Operational Nominal Game Semantics

Guilhem Jaber

Queen Mary University of London, UK

Abstract. We present a formal correspondence between Laird's trace semantics and the nominal game model of Murawski and Tzevelekos for RefML, a call-by-value language with higher-order references. This gives an operational flavor to game semantics, where denotation of terms is generated via an interactive reduction, which allows to reduce terms with free functional variables, and where pointer structure is represented with name pointers. It also leads to transferring the categorical structure defined on the game model to the trace model. Then, representing the notion of view from game semantics in terms of available name pointers, we restrict our trace semantics to GroundML, a language with first-order references and show its full abstraction using a correspondence with visible strategies. This gives the first fully abstract trace model for this language.

1 Introduction

Game Semantics [6,3] is a powerful theory to build fully abstract denotational models of various programming languages. The denotation of a term is represented as a *strategy*, a set of *plays* between that term and any context in a *game arena*, which sets the rules the plays have to satisfy. One of its most important contributions, the so-called "Abramsky Cube", is the characterization of the absence of various impure effects in terms of extra conditions on the denotation of terms, namely *well-bracketing* for the absence of control operators, *visibility* for the absence of higher-order store, *innocence* for pure terms. In recent years, game semantics has been developed to deal with languages with nominal aspects, from the ν-calculus [1], an extension with storage cells [10], to ML-like languages with higher-order nominal references [13].

The starting point of this article is the nominal game semantics of Murawski and Tzevelekos [13], which is fully abstract for RefML, but also for GroundML as soon as one adds a visibility condition to strategies [14]. As opposed to previous games models for languages with stores, initiated by Abramsky, Honda and McCusker [2], it uses nominal techniques [5], to avoid the problem of *bad variables*. In a more operational setting, Laird [9] has introduced a trace semantics for a variant of RefML, and has proven its full abstraction. This model marries a trace representation inspired by game semantics with an operational definition, *i.e.* denotations of terms are computed via a rewriting system rather than defined by induction on their typing judgment.

In this article, we introduce a trace semantics for RefML, whose definition is a typed variant to the one introduced by Laird. Traces are generated by an *interactive reduction*, which can be seen as an extension of the usual operational semantics to open terms with free functional variables. Then, the denotation of terms is defined via *trace-strategies*, *i.e.* sets of traces that terms generate using this reduction. In fact, traces can be seen

A. Pitts (Ed.): FOSSACS 2015, LNCS 9034, pp. 264–278, 2015.
DOI: 10.1007/978-3-662-46678-0_17

$$
\begin{array}{lll}
\tau, \sigma & \stackrel{def}{=} & \mathrm{Int} \mid \mathrm{ref}\,\tau \mid \tau \to \sigma \\
v & \stackrel{def}{=} & () \mid \widehat{n} \mid x \mid l \mid \lambda x : \tau.M \mid \Omega_\tau \qquad \text{(where } n \in \mathbb{Z}, l \in \mathrm{Loc}\text{)} \\
M, N & \stackrel{def}{=} & v \mid MN \mid M + N \mid \mathtt{case_z}(M_1)(M_2, M_3) \mid M == N \mid \\
 & & \mathrm{ref}\, M \mid !M \mid M := N \\
K & \stackrel{def}{=} & \bullet \mid KM \mid vK \mid K + M \mid v + K \mid \mathtt{case_z}(K)(M, M') \mid \mathrm{ref}\, K \mid \\
 & & !K \mid K := M \mid v := K \mid K == M \mid v == K
\end{array}
$$

Fig. 1. Definition of RefML

as a representation of plays used in game semantics where the usual pointer structure, which represents the causality between the different moves, is encoded with variables. Such variables that are of functional type are called *name pointers*.

Following this idea, we build a correspondence between the trace and the game model of RefML. To do so, we impose on trace-strategies a categorical apparatus that capture call-by-value languages, namely a *closed-Freyd category* [16]. To build such a structure on traces, we recast the definitions of game semantics from [10,13] in the setting of trace semantics. The main difficulty is that the pointer structure of traces is no more defined explicitly, but need to be rebuilt from a study of freshness of name pointers (*i.e.* functional variables). Finally, we answer a question asked by Laird at the end of [9] about a possible trace semantics for a language with restricted references, rephrasing the usual notion of *visibility* which characterizes this restriction in the setting of trace semantics.

All missing definitions and proofs can be found in Chapter 4 of the PhD thesis of the author [7].

2 RefML and GroundML

Let us first introduce RefML, a call-by-value λ-calculus with *higher-order references*. The syntax of types τ, values v, terms M and evaluation contexts K of RefML is given in Figure 1.A type is said to be *ground* if it is not equal to $\tau \to \sigma$. So for example $\mathrm{ref}\,(\tau \to \sigma)$ is ground. As usual, $\mathtt{let}\ x\ \mathtt{=}\ N\ \mathtt{in}\ M$ is defined as $(\lambda x.M)N$ and $M; N$ is defined as $(\lambda x.M)N$ with x fresh in M.

Locations live in sets Loc_τ where τ is the type of values they are storing. We define Loc as $\biguplus_\tau \mathrm{Loc}_\tau$ and Loc_ϕ as $\biguplus_{\sigma,\tau} \mathrm{Loc}_{\sigma\to\tau}$. Heaps h are defined as finite partial maps $\mathrm{Loc} \rightharpoonup \mathrm{Val}$ respecting types, *i.e.* $h(l)$ is a closed value of type τ when $l \in \mathrm{Loc}_\tau$. The empty heap is written ε. Adding a new element to a partial map h is written $h \cdot [l \hookrightarrow v]$, and is defined only if $l \notin \mathrm{dom}(h)$. We also define $h[l \hookrightarrow v]$, for $l \in \mathrm{dom}(h)$, as the partial function h' which satisfies $h'(l') = h(l')$ when $l' \neq l$, and $h'(l) = v$. The restriction of a heap h to a set of locations L is written $h_{|L}$. We write h_{fn} for the subheap of h which stores higher-order values A heap is said to be *closed* when, for all $l \in \mathrm{dom}(h)$, if $h(l)$ is itself a location l' then $l' \in \mathrm{dom}(h)$. Taking a set L of locations and h a heap, we define the image of L by h, written $h^*(L)$ as $h^*(L) \stackrel{def}{=} \bigcup_{j\leq 0} h^j(L)$ with $h^0(L) = L$, $h^{j+1}(L) = h(h^j(L)) \cap \mathrm{Loc}$. Using it, we define $\mathbf{Cl}(L)$, the set of minimal closed heaps whose domain contains L, as $\mathbf{Cl}(L) \stackrel{def}{=} \{h \mid h \text{ closed}, \mathrm{dom}(h) = h^*(L)\}$.

$$
\begin{array}{llll}
(K[(\lambda x.M)v], h) & \mapsto (K[M\{v/x\}], h) & (K[\mathrm{ref}\, v], h) & \mapsto (K[l], h \cdot [l \hookrightarrow v]) \\
(K[!l], h) & \mapsto (K[h(l)], h) & (K[l := v], h) & \mapsto (K[()], h[l \hookrightarrow v]) \\
(K[l == l], h) & \mapsto (K[\widehat{1}], h) & (K[l == l'], h) & \mapsto (K[\widehat{0}], h) \\
\multicolumn{4}{l}{(K[\mathtt{case}_{\mathtt{z}}(\widehat{n})(M_1, M_2)], h) \mapsto (K[M_i], h) \quad (i = 1 \text{ if } n = 0, \text{otherwise } i = 2)}
\end{array}
$$

Fig. 2. Operational Semantics of RefML

The small step operational semantics of RefML is defined in Figure 2. We write $M\{v/x\}$ to represent the (capture-free) substitution of x by v in M. This reduction is deterministic, so we suppose that the reduction $(K[\mathrm{ref}\, v], h) \mapsto (K[l], h \cdot [l \hookrightarrow v])$ chooses a location $l \notin \mathrm{dom}(h)$. We also consider the non-deterministic reduction $\mapsto_{nd}$, defined in the same way but for the rule of allocation, which is *s.t.* $(K[\mathrm{ref}\, v], h) \mapsto_{nd} (K[l], h \cdot [l \hookrightarrow v])$ for any $l \notin \mathrm{dom}(h)$.

Then typing judgments are of the form $\Sigma; \Gamma \vdash M : \tau$, where Γ is a variable context and $\Sigma = (l_1, \ldots, l_n)$ is a location context. Notice that we do not need to indicate the types of locations of l in Σ, since the membership $l \in \mathrm{Loc}_\tau$ already gives its type.

In the following, we make a clear distinction between variables of ground types and variables of functional types. Such variable of functional types are called name pointers, which live in the set $\mathbb{P} \subseteq \mathrm{Var}$. Indeed, we deal abstractly with variables of functional types, so we write $\Sigma; \Gamma_g, \Gamma_f \vdash M : \tau$ to distinguish between typing contexts Γ_g, Γ_f containing respectively ground type variables and functional type variables.

From a typing context Γ and a function $\gamma : \mathrm{Var} \rightharpoonup \mathrm{Val}$, we say that γ is a *substitution* on Γ—written $\gamma : \Gamma$—if γ is defined exactly on all the variables occurring in Γ, and $\gamma(x)$ is a value of type τ whenever $(x, \tau) \in \Gamma$. Then, the action of the substitution γ on a term M, defined as $M\{\overrightarrow{\gamma(x_i)/x_i}\}$ with x_i ranging over Γ, is written $\gamma(M)$.

In this article, we also consider GroundML, a restriction of RefML with only full ground references, *i.e.* references which can store integers or other full ground references. It is formally defined as the set of terms of RefML whose type do not contain any subtypes $\mathrm{ref}\,(\tau \to \sigma)$, and which do not contain any subterms of the form $\mathrm{ref}\, M$ with M functional. In GroundML, we cannot define diverging terms anymore via higher-order references, so we rely on special terms Ω_τ for each type τ that always diverge. It is important to have such diverging terms when studying contextual equivalence.

To reason abstractly over name pointers and locations, we use the framework of *nominal sets* [5] over a set of names $\mathbb{A}$. More precisely, all the objects we consider in this article can be seen as nominal sets which is either the set of locations or name pointers, so $\mathbb{A}$ will either be Loc or $\mathbb{P}$. Then two elements t, u of a ($\mathbb{A}$-) nominal set X are said to be *nominally-equivalent*, written $t \sim_{\mathbb{A}} u$ if there exists a finite permutation π over $\mathbb{A}$ *s.t.* $t = \pi \cdot u$ holds. Then a subset X of a nominal set is *nominally closed* if for all $t \in X$ and permutation π over $\mathbb{A}$, $\pi \cdot t \in X$. We write $\nu_{\mathbb{A}}(t)$ for the *support* of an element t of a nominal set X.

3 Trace Semantics

We now introduce a semantics for RefML where denotations of terms are sets of *traces*. It is a variant of the work of Laird [9] more amenable to a comparison with game semantics by taking track of type informations. Traces are used to represent all possible

interactions of terms with contexts. They are generated by an *interactive reduction*, defined as a *labeled transition system*, which generalizes the small-step reduction of Figure 2 by producing *actions* to deal with free functional variables.

To be able to generate all possible executions, we need to keep track of values disclosed to contexts, namely location—so that a context can set arbitrary values in it—or λ-abstraction—so that a context can call it at any time when it takes control back.

Notice that this reduction is history-independent, *i.e.* the reduction of a callback $K[f\ v]$ does not depend on the possible previous occurrences of $K'[f\ v]$ in the reduction. This is due to the fact that our language has references, so that contexts can keep track of the number of times their functional arguments provided to the term are called, and thus give each time a different answer. This corresponds to the fact that strategies for RefML are not innocent.

3.1 Game-Like Definitions

We start introducing traces following the usual presentation of game semantics, mimicking the definitions of the previous section. The notion corresponding to a game move is called here an *action*. Actions are formed over ground values and variables, used to represent higher-order values. These variables, of functional type, are called *opponent* and *player* name pointers. opponent name pointers represent higher-order values provided by contexts (*i.e.* opponent) to terms (*i.e.* player), while it is the opposite for player name pointers. There are four kinds of *basic* actions:

- a question of the term (resp. context) via a name pointer x with argument v (with $x \neq v$), represented by the action $\bar{x}\,\langle v\rangle$ (resp. $x\,\langle v\rangle$),
- an answer by the term (resp. context) of the value v, represented by the action $\langle \bar{v}\rangle$ (resp. $\langle v\rangle$).

A name pointer y appearing as an argument of a player (resp. opponent) question $\bar{x}\,\langle y\rangle$ (resp. $x\,\langle y\rangle$) or in a player (resp. opponent) answer $\langle \bar{y}\rangle$ (resp. $\langle y\rangle$) is called a *player name pointer* (resp. *opponent name pointer*). This means that being an opponent or a player name pointer depends on the action, and is not inherent to the name pointer. The sets of player and opponent name pointers (*i.e.* their supports) of an element X formed by actions are respectively written $\nu^P_{\mathbb{P}}(X)$ and $\nu^O_{\mathbb{P}}(X)$.

Actions a are defined as pairs (a, ξ) of a basic action a and a tag ξ. Tags are words over the alphabet $\{\ell, r, s\}$, where the concatenation is written $\xi \cdot \xi'$ and the empty word is written ε. Such tags are used to indicate to which set (*i.e.* arena) a basic action belongs to. This is useful to avoid the use of disjoint unions (as coproduct) and the corresponding injections $\mathbf{in_l}, \mathbf{in_r}$ which are usually used in game semantics. In our setting, such injections are represented respectively by tags beginning with ℓ and r, while s is used to represent actions corresponding to functions stored in heaps.

Player, Opponent and initial actions are respectively written p, o and i. The labeling of actions (*i.e.* the fact they are *player* or *opponent* and *question* or *answer* actions) is hard-wired, while labeling of moves in game semantics depends on the underlying arena. As we will see, this complicates some definitions (arrow arenas and restrictions of traces to a given arena) where we need to change the labeling of actions.

We define the operation $a^\perp$ as the operation which simply transforms an opponent action into the corresponding player action, and vice-versa (leaving the tag unchanged).

$$\mathfrak{I}_{\mathfrak{A}\otimes\mathfrak{B}} \stackrel{def}{=} \ell\cdot\mathfrak{I}_{\mathfrak{A}} \times r\cdot\mathfrak{I}_{\mathfrak{B}} \qquad \mathfrak{M}_{\mathfrak{A}\otimes\mathfrak{B}} \stackrel{def}{=} \mathfrak{I}_{\mathfrak{A}\otimes\mathfrak{B}} \uplus \ell\cdot(\mathfrak{M}_{\mathfrak{A}}\backslash\mathfrak{I}_{\mathfrak{A}}) \uplus r\cdot(\mathfrak{M}_{\mathfrak{B}}\backslash\mathfrak{I}_{\mathfrak{B}})$$
$$\vdash_{\mathfrak{A}\otimes\mathfrak{B}} \stackrel{def}{=} \{((\ell\cdot \mathrm{i}_{\mathfrak{A}}, r\cdot \mathrm{i}_{\mathfrak{B}}), \ell\cdot a) \mid \mathrm{i}_{\mathfrak{A}} \vdash_{\mathfrak{A}} a\} \uplus \{((\ell\cdot \mathrm{i}_{\mathfrak{A}}, r\cdot \mathrm{i}_{\mathfrak{B}}), r\cdot a) \mid \mathrm{i}_{\mathfrak{B}} \vdash_{\mathfrak{B}} a\}$$
$$\uplus(\ell\cdot \vdash_{\mathfrak{A}\mid(\mathfrak{M}_{\mathfrak{A}}\backslash\mathfrak{I}_{\mathfrak{A}})^2}) \uplus (r\cdot \vdash_{\mathfrak{B}\mid(\mathfrak{M}_{\mathfrak{B}}\backslash\mathfrak{I}_{\mathfrak{B}})^2})$$
$$\mathfrak{I}_{\mathfrak{A}\Rightarrow\mathfrak{B}} \stackrel{def}{=} \{\langle\bar{x}\rangle \mid x\in\mathbb{P}\}$$
$$\mathfrak{M}_{\mathfrak{A}\Rightarrow\mathfrak{B}} \stackrel{def}{=} \mathfrak{I}_{\mathfrak{A}\Rightarrow\mathfrak{B}} \uplus \ell\cdot(\mathfrak{M}_{\mathfrak{A}}\backslash\mathfrak{I}_{\mathfrak{A}})^{\perp} \uplus \{(x\,\langle u\rangle, \ell\cdot\xi) \mid x\in\mathbb{P}, (\langle\bar{u}\rangle,\xi)\in\mathfrak{I}_{\mathfrak{A}}\} \uplus r\cdot\mathfrak{M}_{\mathfrak{B}}$$
$$\vdash_{\mathfrak{A}\Rightarrow\mathfrak{B}} \stackrel{def}{=} \{(\langle\bar{x}\rangle, (x\,\langle u\rangle, \ell\cdot\xi)) \mid x\in\mathbb{P}, (\langle\bar{u}\rangle,\xi)\in\mathfrak{I}_{\mathfrak{A}}\}$$
$$\uplus\{((x\,\langle u\rangle, \ell\cdot\xi), \ell\cdot a) \mid x\in\mathbb{P}, (\langle\bar{u}\rangle,\xi)\in\mathfrak{I}_{\mathfrak{A}}, a\in\mathfrak{M}_{\mathfrak{A}}, (\langle\bar{u}\rangle,\xi)\vdash_{\mathfrak{A}} a\}$$
$$\uplus\{((x\,\langle u\rangle, \ell\cdot\xi), r\cdot \mathrm{i}_{\mathfrak{B}}) \mid x\in\mathbb{P}, (\langle\bar{u}\rangle,\xi)\in\mathfrak{I}_{\mathfrak{A}}\} \uplus (\ell\cdot\vdash_{\mathfrak{A}\mid(\mathfrak{M}_{\mathfrak{A}}\backslash\mathfrak{I}_{\mathfrak{A}})^2})^{\perp} \uplus (r\cdot\vdash_{\mathfrak{B}})$$
$$\mathfrak{I}_{\mathfrak{A}\to\mathfrak{B}} \stackrel{def}{=} \{(?\,\langle u\rangle, \ell\cdot\xi) \mid (\langle\bar{u}\rangle,\xi)\in\mathfrak{I}_{\mathfrak{A}}\} \qquad \mathfrak{M}_{\mathfrak{A}\to\mathfrak{B}} \stackrel{def}{=} \mathfrak{I}_{\mathfrak{A}\to\mathfrak{B}} \uplus \ell\cdot(\mathfrak{M}_{\mathfrak{A}}\backslash\mathfrak{I}_{\mathfrak{A}})^{\perp} \uplus r\cdot\mathfrak{M}_{\mathfrak{B}}$$
$$\vdash_{\mathfrak{A}\to\mathfrak{B}} \stackrel{def}{=} \{((?\,\langle u\rangle, \ell\cdot\xi), r\cdot \mathrm{i}_{\mathfrak{B}}) \mid (\langle\bar{u}\rangle,\xi)\in\mathfrak{I}_{\mathfrak{A}}, \mathrm{i}_{\mathfrak{B}}\in\mathfrak{I}_{\mathfrak{B}}\} \uplus (\ell\cdot\vdash_{\mathfrak{A}\mid(\mathfrak{M}_{\mathfrak{A}}\backslash\mathfrak{I}_{\mathfrak{A}})^2})^{\perp} \uplus (r\cdot\vdash_{\mathfrak{B}})$$
$$\uplus\{((?\,\langle u\rangle, \ell\cdot\xi), \ell\cdot a) \mid (\langle\bar{u}\rangle,\xi)\in\mathfrak{I}_{\mathfrak{A}}, a\in\mathfrak{M}_{\mathfrak{A}}, (\langle\bar{u}\rangle,\xi)\vdash_{\mathfrak{A}} a\}$$

Fig. 3. Definition of compound arenas

It is extended to sets and relations of actions. Then, we introduce the notion of *arenas*, which are simply triples $(\mathfrak{M},\mathfrak{I},\vdash)$ of a set of actions $\mathfrak{M}$, a set of initial actions $\mathfrak{I}\subseteq\mathfrak{M}$ and a justification relation $\vdash\subseteq \mathfrak{M}\times\mathfrak{M}\backslash\mathfrak{I}$. Following the correspondence with game semantics, we define: *value-arenas* (resp. *term-arenas*) as arenas whose initial actions are player answers (resp. opponent question).

From two trace value-arenas $\mathfrak{A},\mathfrak{B}$ we construct the value arenas $\mathfrak{A}\otimes\mathfrak{B}$ and $\mathfrak{A}\Rightarrow\mathfrak{B}$ and the term arena $\mathfrak{A}\to\mathfrak{B}$ in Figure 3. In the definition of $\mathfrak{A}\to\mathfrak{B}$, the symbol $?$ is used as a distinguished name pointer to represent what is interrogated by the initial opponent question. To each type τ, we associate a trace value-arena $[\tau]$ as:

- $[\iota] \stackrel{def}{=} (\mathfrak{M}_\iota,\mathfrak{M}_\iota,\varnothing)$ where $\mathfrak{M}_\iota \stackrel{def}{=} \{\langle\bar{v}\rangle \mid v \text{ a value of type } \iota\}$, $\iota = \mathrm{Unit},\mathrm{Int},\mathrm{ref}\,\tau$
- $[\sigma\to\tau] \stackrel{def}{=} [\sigma]\Rightarrow[\tau]$.

To relate actions to the evolution of the heap, we introduce *actions-with-heap* on an arena $\mathfrak{A}$, *i.e.* pairs (a,h) of an action $a\in\mathfrak{M}_{\mathfrak{A}}$ and a *functional-free heap* h, that is a heap where stored higher-order values are represented by distinct name pointers. An action-with-heap (a,ξ,h) is said to introduce the name pointer x if either a is of the form $\bar{y}\,\langle x\rangle, y\,\langle x\rangle, \langle\bar{x}\rangle$ or $\langle x\rangle$, or if x is in the co-domain of h (written $\mathrm{codom}(h)$). In the latter case, we say that x is l-introduced when $h(l)=x$.

Such actions which l-introduce name pointers, called *ϕ-actions*, correspond to call-backs coming from disclosed locations storing functions. They belong to the set $\mathfrak{M}_\phi$ defined as $\bigcup_{\tau,\tau'}\mathfrak{M}_{[\tau\to\tau']}$. Using it, we define the set $\mathrm{Trace}_{\mathfrak{A}}$ over an arena $\mathfrak{A}$ as the set of sequences T of actions-with-heap on $\mathfrak{M}_{\mathfrak{A}}\uplus(s\cdot\mathfrak{M}_\phi)$ *s.t.* for each name pointer x in T, x is introduced by at most one action-with-heap in T. $\mathrm{Trace}_{\mathfrak{A}}$ can be seen as a nominal set over Loc and $\mathbb{P}$. We write $T'\sqsubseteq T$ when T' is a prefix of T.

We say that a trace $T\in\mathrm{Trace}_{\mathfrak{A}}$ is *justified* if every name pointer x in T is introduced by a previous action in T. Then, we define the *depth* of an action (a,h) in a trace T, written $\mathbf{depth}_T(a,h)$ as the difference between the number of questions and the number of answers of T_1, where $T = T_1\cdot(a,h)\cdot T_2$.

Definition 1. *Let $(a_1,h_1),(a_2,h_2)$ two actions-with-heap* s.t. *(a_1,h_1) appears before (a_2,h_2) in a trace T. We say that (a_1,h_1) justifies (a_2,h_2) when:*

- *a_2 is an answer and a_1 is the latest question of T appearing before a_2* s.t. *$\mathbf{depth}_T(a_2,h_2) = \mathbf{depth}_T(a_1,h_1)+1$,*

– *or* a_2 *is a question* $\bar{x}\langle u\rangle$ *or* $x\langle u\rangle$*, and* (a_1, h_1) *is the first action introducing* x*, so* a_1 *is either equal to* $\langle\bar{x}\rangle, \langle x\rangle, \bar{y}\langle x\rangle, y\langle x\rangle$ *or* $x \in \mathrm{codom}(h_1)$.

In the latter case, we say that x *is* l*-justified when* $h_1(l) = x$*. A question of* T *which does not justify any answer is said to be* pending.

We define the set of *available locations* of a trace T, written $\mathbf{Av}(T)$, as $\mathbf{Av}(\varepsilon) \stackrel{def}{=} \varnothing$ and $\mathbf{Av}(T \cdot (a, h)) \stackrel{def}{=} h^*(\mathbf{Av}(T) \cup \nu_L(a))$. A justified trace over $\mathfrak{A}$ is said to be *legal* if only its first action-with-heap is in $\mathfrak{I}_{\mathfrak{A}}$ and it alternates between player and opponent actions, and is said to be a *play* if it is furthermore *frugal*, *i.e.* for all $T' \cdot (a, h) \sqsubseteq T$, $\mathrm{dom}(h) = \mathbf{Av}(T' \cdot (a, h))$. Using all these definitions, we can finally introduce the notion of *trace-strategy* used to define the denotation of terms.

Definition 2. *A* trace-strategy $\mathfrak{s}$ *over an arena* $\mathfrak{A}$ *is a non-empty set of even-length plays on* $\mathfrak{A}$ s.t.*:*

– *If* $T \cdot (o, h) \cdot (p, h') \in \mathfrak{s}$ *then* $T \in \mathfrak{s}$.
– *If* $T \in \sigma$ *and* $T \sim T'$ *then* $T' \in \mathfrak{s}$.
– *If* $T_1 \cdot (p_1, h_1)$ *and* $T_2 \cdot (p_2, h_2)$ *are in* $\mathfrak{s}$ *and* $T_1 \sim T_2$*, then* $T_1 \cdot (p_1, h_1) \sim T_2 \cdot (p_2, h_2)$.

3.2 A Correspondence between Traces and Plays

There is a direct correspondence between actions introduced in this paper and moves in game semantics defined in [13], where we suppose that injections coming from co-products are also represented by tags. It is obtained by transforming questions $\bar{x}\langle v\rangle$ and $x\langle v\rangle$ into v, transforming answers $\langle\bar{v}\rangle$ and $\langle v\rangle$ into v, and then transforming all remaining name pointers into $\star$, the initial move of game arenas for functional types[1]. The function θ, which performs this two-step translation, transforms actions from an arena $\mathfrak{A}$ to moves to the corresponding game-arena A, leaving tags unchanged. The labeling function is then defined straightforwardly.

We extend this correspondence to justified traces and *well-bracketed* justified sequences of game semantics. More precisely, we first extend the function θ to actions-with-heaps, transforming name pointers stored in heaps into the move $\star$, and then pointwisely from traces to sequences of moves. So we define a function Θ which transforms a justified trace T on an arena $\mathfrak{A}$ to a sequence of moves $\theta(T)$ on the corresponding arena A, and *s.t.* for two actions-with-heaps (a_1, h_1), (a_2, h_2) of T, there is a pointer from $\theta(a_2, h_2)$ to $\theta(a_1, h_1)$ when (a_2, h_2) is justified by (a_1, h_1). Notice that two traces which are $\mathbb{P}$-nominal equivalent give rise to the same sequence of moves.

Extending Θ to sets of traces, it is direct that $\Theta(\mathfrak{s})$ is a game-strategy on an arena A when $\mathfrak{s}$ is a trace-strategy on the corresponding arena $\mathfrak{A}$.

3.3 Interactive Reduction

We now introduce an *interactive reduction* which generates traces from terms, representing their interactions with any possible applicative contexts $K[\bullet_{\tau,\xi}]$, where the

[1] We do not need to transform the symbol "?" since it is automatically removed, appearing only in $?\langle v\rangle$.

Intern $\langle (M,\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \longrightarrow \langle (M',\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h',D\rangle$
(when $(M,h)\mapsto_{nd}(M',h')$)

P-AnsG $\langle (v,\iota,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(\langle\overline{v}\rangle,r\cdot\xi,h'_{|D'})} \langle\overrightarrow{K_i},\gamma',\mathcal{I},h',D'\rangle$
(v of type ι, $\gamma' = \gamma\cdot[\overrightarrow{x_i\hookrightarrow(h(l_i),\tau_i,s)}]$)

P-Ans $\langle (v,\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(\langle\overline{x}\rangle,r\cdot\xi,h'_{|D'})} \langle\overrightarrow{K_i},\gamma',\mathcal{I},h',D'\rangle$
(x fresh, $\gamma' = \gamma\cdot[x\hookrightarrow(v,\tau,r\cdot\xi)]\cdot[\overrightarrow{x_i\hookrightarrow(h(l_i),\tau_i,s)}]$)

P-QuestG $\langle (K[x\,v],\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(\overline{x}\langle v\rangle,l\cdot\xi',h'_{|D'})} \langle (K[\bullet_{\iota,\xi'}],\tau,\xi)\cdot\overrightarrow{K_i},\gamma',\mathcal{I},h',D'\rangle$
($(x,\iota\to\sigma,\xi')\in\mathcal{I}$, v of type ι, $\gamma' = \gamma\cdot[\overrightarrow{x_i\hookrightarrow(h(l_i),\tau_i,s)}]$)

P-Quest $\langle (K[x\,v],\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(\overline{x}\langle y\rangle,l\cdot\xi',h'_{|D'})} \langle (K[\bullet_{\sigma',\xi'}],\tau,\xi)\cdot\overrightarrow{K_i},\gamma',\mathcal{I},h',D'\rangle$
($(x,\sigma\to\sigma',\xi')\in\mathcal{I}$, y fresh, $\gamma' = \gamma\cdot[y\hookrightarrow(v,\sigma,l\cdot\xi')]\cdot[\overrightarrow{x_i\hookrightarrow(h(l_i),\tau_i,s)}]$)

in all P-rules: $D' = \mathrm{discl}(v,h,D)$ and $h' = h[\overrightarrow{l_i\hookrightarrow x_i}]$ with the x_i fresh
where l_i ranges over $\mathrm{dom}(h_{\mathrm{fn}})\cap D'$ with $l_i\in\mathrm{Loc}_{\tau_i}$)

O-AnsG $\langle (K[\bullet_{\iota,\xi'}],\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(\langle v\rangle,r\cdot\xi',h'_{|D'})} \langle (K[v],\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I}',h',D'\rangle$
(v of type ι, $v\notin\mathrm{dom}(h)\cap\overline{D}$, $D' = \mathrm{discl}(v,h',D)$, $\mathcal{I}' = \mathcal{I}\cdot\overrightarrow{(h(l_i),\tau_i,s)}$)

O-Ans $\langle (K[\bullet_{\sigma,\xi'}],\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(\langle y\rangle,r\cdot\xi',h'_{|D'})} \langle (K[y],\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I}',h',D'\rangle$
($D' = \mathrm{discl}(h',D)$, $\mathcal{I}' = \mathcal{I}\cdot(y,\sigma,r\cdot\xi')\cdot\overrightarrow{(h(l_i),\tau_i,s)}$)

O-QuestG $\langle\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(x\langle v\rangle,l\cdot\xi,h'_{|D'})} \langle (u\,v,\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I}',h',D'\rangle$
($\gamma(x) = (u,\iota\to\tau,\xi)$, v of type ι, $v\notin\mathrm{dom}(h)\cap\overline{D}$, $D' = \mathrm{discl}(v,h',D)$, $\mathcal{I}' = \mathcal{I}\cdot\overrightarrow{(h'(l_i),\tau_i,s)}$)

O-Quest $\langle\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(x\langle y\rangle,l\cdot\xi,h'_{|D'})} \langle (u\,y,\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I}',h',D'\rangle$
($\gamma(x) = (u,\sigma\to\tau,\xi)$, $D' = \mathrm{discl}(h',D)$, $\mathcal{I}' = \mathcal{I}\cdot(y,\tau,l\cdot\xi))\cdot\overrightarrow{(h'(l_i),\tau_i,s)}$)

in all O-Rules: l_i ranges over $\mathrm{dom}(h'_{\mathrm{fn}})\cap\mathcal{D}'$ *s.t.* $l_i\in\mathrm{Loc}_{\tau_i}$, y and $h'(l_i)$ fresh,
$h'_{|\overline{D'}} = h_{|\overline{D}}$ and $h'_{|D'}$ is closed and functional-free)

Fig. 4. Definition of the interaction semantics

symbol $\bullet$, representing a "hole" (*i.e.* a pending question), is tagged with a type τ and a tag ξ, representing the type and the arena of the expected answer which will fill the hole. This reduction is defined on "stacks" $(M,\tau,\xi)\cdot\overrightarrow{(K_i,\tau_i,\xi_i)}$ formed by a term M and contexts $\overrightarrow{K_i}$ for *player configurations*, or on stacks $\overrightarrow{(K_i,\tau_i,\xi_i)}$ for *opponent configurations*. Such elements of the stacks comes also with a type τ and a tag ξ. The empty stack is simply written $\Diamond$. When Player provides a higher-order value to Opponent, either via a callback (*i.e.* a question) or directly when reducing to a λ-abstraction (*i.e.* an answer), it is stored in an environment γ, which is a partial map from $\mathbb{P}$ to Val. Then Opponent can interrogate what is stored in γ, by asking a question. Opponent only provides opponent name pointers to represent higher-order values. They are stored in a set $\mathcal{I}\subseteq\mathbb{P}$. They can also be interrogated by Player.

To represent disclosure of locations, we use a set D which grows as the term or the context discloses new locations. To determine which locations are disclosed when a value v is played with a heap h, we define a function $\mathrm{discl}(v,h,D)$ as $h^*(D\cup\{l\})$ if v is a location l, $h^*(D)$ otherwise. We simply write $\mathrm{discl}(h,D)$ for $h^*(D)$.

The interactive reduction is defined in Figure 4 between player and opponent configurations. The rule **Intern** allows us to perform the usual (operational) reduction of terms. Notice that it uses the non-deterministic reduction $\mapsto_{nd}$ rather than the usual $\mapsto$, in order to be exhaustive w.r.t. names of locations created. The rules **P-AnsG** and **P-Ans** represent player answers. If the answer is a ground value, then if it is a location

it is put in D', otherwise it is simply forgotten by Opponent since it has no meaning to interrogate it. Otherwise, it is a higher-order value, which is thus stored in γ'. The rules **P-QuestG** and **P-Quest** represent player questions. That is, Player interrogates an opponent name pointer of $\mathcal{I}$ with a value. Player can also disclose indirectly either new locations or new higher-order values via the already disclosed locations. The new disclosed locations are caught via $\mathrm{discl}(v, h, D)$, while disclosed higher-order values live in $\mathrm{dom}(h_{\mathrm{fn}}) \cap \mathcal{D}'$ (recall that h_{fn} is the subheap of h which stores higher-order values), so that they are replaced by fresh player name pointers in h' and γ is updated consequently. The rules **O-AnsG** and **O-Ans** represent opponent answers. When Opponent answers a location, it cannot be one which is private to Player. This explain the condition "v is ground and not in $\mathrm{dom}(h) \cap \overline{D}$" (where $\overline{D}$ is the complement of D) in the definition of **O-AnsG**. And when it should be a higher-order value, it is simply represented by a *fresh* opponent name pointer. the rules **O-QuestG** and **O-Quest** represent opponent questions. In that case, Opponent adds to the current execution stack a new thread, corresponding to the higher-order values stored in γ. Opponent can also provide new name pointers via the disclosed part of the heap. Those ones live in the disclosed part of h'_{fn}, *i.e.* in $\mathrm{codom}(h'_{\mathrm{fn}|\mathcal{D}'})$. An important point here is that the context can also disclose *indirectly* new locations via the already disclosed ones. This explain the great liberty Opponent has when it extends h to h' with new disclosed locations. It must however satisfy the equation $D' = \mathrm{discl}(v, h', D)$, so that it cannot add as many new (necessarily disclosed) locations as it wants.

This reduction is highly non-deterministic, since we consider the interactions with all possible contexts. Moreover, the choice of name pointers and locations is also non-deterministic (even in the rule **P-Intern** with the use of $\mapsto_{nd}$).

We say that a trace T is *generated* by a configuration C if it can be written as a sequence $(a_1, h_1) \cdots (a_n, h_n)$ of actions-with-heap *s.t.* $C \xrightarrow{(a_1,h_1)} C_1 \xrightarrow{(a_2,h_2)} \ldots \xrightarrow{(a_n,h_n)} C_n$, and we write $C \xrightarrow{T} C_n$. We can see that it is indeed a trace due to the freshness conditions in the rules P-Ans, P-Quest, O-Ans and O-Quest. The set of traces generated by C is written $\mathbf{Tr}(C)$. Notice that such traces are not in general justified, since name-pointers of C are not introduced. As we will see, the initial (opponent question) action is missing. Moreover, $\mathbf{Tr}(C)$ is not nominally-closed: if π is a permutation *s.t.* $\pi(a) \neq a$ for $a \in \nu_{\mathbb{A}}(C)$, then taking $T \in \mathbf{Tr}(C)$, $\pi * T$ is not in general in $\mathbf{Tr}(C)$. This is useful to distinguish sets $\mathbf{Tr}\langle x, \gamma, \mathcal{I}, h, D\rangle$ and $\mathbf{Tr}\langle x', \gamma, \mathcal{I}, h, D\rangle$ for different opponent name pointers $x, x' \in \mathcal{I}$, or to distinguish $\mathbf{Tr}\langle l, \gamma, \mathcal{I}, h, D\rangle$ and $\mathbf{Tr}\langle l', \gamma, \mathcal{I}, h, D\rangle$ for locations $l, l' \in \mathrm{dom}(h)$.

Example. Let us consider the term M_{inc} defined as

$$\texttt{let x} = \ \mathrm{ref}\ \texttt{0 in let f} = \mathrm{ref}\ (\lambda\_.\texttt{x} := !\texttt{x} + \texttt{1})\ \texttt{in}\ \lambda\texttt{g.g f};\texttt{!f}();\texttt{!x}$$

The reader can check that one possible trace of this term starting from the initial configuration $\langle M_{inc}, \varepsilon, \varepsilon, \varepsilon, \varepsilon\rangle$ is

$$(\langle \bar{a}\rangle, \varepsilon) \cdot (a\,\langle b\rangle, \varepsilon) \cdot (\bar{b}\,\langle l\rangle, [l \hookrightarrow c]) \cdot (c\,\langle()\rangle, [l \hookrightarrow c]) \cdot (\langle \bar{()}\rangle, [l \hookrightarrow c]) \cdot$$
$$(\langle()\rangle, [l \hookrightarrow d]) \cdot (\bar{c}\,\langle()\rangle, [l \hookrightarrow d]) \cdot (\langle()\rangle, [l \hookrightarrow d]) \cdot (\langle \bar{1}\rangle, [l \hookrightarrow d]).$$

Intuitively, this trace corresponds to the interaction with the context defined as $\bullet(\lambda\texttt{c.!c}();\texttt{c} := \lambda\_.())$. Notice that the value stored in x is incremented not by the call to $\texttt{!f}()$ in M_{inc}, but by the call made by the context after the disclosure of f via g.

Indeed, the call to $!\mathtt{f}()$ corresponds to the call to $\lambda\_.()$ since the context has modified the function stored in f.

An important point to notice is that the tags of actions of traces generated by the interactive reduction can be inferred knowing just the tags of initial actions:

Lemma 1. *Let (a_1, ξ_1, h_1) an action justified by (a_2, ξ_2, h_2) in a trace T. Then:*
- *If a_2 is an answers, then $\xi_1 = \ell \cdot \xi$ and $\xi_2 = r \cdot \xi$,*
- *If a_2 is a question, then if a_2 is l-justified by a_1, $\xi_2 = \ell \cdot s \cdot \xi_1$, otherwise $\xi_2 = \ell \cdot \xi_1$.*

So in the following, we often omit tags when considering actions and traces.

3.4 Copycat Behavior

When a term provides to the context an opponent name pointer x (*i.e.* a functional variable), the environment γ is extended with $[y \mapsto x]$, with y a fresh player name pointer. Then, when the context interrogates y, this gives rise to a *copycat* behavior. Following [13], a pair of consecutive actions $(a_1, h_1), (a_2, h_2)$ of a trace T is a *copycat pair* when:
- $\theta((a_1, h_1)) = \theta((a_2, h_2))$,
- if (a_1, h_1) is justified by (a'_1, h'_1) then (a_2, h_2) is justified by (a'_2, h'_2) *s.t.* (a'_2, h'_2), (a'_1, h'_1) are consecutive in T, and moreover if (a_1, h_1) is l-justified with $l \in \mathrm{dom}(h'_1)$, then (a_2, h_2) is l-justified so $l \in \mathrm{dom}(h'_2)$.

Notice that a copycat pair $(\langle u_1 \rangle, h_1) \cdot (\langle \bar{u_2} \rangle, h_2))$ will not satisfy $u_1 = u_2$ nor $h_1 = h_2$ when the u_i are name pointers and when the h_i store name pointers, due to the freshness condition of name pointers in the rules of the interactive reduction. This justifies the use of θ in the definition. Such copycat pairs occur frequently in the heap, when the term does not modify a higher-order value stored in a disclosed location. Indeed, only name pointers are stored in disclosed functional part of heaps. And even if what is stored in a location is not modified, the interaction rule refreshes this name pointer. We use the following definition to control this refreshing.

Definition 3. *Let T a legal trace and $T' \sqsubseteq T$, with T' ending with $(a_1, h_1) \cdot (a_2, h_2)$ and $l \in \mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) \cap \mathrm{Loc}_\phi$ a location of functional type, we say that (T, T', l) is a* copycat triple *if for all ϕ-actions (a'_1, h'_1) of T which are hereditarily l-justified by (a_1, h_1) or (a_2, h_2), there exists an action (a'_2, h'_2)* s.t.*:*
- *if a'_1 has the same player as a_1, then $(a'_1, h'_1) \cdot (a'_2, h'_2)$ is a copycat pair of T,*
- *if a'_1 has the same player as a_2, then $(a'_2, h'_2) \cdot (a'_1, h'_1)$ is a copycat pair of T.*

3.5 Interpretation of Terms

Given a term M *s.t.* $\Sigma; \Gamma_g, \Gamma_f \vdash M : \tau$, we define an associated trace strategy. It is generated using the interactive reduction. To do so, we first define the list of opponent name pointers $\mathcal{I}_\xi^{\Gamma_f}$ as $\mathcal{I}_\xi^{\varepsilon} = \varepsilon$ and $\mathcal{I}_\xi^{(x:\tau),\Gamma} \stackrel{def}{=} (x, \tau, \ell \cdot \xi) \cdot \mathcal{I}_{r \cdot \xi}^{\Gamma}$.

Definition 4 (Trace Semantics). *Let M a term* s.t. *$\Sigma; \Gamma_g, \Gamma_f \vdash M : \tau$. We define $[\Sigma; \Gamma_g, \Gamma_f \vdash M : \tau]$ as the set of even-length traces belonging to the nominal closure over* Loc *and $\mathbb{P}$ of*

$$\left\{ \left(? \left\langle \Sigma, \overrightarrow{\gamma_g(x_i)}, \mathcal{I} \right\rangle, \ell, h\right) \cdot \mathbf{Tr}\langle(\gamma_g(M), \tau, \varepsilon), \varepsilon, \mathcal{I}, h, D\rangle \;\middle|\; \begin{array}{l} \gamma_g : \Gamma_g \\ h \in \mathbf{Cl}(\Sigma, \mathrm{codom}(\gamma_g)) \\ \mathrm{codom}(h_{\mathrm{fn}}) \subseteq \mathbb{P} \end{array} \right\}$$

where $\overrightarrow{x_i}$ *ranges overs the variables of* Γ_g, $D = \mathrm{dom}(h)$ *and* $\mathcal{I} = \mathcal{I}_\xi^{\Gamma_f} \cdot \overrightarrow{(h(l_i), \tau_i, s)}$ s.t. l_i *ranges over* $\mathrm{dom}(h_{\mathrm{fn}})$ *and* $l_i \in \mathrm{Loc}_{\tau_i}$.

Recall that h_{fn} is the subheap of h formed by higher-order references. We reason up to nominal equivalence of $\nu_{\mathrm{Loc}}(M)$ (*i.e.* Σ) and $\nu_{\mathbb{P}}(M)$ (*i.e.* Γ_f) so that $[\Sigma; \Gamma_g, \Gamma_f \vdash M : \tau]$ is nominally closed. Moreover, the substitution γ_g of ground variables of M introduces new locations for variables of type $\mathrm{ref}\,\tau$, so we must consider them in h to have a closed heap. Finally, h is functional-free (*i.e.* $\mathrm{codom}(h_{\mathrm{fn}}) \subseteq \mathbb{P}$), so that for any location $l \in \mathrm{Loc}_\phi$, if $h(l)$ or $h'(l)$ is defined, it has to store an opponent name pointer.

Let us define $[\Gamma]$ as $[\tau_1] \otimes \ldots \otimes [\tau_m]$ when $\Gamma = (x_1 : \tau_1) \ldots (x_m : \tau_m)$ and $[\Sigma]$ is defined as $[\mathrm{ref}\,\tau_1] \otimes \ldots \otimes [\mathrm{ref}\,\tau_n]$ when $\Sigma = (l_1, \ldots, l_n : \tau_n)$ with $l_i \in \mathrm{Loc}_{\tau_i}$.

Theorem 1. *Let* M *a term* s.t. $\Sigma; \Gamma_g, \Gamma_f \vdash M : \tau$, *then* $[\Sigma; \Gamma_g, \Gamma_f \vdash M : \tau]$ *is a trace-strategy over the arena* $[\Sigma] \otimes [\Gamma_g, \Gamma_f] \to [\tau]$.

A trace $T \in \mathbf{Tr}\langle M \cdot \overrightarrow{K_i}, \gamma, \mathcal{I}, h, D\rangle$ is said to be *complete* if the number of answers occurring in the trace is greater than its number of questions plus the length of the sequence $\overrightarrow{K_i}$. The set of complete traces of a configuration C is written $\mathbf{comp}(\mathbf{Tr}(C))$.

4 A Correspondence between Trace and Game Denotations

We now prove a formal link between the denotation of a term in trace semantics and in game semantics. The problem is that the definition of $[\Sigma; \Gamma \vdash M : \tau]$ is done operationally, while $[\![\Sigma; \Gamma \vdash M : \tau]\!]$—the game interpretation of terms defined in [13]—is given denotationally, by induction on the typing judgment $\Sigma; \Gamma \vdash M : \tau$. To fill this gap, we show in this section that $[\Sigma; \Gamma \vdash M : \tau]$ can actually be decomposed by similar induction steps on the typing judgment. Using the definition of Θ, which transforms a trace strategy on a $\mathfrak{A}$ into a game strategy of the corresponding game arena A, introduced in Section 3.1, we can state a correspondence between the two semantics.

Theorem 2 (Equivalence of the trace and the game semantics). *Let* M *a term of RefML* s.t. $\Sigma; \Gamma \vdash M : \tau$, *then* $[\![\Sigma; \Gamma \vdash M : \tau]\!]$ *is equal to* $\Theta([\Sigma; \Gamma \vdash M : \tau])$.

Using this correspondence with game semantics, we can import the full abstraction result of [13] to trace semantics. Notice that Laird has already proven this result directly [9], where he needed a complex proof of definability of trace strategy to achieve it. The proof of Theorem 2 goes in four steps:

- we build a category $\mathcal{T}$ whose objects are arenas and morphisms are trace strategies,
- we equip $\mathcal{T}$ with a structure of closed-Freyd category, that is a symmetric premonoidal structure $(\mathcal{T}, I, \otimes)$, a lluf subcategory $\mathcal{T}_{sst}$ of $\mathcal{T}$ for which $\otimes$ is cartesian, and a premonoidal functor $(.)^\dagger$ between $\mathcal{T}_{sst}$ and $\mathcal{T}$, which is identity on objects, *s.t.* for every object $\mathfrak{A}$ of $\mathcal{T}$, the functor $(\_ \otimes \mathfrak{A})^\dagger : \mathcal{T}_{sst} \to \mathcal{T}$ has a right adjoint,
- we show that Θ is a functor from $\mathcal{T}$ to the game category $\mathcal{G}$ which conserve the closed-Freyd structure,

– we prove that the canonical interpretation of terms derived from the closed-Freyd structure is equal to the interpretation of terms, built using the interactive reduction, of Definition 4.

In the following, we sketch the main points of these four steps. Let us first define a composition between trace strategy, as parallel composition plus hidden like in game semantics. following the definition from game semantics, we introduce *interaction traces* over a term-arena $\mathfrak{A} \to \mathfrak{B} \to \mathfrak{C}$ formed from three trace value-arenas $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}$. Given a trace T on $\mathfrak{A} \to \mathfrak{B} \to \mathfrak{C}$, we reason on the restricted traces $T_{|(\mathfrak{A},\mathfrak{B})}$, $T_{|(\mathfrak{B},\mathfrak{C})}$ and $T_{|(\mathfrak{A},\mathfrak{C})}$ which are not in general frugal. So we introduce a function $\mathbf{Frug}$ which removes part of the heap which has not been disclosed in traces, imposing frugality.

Definition 5. *A justified trace T on $\mathfrak{A} \to \mathfrak{B} \to \mathfrak{C}$ is an* interaction trace *if*

– *it is frugal,*
– *$T_{|(\mathfrak{A},\mathfrak{B})}, T_{|(\mathfrak{B},\mathfrak{C})}$ and $T_{|(\mathfrak{A},\mathfrak{C})}$ are legal,*
– *$\mathbf{P}(\mathbf{Frug}(T_{|(\mathfrak{A},\mathfrak{B})})), \mathbf{P}(\mathbf{Frug}(T_{|(\mathfrak{B},\mathfrak{C})})), \mathbf{O}(\mathbf{Frug}(T_{|(\mathfrak{A},\mathfrak{C})}))$ are two by two disjoint, where $\mathbf{P}(T)$ (resp. $\mathbf{O}(T)$) is the set of locations introduced by Player (resp. Opponent),*
– *for each $T' \sqsubseteq T$ ending in $(a,h)\cdot(a',h')$ and $l \in \mathrm{dom}(h')$,*
  - *if a is a player action in X and $l \notin \mathbf{Av}(T_{|X})$ with $X \in \{(\mathfrak{A},\mathfrak{B}),(\mathfrak{B},\mathfrak{C})\}$,*
  - *or a is an opponent action in $(\mathfrak{A},\mathfrak{C})$ and $l \notin \mathbf{Av}(T_{|(\mathfrak{A},\mathfrak{C})})$,*

  then $\theta(h(l)) = \theta(h'(l))$ and, moreover, if $l \in \mathrm{Loc}_\phi$ then $(T_{|X}, T'_{|X}, l)$ are a copycat triple, where X is the respective element of $\{(\mathfrak{A},\mathfrak{B}),(\mathfrak{B},\mathfrak{C}),(\mathfrak{A},\mathfrak{C})\}$.

We now define the parallel composition of two trace strategies $\mathfrak{s}, \mathfrak{t}$, written $\mathfrak{s}||\mathfrak{t}$ as the set of interaction traces $T \in \mathfrak{A} \to \mathfrak{B} \to \mathfrak{C}$ *s.t.* $T_{|_{\mathbf{F}}(\mathfrak{A},\mathfrak{B})} \in \mathfrak{s}$ and $T_{|_{\mathbf{F}}(\mathfrak{B},\mathfrak{C})} \in \mathfrak{t}$.

Definition 6. *Given $\mathfrak{s}, \mathfrak{t}$ two trace-strategies defined respectively on $\mathfrak{A} \to \mathfrak{B}$ and $\mathfrak{B} \to \mathfrak{C}$, we define their* composition*, written $\mathfrak{s};\mathfrak{t}$, as the trace-strategy on $\mathfrak{A} \to \mathfrak{C}$ formed by plays T on $\mathfrak{A} \to \mathfrak{C}$* s.t. *there exists $T' \in \mathfrak{s}||\mathfrak{t}$ with $T = T'_{|_{\mathbf{F}}(\mathfrak{A},\mathfrak{C})}$.*

This composition can be shown to be associative, so that we can define a category $\mathcal{T}$ whose objects are arenas and whose morphisms are trace strategies. It is straightforward to see that Θ is a functor between $\mathcal{T}$ and the game category $\mathcal{G}$ from [13]. Moreover, this composition corresponds to the one coming from the interactive reduction.

Theorem 3. $[\Gamma \vdash N : \sigma]\,;[x : \sigma \vdash M : \tau] = [\Gamma \vdash \texttt{let}\ \ x = N\ \ \texttt{in}\ \ M : \tau]$.

Next, we build a lluf category $\mathcal{T}_{sst}$ of $\mathcal{T}$, in order to get a closed-Freyd category. Morphisms of $\mathcal{T}_{sst}$ are formed by *strongly single threaded* strategies. To define them, we first introduce the notion of *total* strategies, for which traces begin with an opponent question $(?\,\langle u_1\rangle, h_1)$ followed by a player answer $(\langle \bar{u_2}\rangle, h_2)$, without modifying the heap (this last point is controlled using copycat triples from Section 3.4).

For such total strategies, we define the notion of *threads*, which are subtraces which are generated by the opponent questions of the player answer $(\langle \bar{u_1}\rangle, h_1)$. Total strategies which are formed by such frugal threads can be characterized, they are called *strongly single-threaded* strategies Then, we define a function $\mathfrak{L}^{\mathfrak{B}}_{\mathfrak{A},\mathfrak{C}}$ which maps trace-strategies of $(\mathfrak{A} \otimes \mathfrak{B}) \to \mathfrak{C}$ into strongly single-threaded trace-strategy of $\mathfrak{A} \to (\mathfrak{B} \Rightarrow \mathfrak{C})$. From this, we can define the notion of *thread-independent* trace plays. They correspond to plays where there is no interaction between their threads. Thus, following [13],

we can define a "shuffle" operation $(\cdot)^\dagger$ which transforms a strongly single-threaded strategy $\mathfrak{s}$ on $\mathfrak{A}$ into a thread-independent strategy $\mathfrak{s}^\dagger$. This shuffle operation satisfies that for any thread-independent trace strategy $\mathfrak{s}$, $(\gamma(\mathbf{thr}(\mathfrak{s})))^\dagger = \mathfrak{s}$. Then, we can easily check that $[\Sigma; \Gamma \vdash \lambda x.M : \sigma \to \tau]$ is thread-independent. Using it, we can decompose the denotation of λ-abstraction exactly as it is done in game semantics:

Theorem 4. $[\Sigma; \Gamma \vdash \lambda x.M : \sigma \to \tau] = (\mathfrak{L}^{[\sigma]}_{[\Sigma;\Gamma],[\tau]}([\Sigma; \Gamma, x : \sigma \vdash M : \tau]))^\dagger$.

5 Trace Semantics for GroundML

In this section, we refine our trace semantics to handle GroundML, importing the characterizations of terms of this language, in terms of visible strategies [14], into trace semantics.

Definition 7 (View and Visibility). *The* view $\ulcorner T \urcorner$ *of a legal trace* T *on* $\mathfrak{A}$ *is a subsequence of* T *defined by induction:*

- $\ulcorner \varepsilon \urcorner = \varepsilon$ *and* $\ulcorner (\mathrm{i}_{\mathfrak{A}}, h) \urcorner = (\mathrm{i}_{\mathfrak{A}}, h)$,
- $\ulcorner T' \cdot (a, h) \cdot T'' \cdot (a', h') \urcorner = \ulcorner T' \urcorner \cdot (a, h) \cdot (a', h')$ *when* (a', h') *is justified by* (a, h).

A trace T *is* P-visible *(resp.* O-visible*) if for all* $T' \cdot (a, h) \sqsubseteq^{\mathrm{even}} T$ *(resp.* $T' \cdot (a, h) \sqsubseteq^{\mathrm{odd}} T$ *with* (a, h) *a* player *(resp.* opponent*) action, the justifier of* (a, h) *is in* $\ulcorner T' \urcorner$*. A trace strategy is said to be* X-visible *if all its traces are X-visible, for* $X \in \{P, O\}$.

In our setting, justification is defined using freshness of name pointers, so we introduce the notion of *available X-name pointers*, for $X \in \{P, O\}$, to reason on the view.

Definition 8 (Available Name-Pointers). *We define the set of* available opponent *or* player name pointers $\mathbf{Av}_{\mathbf{X}}(T)$ *(*$X \in \{O, P\}$*) inductively as*

- $\mathbf{Av}_{\mathbf{X}}(\varepsilon) \stackrel{def}{=} \varnothing$ *and* $\mathbf{Av}_{\mathbf{X}}((\mathrm{i}, h)) \stackrel{def}{=} \nu^X_{\mathbb{P}}(\mathrm{i})$,
- $\mathbf{Av}_{\mathbf{X}}(T \cdot (a_1, h_1) \cdot T' \cdot (a_2, h_2)) \stackrel{def}{=} \mathbf{Av}_{\mathbf{X}}(T) \cup \nu^X_{\mathbb{P}}(a_2)$ *when* (a_1, h_1) *is justified by* (a_2, h_2).

Notice in the previous definition that we do not need to consider name pointers in heaps, since we consider terms of GroundML, which do not store any functions in heaps, and for which contexts cannot disclose higher-order references. Moreover, in the last clause above, we do not need to consider the name pointers of a_1 since its polarity is opposed as the one of a_2. Next, we link available name pointers to the notion of view:

Lemma 2. *Let* T *a justified trace, then for all name pointers* $x \in \mathbf{Av}_{\mathbf{X}}(T)$ *(*$X \in \{P, O\}$*),* x *is introduced by an action* (a, h) *which appears in* $\ulcorner T \urcorner$.

5.1 Ground-refererences Terms of RefML and P-visible Strategies

The characterization of trace-strategies coming from terms of GroundML is given by the following theorem.

Theorem 5. *Let* M *a term of GroundML* s.t. $\Sigma; \Gamma \vdash M : \tau$. *Then* $[\Sigma; \Gamma \vdash M : \tau]$ *is a P-visible trace strategy.*

$$\textbf{Intern}\quad \langle (M,\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \longrightarrow \langle (M',\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h',D\rangle$$
$$(\text{when } (M,h)\mapsto_{nd}(M',h'))$$

$$\textbf{P-AnsG}\quad \langle (v,\iota,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(\langle\bar{v}\rangle,r\cdot\xi,h'_{|D})} \langle\overrightarrow{K_i},\gamma',\mathcal{I},h,D',\mathcal{A}\rangle$$
$$(v \text{ of type } \iota,\ D'=\mathrm{discl}(v,h,D))$$

$$\textbf{P-Ans}\quad \langle (v,\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(\langle\bar{x}\rangle,r\cdot\xi,h_{|D})} \langle\overrightarrow{K_i},\gamma',\mathcal{I},h,D,x\cdot\mathcal{A}\rangle$$
$$(x \text{ fresh},\ \gamma'=\gamma\cdot[x\hookrightarrow(v,\tau,r\cdot\xi)])$$

$$\textbf{P-QuestG}\quad \langle (K[x\ v],\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(\bar{x}\langle v\rangle,\ell\cdot\xi',h_{|D'})} \langle (K[\bullet_{\iota,\xi'}],\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D',\mathcal{A}'\rangle$$
$$((x,\iota\to\sigma,\xi',\mathcal{A}')\in\mathcal{I},\ v \text{ of type } \iota,\ D'=\mathrm{discl}(v,h,D))$$

$$\textbf{P-Quest}\quad \langle (K[x\ v],\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D\rangle \xrightarrow{(\bar{x}\langle y\rangle,\ell\cdot\xi',h_{|D})} \langle (K[\bullet_{\sigma',\xi'}],\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma',\mathcal{I},h,D,y\cdot\mathcal{A}'\rangle$$
$$((x,\sigma\to\sigma',\xi',\mathcal{A}')\in\mathcal{I},\ y \text{ fresh},\ \gamma'=\gamma\cdot[y\hookrightarrow(v,\sigma,\ell\cdot\xi')])$$

$$\textbf{O-AnsG}\quad \langle (K[\bullet_{\iota,\xi'}],\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D,\mathcal{A}'\rangle \xrightarrow{(\langle v\rangle,r\cdot\xi',h'_{|D'})} \langle (K[v],\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h',D'\rangle$$
$$(v \text{ of type } \iota,\ v\notin \mathrm{dom}(h)\cap\overline{D})$$

$$\textbf{O-Ans}\quad \langle (K[\bullet_{\sigma,\xi'}],\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I},h,D,\mathcal{A}'\rangle \xrightarrow{(\langle x\rangle,r\cdot\xi',h'_{|D'})} \langle (K[x],\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I}',h',D'\rangle$$
$$(x \text{ fresh},\ \mathcal{I}'=\mathcal{I}\cdot(x,\sigma,r\cdot\xi',\mathcal{A}'))$$

$$\textbf{O-QuestG}\quad \langle\overrightarrow{K_i},\gamma,\mathcal{I},h,D,\mathcal{A}\rangle \xrightarrow{(x\langle v\rangle,\ell\cdot\xi,h'_{|D'})} \langle (u\ v,\tau,\xi,\mathcal{A})\cdot\overrightarrow{K_i},\gamma,\mathcal{I}',h',D'\rangle$$
$$(x\in\mathcal{A},\gamma(x)=(u,\iota\to\tau,\xi),\ v \text{ of type } \iota \text{ and not in } \mathrm{dom}(h)\cap\overline{D})$$

$$\textbf{O-Quest}\quad \langle\overrightarrow{K_i},\gamma,\mathcal{I},h,D,\mathcal{A}\rangle \xrightarrow{(x\langle y\rangle,\ell\cdot\xi,h'_{|D'})} \langle (u\ y,\tau,\xi)\cdot\overrightarrow{K_i},\gamma,\mathcal{I}',h',D'\rangle$$
$$(x\in\mathcal{A},\gamma(x)=(u,\sigma\to\tau,\xi),\ y \text{ fresh},\ \mathcal{I}'=\mathcal{I}\cdot(y,\tau,\ell\cdot\xi,\mathcal{A}))$$
$$\textbf{in all O-Rules: } D'=\mathrm{discl}(v,h',D),\ h'_{|\overline{D'}}=h_{|\overline{D}} \text{ and } h'_{|D'} \text{ is closed})$$

Fig. 5. Definition of the interaction semantics for GroundML

To conduct the proof, we need to analyze the structure of traces more closely. We first notice a crucial property of the interactive reduction of such terms, that when reducing (M,h) to (M',h'), we know that the name pointers contained in M' are also in M. Using this property, we prove that a term M' appearing in the interactive reduction of M via a trace T only contains name pointers from $\mathbf{Av_O}(T)$.

Lemma 3. *Let M a term of GroundML* s.t. $\Sigma;\Gamma\vdash M:\tau$, *and* $((a_0,h_0)\cdot T\cdot(a,h))\in[\Sigma;\Gamma\vdash M:\tau]$ s.t. $\langle M,\gamma,\mathcal{I},h_0,D\rangle \xrightarrow{T} \langle M'\cdot\overrightarrow{K_i},\gamma',\mathcal{I}',h',D'\rangle$ *Then* $\nu^O_{\mathbb{P}}(M')\subseteq\mathbf{Av_O}((a_0,h_0)\cdot T)$.

From this lemma we get directly the following corollary:

Corollary 1. *Let M a term of GroundML* s.t. $\Sigma;\Gamma\vdash M:\tau$, *and* $T\cdot(\bar{x}\langle u\rangle,h)\in[\Sigma;\Gamma\vdash M:\tau]$. *Then* $x\in\mathbf{Av_O}(T)$.

We can finally prove Theorem 5 using the conjunction of Lemma 2 and Corollary 1.

5.2 Full Abstraction for GroundML

We have seen in Section 5.1 that terms of GroundML give rise to P-visible traces. However, the trace semantics of Section 3.3 is not fully abstract, since there are still traces which are not generated by an interaction between contexts and terms of GroundML. To get full abstraction, we need to constrain traces to be O-visible by modifying the interactive reduction to control the scope of name pointers *s.t.* only pointers appearing in the

view of an action are available. We present the modified interactive reduction in Figure 5. First notice that, since we do not have higher-order references in our language, we do not need to extend $\mathcal{I}$ and γ with pointers representing functions stored in the disclosed part of the heap. Then, we keep track of available player name pointers, represented by a set $\mathcal{A}$, in different places of the configuration:

- in opponent configurations $\langle \overrightarrow{K_i}, \gamma, \mathcal{I}, h, D, \mathcal{A} \rangle$, representing the current available player name pointers;
- within each element $(M, \tau, \xi, \mathcal{A})$ or $(K[\bullet_{\tau,\xi}], \sigma, \xi', \mathcal{A})$ of the execution stack, representing the player name pointers available when Opponent add this element to the execution stack via a question;
- within each opponent name pointers $x \in \mathcal{I}$, representing the player name pointers available when x has been introduced by an opponent action.

Using this set of available player name pointers, we can control the questions Opponent can interrogate, as shown by the condition $x \in \mathcal{A}$ in the rules **O-QuestG** and **O-Quest**. This idea is formalized in the following lemma:

Lemma 4. *For* $(\mathrm{i}, h_0) \cdot T \in [\Sigma; \Gamma_f \vdash M : \tau]$ s.t. $\langle (M, \tau, \xi, \varnothing), \gamma, \mathcal{I}, h, \mathcal{D} \rangle \xrightarrow{T} \langle \overrightarrow{K_i}, \gamma', , \mathcal{I}', h', \mathcal{D}' \rangle \mathcal{A}$ *we have* $\mathbf{Av_P}(T) = \mathcal{A}$.

Using our restricted interactive reduction, we define the ground interpretation of a judgment $[\Sigma; \Gamma \vdash M : \tau]_G$ in the same way than in Definition 4, with the empty set of pointers associated to M in the execution stack. From Lemma 4, we can deduce that the ground interpretation always gives rise to visible strategies.

Theorem 6. *Let* M *a term of GroundML* s.t. $\Sigma; \Gamma \vdash M : \tau$. *Then* $[\Sigma; \Gamma \vdash M : \tau]_G$ *is a visible strategy (*i.e. *both P-visible and O-visible).*

It is straightforward to see that the notion of visibility introduced here corresponds exactly to the usual notion of visibility of game semantics. So we can import the full abstraction result of [14] into our trace semantics of GroundML.

6 Discussion and Future Work

We would like to extend our trace semantics to a polymorphic language, where the mechanism to represent the arenas an action belongs to in terms of tags would become crucial, as in [11]. It would also be interesting to extend the characterization of fragments of RefML in trace semantics by dealing with the restriction to integer references, namely RedML, as in [15]. We believe that our interactive reduction defined for GroundML can be restricted to give a fully abstract model of RedML, using similar restrictions on the use of name pointers in locations.

As we have said, the well-bracketing condition is hard-wired in the definition of justified traces. To remove it, we would need to specify which question an answer is answering. One possibility to do that would be to use the work of Gabbay and Ghica [4], which uses nominal sets to represent strategies. In fact, our work is halfway to them: the way they name questions, and the freshness condition they impose, seems similar to our use of name-pointers and the nominal reasoning we perform on them. It should thus

be possible to use their work to also give fresh name to answers actions. This would allow us to study languages where we need semantically to remove the well-bracketing condition, namely languages with control operators like `call/cc` or exceptions.

Finally, a correspondence between trace and game models has been previously built by Laird [8] in a different setting, the asynchronous π-calculus (so without references). Levy and Staton [12] have also recently build an abstract categorical setting to study such correspondences. It would be interesting if our work could be spelt out in their framework, and therefore give a high-level categorical meaning to our construction.

References

1. Abramsky, S., Ghica, D., Murawski, A., Ong, L., Stark, I.: Nominal games and full abstraction for the nu-calculus. In: Proceedings of LICS 2004, pp. 150–159. IEEE (2004)
2. Abramsky, S., Honda, K., McCusker, G.: A fully abstract game semantics for general references. In: Proceedings of LICS 1998, pp. 334–344. IEEE (1998)
3. Abramsky, S., Jagadeesan, R., Malacaria, P.: Full abstraction for PCF. Information and Computation 163(2), 409–470 (2000)
4. Gabbay, M., Ghica, D.: Game semantics in the nominal model. In: Proceedings of the 28th Conference on the Mathematical Foundations of Programming Semantics (MFPS 2012). Electronic Notes in Theoretical Computer Science, vol. 286, pp. 173–189. Elsevier (2012)
5. Gabbay, M., Pitts, A.: A new approach to abstract syntax with variable binding. Formal Aspects of computing 13(3-5), 341–363 (2002)
6. Hyland, M., Ong, L.: On full abstraction for PCF: I, II and III. Information and Computation 163(2), 285–408 (2000)
7. Jaber, G.: A Logical Study of Program Equivalence. PhD thesis, École des Mines de Nantes (2014)
8. Laird, J.: A game semantics of the asynchronous *p*-calculus. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 51–65. Springer, Heidelberg (2005)
9. Laird, J.: A fully abstract trace semantics for general references. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 667–679. Springer, Heidelberg (2007)
10. Laird, J.: A game semantics of names and pointers. Annals of Pure and Applied Logic 151(2), 151–169 (2008)
11. Laird, J.: Game semantics for call-by-value polymorphism. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6199, pp. 187–198. Springer, Heidelberg (2010)
12. Levy, P., Staton, S.: Transition systems over games. In: Proceedings of LICS 2014, pp. 64:1–64:10 (2014)
13. Murawski, A., Tzevelekos, N.: Game semantics for good general references. In: Proceedings of LICS 2011, pp. 75–84. IEEE (2011)
14. Murawski, A., Tzevelekos, N.: Algorithmic games for full ground references. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part II. LNCS, vol. 7392, pp. 312–324. Springer, Heidelberg (2012)
15. Murawski, A., Tzevelekos, N.: Full abstraction for reduced ml. Annals of Pure and Applied Logic (2013)
16. Power, J., Thielecke, H.: Closed freyd- and κ-categories. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 625–634. Springer, Heidelberg (1999)