \mathit{condition}\ \mathsf{then}\ ()\ \mathsf{else}\ \Omega_{\mathsf{unit}}$ and $\mathit{condition}$ is a conjunction of constraints of the form $!^{k_1}x_{l_1} =\!!^{k_2}x_{l_2}$ or $!^{k_1}x_{l_1} \neq\!!^{k_2}x_{l_2}$ (between reference cells of the same type or integers) characterising[5] $\mathrm{i}^{\Sigma_\mathrm{i}}$;
- $\Sigma_\mathrm{a}^{\mathsf{int}}$ and $\Sigma_\mathrm{a}^{\mathsf{ref}}$ are restrictions of Σ_a to integer references and references to references respectively;
- M_b is the following term
$$\begin{array}{l}\mathsf{let}\, n = \mathsf{ref}\,(0)\, \mathsf{in}\\ \quad \lambda y^{\theta_1}.\mathsf{if}\,(n = 0)\,\mathsf{then}\,(n := !n + 1; M_{s''})\,\mathsf{else}\,(!\overline{\overline{z_b}})y\end{array}$$
M_b behaves like $M_{s''}$ when queried for the first time, afterwards it follows the behaviour of the function stored at $\overline{b}$ at m in s';
- $\mathsf{play}(\mathrm{a})$ is $()$ for $\theta \equiv \mathsf{unit}$, a for $\theta \equiv \mathsf{int}$, a_Γ for $\theta \equiv \mathsf{ref}\,(\cdots)$ and $\lambda x^{\theta_1''}.\Omega_{\theta_2''}$ for $\theta \equiv \theta_1'' \to \theta_2''$.

Lemma 34 can now be used to prove the following proposition. Propositions 33 and 35 entail the main result.

Proposition 35: Let $\Gamma \vdash M_1, M_2 : \theta$ be terms of RefML. $\Gamma \vdash M_1 \sqsubseteq\!\!\!\approx M_2$ implies $\mathsf{comp}([\![\Gamma \vdash M_1 : \theta]\!]) \subseteq \mathsf{comp}([\![\Gamma \vdash M_2 : \theta]\!])$.

Theorem 36 (Full Abstraction): Let $\Gamma \vdash M_1, M_2 : \theta$ be terms of RefML. $\Gamma \vdash M_1 \sqsubseteq\!\!\!\approx M_2$ if, and only if, $\mathsf{comp}([\![\Gamma \vdash M_1 : \theta]\!]) \subseteq \mathsf{comp}([\![\Gamma \vdash M_2 : \theta]\!])$. Hence, $\Gamma \vdash M_1 \cong M_2$ if, and only if, $\mathsf{comp}([\![\Gamma \vdash M_1 : \theta]\!]) = \mathsf{comp}([\![\Gamma \vdash M_2 : \theta]\!])$.

[5]E.g. for $(a, b, a)^{(a,d)(d,5)(b,c)(c,d)}$ in $[\![\mathsf{ref}^2\,\mathsf{int}, \mathsf{ref}^3\,\mathsf{int}, \mathsf{ref}^2\,\mathsf{int} \vdash \mathsf{unit}]\!]$, one would use $x_1 = x_3$, $x_1 \neq\!! x_2$, $!x_1 =\!!x_2$ and $!!x_1 = 5$.

References

[1] S. Abramsky, D. R. Ghica, A. S. Murawski, and C.-H. L. Ong. Applying game semantics to compositional software modelling and verification. In *TACAS'04*, LNCS 2988, pp 421–435.

[2] 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 *LICS'04*, pp 150–159.

[3] S. Abramsky, K. Honda, and G. McCusker. Fully abstract game semantics for general references. In *LICS'98*, pp 334–344.

[4] S. Abramsky and G. McCusker. Call-by-value games. In *CSL'97*, LNCS 1414, pp 1–17.

[5] A. Ahmed, D. Dreyer, and A. Rossberg. State-dependent representation independence. In *POPL'09*, pp 340–353.

[6] L. Birkedal, K. Støvring, and J. Thamsborg. Realisability semantics of parametric polymorphism, general references and recursive types. *MSCS*, 20(4):655–703, 2010.

[7] N. Bohr and L. Birkedal. Relational reasoning for recursive types and references. In *APLAS'06*, LNCS 4279, pp 79–96.

[8] K. B. Bruce, L. Cardelli, and B. C. Pierce. Comparing object encodings. *Inf. Comput.*, 155(1-2):108–133, 1999.

[9] N. Charlton and B. Reus. A decidable class of verification conditions for programs with higher order store. *ECEASST*, 23, 2009.

[10] D. Dreyer, G. Neis, and L. Birkedal. The impact of higher-order state and control effects on local relational reasoning. In *ICFP'10*, pp 143–156.

[11] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Asp. Comput.*, 13:341–363, 2002.

[12] D. R. Ghica and G. McCusker. The regular language semantics of second-order Idealized Algol. *Theor. Comput. Sci.*, 309:469–502, 2003.

[13] K. Honda and N. Yoshida. Game-theoretic analysis of call-by-value computation. *Theor. Comput. Sci.*, 221(1–2):393–456, 1999.

[14] V. Koutavas and M. Wand. Small bisimulations for reasoning about higher-order imperative programs. In *POPL'06*, pp 141–152.

[15] J. Laird. Game semantics for higher-order concurrency. In *FSTTCS'06*, LNCS 4337, pp 417–428.

[16] J. Laird. A fully abstract trace semantics for general references. In *ICALP'07*, LNCS 4596, pp 667–679.

[17] J. Laird. A game semantics of names and pointers. *Ann. Pure Appl. Logic* , 151:151–169, 2008.

[18] J. Laird. Game semantics for call-by-value polymorphism. In *ICALP'10*, LNCS 6199, pp 187–198.

[19] S. B. Lassen and P. B. Levy. Typed normal form bisimulation for parametric polymorphism. In *LICS'08*, pp 341–352.

[20] P. B. Levy. Possible world semantics for general storage in call-by-value. In *CSL'02*, LNCS 2471, pp 232–246.

[21] P. B. Levy. Global state considered helpful. *Electr. Notes Theor. Comput. Sci.*, 218:241–259, 2008.

[22] P.-A. Melliès and N. Tabareau. An algebraic account of references in game semantics. *Electr. Notes Theor. Comput. Sci.*, 249:377–405, 2009.

[23] A. S. Murawski and N. Tzevelekos. Full abstraction for Reduced ML. In *FOSSACS'09*, LNCS 5504, pp 32–47.

[24] A. M. Pitts and I. Stark. On the observable properties of higher order functions that dynamically create local names, or: What's new? In *MFCS'93*, pp 122–141.

[25] J. C. Reynolds. The essence of Algol. In J. W. de Bakker and J.C. van Vliet, editors, *Algorithmic Languages*, pp 345–372.

[26] D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. In *LICS'07*, pp 293–302, 2007.

[27] S. B. Sanjabi and C.-H. L. Ong. Fully abstract semantics of additive aspects by translation. In *AOSD'07*, ACM ICPS 208, pp 135–148.

[28] J. Schwinghammer, L. Birkedal, B. Reus, and H. Yang. Nested hoare triples and frame rules for higher-order store. In *CSL'09*, LNCS 5771, pp 440–454.

[29] I. D. B. Stark. *Names and Higher-Order Functions*. PhD thesis, University of Cambridge, 1995.

[30] N. Tzevelekos. Full abstraction for nominal general references. *LMCS*, 5(3), 2009.

[31] N. Yoshida, K. Honda, and M. Berger. Logical reasoning for higher-order functions with local state. *LMCS*, 4(4), 2008.