Game Semantics for Interface Middleweight Java *

Andrzej S. Murawski
DIMAP and Department of Computer Science
University of Warwick

Nikos Tzevelekos
School of Electronic Engineering and Computer Science
Queen Mary, University of London

Abstract

We consider an object calculus in which open terms interact with the environment through interfaces. The calculus is intended to capture the essence of contextual interactions of Middleweight Java code. Using game semantics, we provide fully abstract models for the induced notions of contextual approximation and equivalence. These are the first denotational models of this kind.

Categories and Subject Descriptors D.3.1 [*Formal Definitions and Theory*]: Semantics; F.3.2 [*Semantics of Programming Languages*]: Denotational semantics

Keywords Full Abstraction, Game Semantics, Contextual Equivalence, Java

1. Introduction

Denotational semantics is charged with the construction of mathematical universes (denotations) that capture program behaviour. It concentrates on compositional, syntax-independent modelling with the aim of illuminating the structure of computation and facilitating reasoning about programs. Many developments in denotational semantics have been driven by the quest for full abstraction [21]: a model is *fully abstract* if the interpretations of two programs are the same precisely when the programs behave in the same way (i.e. are contextually equivalent). A faithful correspondence like this opens the path to a broad range of applications, such as compiler optimisation and program transformation, in which the preservation of semantics is of paramount importance.

Recent years have seen *game semantics* emerge as a robust denotational paradigm [4, 6, 12]. It has been used to construct the first fully abstract models for a wide spectrum of programming languages, previously out of reach of denotational semantics. Game semantics models computation as an exchange of moves between two players, representing respectively the program and its computational environment. Accordingly, a program is interpreted as a strategy in a game corresponding to its type. Intuitively, the plays that game semantics generates constitute the observable patterns that a program produces when interacting with its environment, and this is what underlies the full abstraction results. Game semantics is compositional: the strategy corresponding to a compound program phrase is obtained by canonical combinations of those corresponding to its sub-phrases. An important advance in game semantics was the development of nominal games [3, 17, 26], which underpinned full abstraction results for languages with dynamic generative behaviours, such as the ν-calculus [3], higher-order concurrency [18] and ML references [24]. A distinctive feature of nominal game models is the presence of names (e.g. memory locations, references names) in game moves, often along with some abstraction of the store.

The aim of the present paper is to extend the range of the game approach towards real-life programming languages, by focussing on Java-style objects. To that end, we define an imperative object calculus, called Interface Middleweight Java (IMJ), intended to capture contextual interactions of code written in Middleweight Java (MJ) [9], as specified by interfaces with inheritance. We present both equational (contextual equivalence) and inequational (contextual approximation) full abstraction results for the language. To the best of our knowledge, these are the first denotational models of this kind.

Related Work While the operational semantics of Java has been researched extensively [7], there have been relatively few results regarding its denotational semantics. More generally, most existing models of object-oriented languages, such as [8, 15], have been based on global state and consequently could not be fully abstract.

On the other hand, contextual equivalence in Java-like languages has been studied successfully using operational approaches such as trace semantics [2, 13, 14] and environmental bisimulation [16]. The trace-based approaches are closest to ours and the three papers listed also provide characterizations of contextual equivalence. The main difference is that traces are derived operationally through a carefully designed labelled transition system and, thus, do not admit an immediate compositional description in the style of denotational semantics.

However, similarities between traces and plays in game semantics indicate a deeper correspondence between the two areas, which also manifested itself in other cases, e.g. [20] vs [19]. At the time of writing, there is no general methodology for moving smoothly between the two approaches, but we believe that there is scope for unifying the two fields in the not so distant future.

In comparison to other game models, ours has quite lightweight structure. For the most part, playing consists of calling the opponent's methods and returning results to calls made by the opponent. In particular, there are no justification pointers between moves. This can be attributed to the fact that Java does not feature first-class higher-order functions and that methods in Java objects cannot be updated. On the other hand, the absence of pointers makes definitions of simple notions, such as well-bracketing, less direct, since the dependencies between moves are not given explicitly any

* Research supported by the Engineering and Physical Sciences Research Council (EP/J019577/1) and a Royal Academy of Engineering Research Fellowship (Tzevelekos).

POPL'14, January 22–24, 2014, San Diego, CA, USA.

ACM 978-1-4503-2544-8/14/01…$15.00.
http://dx.doi.org/10.1145/2535838

$$\frac{}{\Delta|\Gamma \vdash x : \theta}\ {}^{(x:\theta)\in\Gamma} \qquad \frac{}{\Delta|\Gamma \vdash a : \mathcal{I}}\ {}^{(a:\mathcal{I})\in\Gamma} \qquad \frac{}{\Delta|\Gamma \vdash \mathsf{skip} : \mathsf{void}} \qquad \frac{}{\Delta|\Gamma \vdash \mathsf{null} : \mathcal{I}}\ {}^{\mathcal{I}\in\mathsf{dom}(\Delta)} \qquad \frac{}{\Delta|\Gamma \vdash i : \mathsf{int}}$$

$$\frac{\Delta|\Gamma \vdash M : \mathsf{int} \quad \Delta|\Gamma \vdash M' : \mathsf{int}}{\Delta|\Gamma \vdash M \oplus M' : \mathsf{int}} \qquad \frac{\Delta|\Gamma, x : \theta' \vdash M : \theta \quad \Delta|\Gamma \vdash M' : \theta'}{\Delta|\Gamma \vdash \mathsf{let}\, x = M' \,\mathsf{in}\, M : \theta} \qquad \frac{\Delta|\Gamma \vdash M : \mathcal{I} \quad \Delta|\Gamma \vdash M' : \mathcal{I}}{\Delta|\Gamma \vdash M = M' : \mathsf{int}} \qquad \frac{\Delta|\Gamma \vdash M : \mathcal{I}'}{\Delta|\Gamma \vdash (\mathcal{I})M : \mathcal{I}}\ {}^{\Delta\vdash\mathcal{I}\le\mathcal{I}'}_{\vee\,\Delta\vdash\mathcal{I}'\le\mathcal{I}}$$

$$\frac{\Delta|\Gamma \vdash M : \mathsf{int} \quad \Delta|\Gamma \vdash M', M'' : \theta}{\Delta|\Gamma \vdash \mathsf{if}\, M \,\mathsf{then}\, M' \,\mathsf{else}\, M'' : \theta} \qquad \frac{\Delta|\Gamma, x : \mathcal{I} \vdash \mathcal{M} : \Theta}{\Delta|\Gamma \vdash \mathsf{new}(x : \mathcal{I}; \mathcal{M}) : \mathcal{I}}\ {}^{\Delta(\mathcal{I})\upharpoonright Meths = \Theta} \qquad \frac{\Delta|\Gamma \vdash M : \mathcal{I} \quad \Delta|\Gamma \vdash M' : \theta}{\Delta|\Gamma \vdash M.\mathsf{f} := M' : \mathsf{void}}\ {}^{\Delta(\mathcal{I}).\mathsf{f}=\theta}$$

$$\frac{\Delta|\Gamma \vdash M : \mathcal{I}}{\Delta|\Gamma \vdash M.\mathsf{f} : \theta}\ {}^{\Delta(\mathcal{I}).\mathsf{f}=\theta} \qquad \frac{\Delta|\Gamma \vdash M : \mathcal{I} \quad \bigwedge_{i=1}^{n}(\Delta|\Gamma \vdash M_i : \theta_i)}{\Delta|\Gamma \vdash M.\mathsf{m}(M_1, \cdots, M_n) : \theta}\ {}^{\Delta(\mathcal{I}).\mathsf{m}=\vec{\theta}\to\theta} \qquad \frac{\bigwedge_{i=1}^{n}(\Delta|\Gamma \uplus \{\vec{x}_i : \vec{\theta}_i\} \vdash M_i : \theta_i)}{\Delta|\Gamma \vdash \mathcal{M} : \Theta}\ {}^{\Theta=\{\mathsf{m}_i:\vec{\theta}_i\to\theta_i \mid 1\le i\le n\}}_{\mathcal{M}=\{\mathsf{m}_i:\lambda\vec{x}_i.M_i \mid 1\le i\le n\}}$$

Figure 1. Typing rules for IMJ terms and method-set implementations

more and need to be inferred from plays. The latter renders strategy composition non-standard. Because it is impossible to determine statically to which arena a move belongs, the switching conditions (cf. [6]) governing interactions become crucial for determining the strategy responsible for each move. Finally, it is worth noting that traditional copycat links are by definition excluded from our setting: a call/return move for a given object cannot be copycatted by the other player, as the move has a fixed polarity, determined by the ownership of the object. In fact, identity strategies contain plays of length at most two!

Further Directions In future work, we would like to look for automata-theoretic representations of fragments of our model in order to use them as a foundation for a program verification tool for Java programs. Our aim is to take advantage of the latest developments in automata theory over infinite alphabets [10], and fresh-register automata in particular [23, 27], to account for the nominal features of the model.

2. The language IMJ

We introduce an imperative object calculus, called Interface Middleweight Java (IMJ), in which objects are typed using interfaces. The calculus is a stripped down version of Middleweight Java (MJ), expressive enough to expose the interactions of MJ-style objects with the environment.

Definition 1. Let $Ints$, $Flds$ and $Meths$ be sets of *interface*, *field* and *method* identifiers. We range over them respectively by $\mathcal{I}$, f, m and variants. The ***types*** θ of IMJ are given below, where $\vec{\theta}$ stands for a sequence $\theta_1, ..., \theta_n$ of types (for any n). An ***interface definition*** Θ is a finite set of typed fields and methods. An ***interface table*** Δ is a finite assignment of interface definitions to interface identifiers.

$$\begin{aligned} Types \ni\ \theta\ &::=\ \mathsf{void} \mid \mathsf{int} \mid \mathcal{I} \\ IDfns \ni\ \Theta\ &::=\ \emptyset \mid (\mathsf{f} : \theta), \Theta \mid (\mathsf{m} : \vec{\theta} \to \theta), \Theta \\ ITbls \ni\ \Delta\ &::=\ \emptyset \mid (\mathcal{I} : \Theta), \Delta \mid (\mathcal{I}\langle\mathcal{I}\rangle : \Theta), \Delta \end{aligned}$$

We write $\mathcal{I}\langle\mathcal{I}'\rangle : \Theta$ for *interface extension*: interface $\mathcal{I}$ extends $\mathcal{I}'$ with fields and methods from Θ. We stipulate that the extension relation must not lead to circular dependencies. Moreover, each identifier f, m can appear at most once in each Θ, and each $\mathcal{I}$ can be defined at most once in Δ (i.e. there is at most one element of Δ of the form $\mathcal{I} : \Theta$ or $\mathcal{I}\langle\mathcal{I}'\rangle : \Theta$). Thus, each Θ can be seen as a finite partial function $\Theta : (Flds \cup Meths) \rightharpoonup Types^*$. We write $\Theta.\mathsf{f}$ for $\Theta(\mathsf{f})$ and $\Theta.\mathsf{m}$ for $\Theta(\mathsf{m})$. Similarly, Δ defines a partial function $\Delta : Ints \rightharpoonup IDfns$ given by

$$\Delta(\mathcal{I}) = \begin{cases} \Theta & (\mathcal{I} : \Theta) \in \Delta \\ \Delta(\mathcal{I}') \cup \Theta & (\mathcal{I}\langle\mathcal{I}'\rangle : \Theta) \in \Delta \\ \text{undefined} & \text{otherwise} \end{cases}$$

An interface table Δ is ***well-formed*** if, for all interface types $\mathcal{I}, \mathcal{I}'$:

- if $\mathcal{I}'$ appears in $\Delta(\mathcal{I})$ then $\mathcal{I}' \in \mathsf{dom}(\Delta)$,
- if $(\mathcal{I}\langle\mathcal{I}'\rangle : \Theta) \in \Delta$ then $\mathsf{dom}(\Delta(\mathcal{I}')) \cap \mathsf{dom}(\Theta) = \emptyset$.

Henceforth we assume that interface tables are well-formed. Interface extensions yield a subtyping relation. Given a table Δ, we define $\Delta \vdash \theta_1 \le \theta_2$ by the following rules.

$$\frac{}{(\mathcal{I}\langle\mathcal{I}'\rangle : \Theta), \Delta \vdash \mathcal{I} \le \mathcal{I}'} \qquad \frac{}{\Delta \vdash \theta \le \theta} \qquad \frac{\Delta \vdash \theta_1 \le \theta_2 \quad \Delta \vdash \theta_2 \le \theta_3}{\Delta \vdash \theta_1 \le \theta_3}$$

We might omit Δ from subtyping judgements for economy.

Definition 2. Let $\mathbb{A}$ be a countably infinite set of ***object names***, which we range over by a and variants. IMJ ***terms*** are listed below, where we let x range over a set of *variables* $Vars$, and i over $\mathbb{Z}$. Moreover, $\oplus$ is selected from some set of binary numeric operations. $\mathcal{M}$ is a ***method-set implementation***. Again, we stipulate that each m appear in each $\mathcal{M}$ at most once.

$$\begin{aligned} M\ ::=\ & x \mid a \mid \mathsf{skip} \mid \mathsf{null} \mid i \mid M \oplus M \mid \mathsf{let}\, x = M \,\mathsf{in}\, M \\ & \mid M = M \mid \mathsf{if}\, M \,\mathsf{then}\, M \,\mathsf{else}\, M \mid (\mathcal{I})M \mid \mathsf{new}(x : \mathcal{I}; \mathcal{M}) \\ & \mid M.\mathsf{f} \mid M.\mathsf{f} := M \mid M.\mathsf{m}(\overrightarrow{M}) \end{aligned}$$

$$MImps \ni\ \mathcal{M}\ ::=\ \emptyset \mid (\mathsf{m} : \lambda\vec{x}.M), \mathcal{M}$$

The terms are typed in contexts comprising an interface table Δ and a variable context $\Gamma = \{x_1 : \theta_1, \cdots, x_n : \theta_n\} \cup \{a_1 : \mathcal{I}_1, \cdots, a_m : \mathcal{I}_m\}$ such that any interface in Γ occurs in $\mathsf{dom}(\Delta)$. The typing rules are given in Figure 1.

For the operational semantics, we define the sets of ***term values***, ***heap configurations*** and ***states*** by:

$$\begin{aligned} TVals \ni\ v\ &::=\ \mathsf{skip} \mid i \mid \mathsf{null} \mid a \\ HCnfs \ni\ V\ &::=\ \emptyset \mid (\mathsf{f} : v), V \\ States \ni\ S\ &:\ \mathbb{A} \rightharpoonup Ints \times (HCnfs \times MImps) \end{aligned}$$

If $S(a) = (\mathcal{I}, (V, \mathcal{M}))$ then we write $S(a) : \mathcal{I}$, while $S(a).\mathsf{f}$ and $S(a).\mathsf{m}$ stand for $V.\mathsf{f}$ and $\mathcal{M}.\mathsf{m}$ respectively, for each f and m.

Given an interface table Δ such that $\mathcal{I} \in \mathsf{dom}(\Delta)$, we let the default heap configuration of type $\mathcal{I}$ be

$$V_{\mathcal{I}} = \{\mathsf{f} : v_\theta \mid \Delta(\mathcal{I}).\mathsf{f} = \theta\},$$

where $v_{\mathsf{void}} = \mathsf{skip}$, $v_{\mathsf{int}} = 0$ and $v_{\mathcal{I}} = \mathsf{null}$. The operational semantics of IMJ is given by means of a small-step transition relation

$(S, i \oplus i') \longrightarrow (S, j),\ \text{if}\ j = i \oplus i'$ $\quad (S, \mathsf{let}\, x = v\, \mathsf{in}\, M) \longrightarrow (S, M[v/x])$ $\quad (S, (\mathcal{I})\mathsf{null}) \longrightarrow (S, \mathsf{null})$

$(S, \mathsf{if}\, 0\, \mathsf{then}\, M\, \mathsf{else}\, M') \longrightarrow (S, M')$ $\quad (S, \mathsf{if}\, 1\, \mathsf{then}\, M\, \mathsf{else}\, M') \longrightarrow (S, M)$ $\quad (S, a = a) \longrightarrow (S, 1)$

$(S, (\mathcal{I})a) \longrightarrow (S, a),\ \text{if}\ S(a) : \mathcal{I}' \wedge \mathcal{I}' \leq \mathcal{I}$ $\quad (S, a = a') \longrightarrow (S, 0),\ \text{if}\ a \neq a'$ $\quad (S, a.\mathsf{f}) \longrightarrow (S, S(a).\mathsf{f})$

$(S, \mathsf{new}(x : \mathcal{I}; \mathcal{M})) \longrightarrow (S \uplus \{(a, \mathcal{I}, (V_{\mathcal{I}}, \mathcal{M}[a/x]))\}, a)$ $\quad (S, a.\mathsf{m}(\vec{v})) \longrightarrow (S, M[\vec{v}/\vec{x}]),\ \text{if}\ S(a).\mathsf{m} = \lambda\vec{x}.M$

$(S, a.\mathsf{f} := v) \longrightarrow (S[a \mapsto (\mathcal{I}, (V[\mathsf{f} \mapsto v], \mathcal{M})], \mathsf{skip}),\ \text{if}\ S(a) = (\mathcal{I}, (V, \mathcal{M}))$

$(S, E[M]) \longrightarrow (S', E[M']),\ \text{if}\ (S, M) \longrightarrow (S', M')$

Figure 2. Operational semantics of IMJ.

between terms-in-state, presented in Figure 2. The transition relation uses *evaluation contexts* E that are defined as follows.

$$\begin{aligned} E ::= \ & \mathsf{let}\, x = \_\, \mathsf{in}\, M \mid \_ \oplus M \mid i \oplus \_ \mid \_ = M \mid a = \_ \\ & \mid \mathsf{if}\, \_\, \mathsf{then}\, M\, \mathsf{else}\, M' \mid (\mathcal{I})\_ \mid \_.\mathsf{f} \mid \_.\mathsf{f} := M \mid a.\mathsf{f} := \_ \\ & \mid \_.\mathsf{m}(\overrightarrow{M}) \mid a.\mathsf{m}(v_1, \cdots, v_i, \_, M_{i+2}, \cdots, M_n) \end{aligned}$$

Given $\Delta|\emptyset \vdash M : \mathsf{void}$, we write $M \Downarrow$ if there exists S such that $(\emptyset, M) \longrightarrow^* (S, \mathsf{skip})$.

Definition 3. Given $\Delta|\Gamma \vdash M_i : \theta\ (i = 1, 2)$, we shall say that $\Delta|\Gamma \vdash M_1 : \theta$ ***contextually approximates*** $\Delta|\Gamma \vdash M_2 : \theta$ if, for all $\Delta' \supseteq \Delta$ and all contexts C such that $\Delta'|\emptyset \vdash C[M_i] : \mathsf{void}$, if $C[M_1] \Downarrow$ then $C[M_2] \Downarrow$. We then write $\Delta|\Gamma \vdash M_1 \lesssim M_2 : \theta$. Two terms are ***contextually equivalent*** (written $\Delta|\Gamma \vdash M_1 \cong M_2 : \theta$) if they approximate each other.

For technical convenience, IMJ features the let construct, even though it is definable: given $\Delta|\Gamma, x : \theta' \vdash M : \theta$ and $\Delta|\Gamma \vdash M' : \theta'$, consider $\mathsf{new}(\, x : \mathcal{I}; \mathsf{m} : \lambda x.M\,).\mathsf{m}(M')$, where $\mathcal{I}$ is a fresh interface with a single method $\mathsf{m} : \theta \to \theta'$. As usual, we write $M; M'$ for $\mathsf{let}\, x = M\, \mathsf{in}\, M'$, where x is not free in M'.

Although IMJ does not have explicit local variables, they could easily be introduced by taking $\mathsf{let}\,(x = \mathsf{new}(y : \mathcal{I}_\theta;))\,\mathsf{in} \cdots$, where $\mathcal{I}_\theta$ has a single field of type θ. In the same manner, one can define variables and methods that are *private* to objects, and invisible to the environment through interfaces.

Example 1 ([16]). Let $\Delta = \{\mathsf{Empty} : \emptyset, \mathsf{Cell} : (\mathsf{get} : \mathsf{void} \to \mathsf{Empty}, \mathsf{set} : \mathsf{Empty} \to \mathsf{void}), \mathsf{Var}_E : (\mathsf{val} : \mathsf{Empty}), \mathsf{Var}_I : (\mathsf{val} : \mathsf{int})\}$ and consider the terms $\Delta|\emptyset \vdash M_i : \mathsf{Cell}\ (i = 1, 2)$ defined by

$$M_1 \equiv \ \mathsf{let}\, v = \mathsf{new}(x : \mathsf{Var}_E;)\, \mathsf{in}\ \ \mathsf{new}(x : \mathsf{Cell}; \mathcal{M}_1)$$

$$\begin{aligned} M_2 \equiv \ & \mathsf{let}\, b = \mathsf{new}(x : \mathsf{Var}_I;)\, \mathsf{in} \\ & \mathsf{let}\, v_1 = \mathsf{new}(x : \mathsf{Var}_E;)\, \mathsf{in} \\ & \mathsf{let}\, v_2 = \mathsf{new}(x : \mathsf{Var}_E;)\, \mathsf{in}\ \ \mathsf{new}(x : \mathsf{Cell}; \mathcal{M}_2) \end{aligned}$$

with

$$\begin{aligned} \mathcal{M}_1 = \ & (\mathsf{get} : \lambda().v.\mathsf{val}, \\ & \ \mathsf{set} : \lambda y.(v.\mathsf{val} := y)) \\ \mathcal{M}_2 = \ & (\mathsf{get} : \lambda().\mathsf{if}\,(b.\mathsf{val})\,\mathsf{then}\,(b.\mathsf{val} := 0; v_1.\mathsf{val}) \\ & \qquad\qquad\qquad\qquad \mathsf{else}\,(b.\mathsf{val} := 1; v_2.\mathsf{val}), \\ & \ \mathsf{set} : \lambda y.(v_1.\mathsf{val} := y;\ v_2.\mathsf{val} := y)). \end{aligned}$$

We have $\Delta|\emptyset \vdash M_1 \cong M_2 : \mathsf{Cell}$. Intuitively, each of the two implementations of Cell corresponds to recording a single value of type Empty (using set) and providing access to it via get. The difference lies in the way the value is stored: a single private variable is used in M_1, while two variables are used in M_2. However, in the latter case the variables always hold the same value, so it does not matter which of the variables is used to return the value.

The game semantics of the two terms will turn out to consist of plays of the shape $*^{\emptyset}\, n^{\Sigma_0}\, G_0^*\, S_1\, G_1^*\, \cdots\, S_k\, G_k^*$, where

$$\begin{aligned} G_i \ &= \ \begin{cases} \mathsf{call}\, n.\mathsf{get}(*)^{\Sigma_0}\ \mathsf{ret}\, n.\mathsf{get}(\mathtt{nul})^{\Sigma_0} & i = 0 \\ \mathsf{call}\, n.\mathsf{get}(*)^{\Sigma_i}\ \mathsf{ret}\, n.\mathsf{get}(n_i)^{\Sigma_i} & i > 0 \end{cases} \\ S_i \ &= \ \mathsf{call}\, n.\mathsf{set}(n_i)^{\Sigma_i}\ \mathsf{ret}\, n.\mathsf{set}(*)^{\Sigma_i} \end{aligned}$$

and $\Sigma_i = \{n \mapsto (\mathsf{Cell}, \emptyset)\} \cup \{n_j \mapsto (\mathsf{Empty}, \emptyset) \mid 0 < j \leq i\}$. Intuitively, the plays describe all possible interactions of a Cell object. The first two moves $*^{\emptyset}\, n^{\Sigma_0}$ correspond to object creation. After that, the G_i segments represent the environment reading the current content (initially having null value), while the S_i segments correspond to updating the content with a reference name provided by the environment. The stores Σ_i attached to moves consist of all names that have been introduced during the interaction so far.

It is worth noting that, because IMJ has explicit casting, a context can always guess the actual interface of an object and extract any information we may want to hide through casting.

Example 2. Let $\Delta = \{\mathsf{Empty} : \emptyset, \mathsf{Point}\langle\mathsf{Empty}\rangle : (\mathsf{x} : \mathsf{int}, \mathsf{y} : \mathsf{int})\}$ and consider the terms $\Delta|\emptyset \vdash M_i : \mathsf{Empty}\ (i = 1, 2)$ defined by:

$$M_1 \equiv \ \mathsf{new}(x : \mathsf{Empty};),$$

$$M_2 \equiv \ \mathsf{let}\, p = \mathsf{new}(x : \mathsf{Point};)\, \mathsf{in}\ \ p.\mathsf{x} := 0; p.\mathsf{y} := 1; (\mathsf{Empty})p.$$

In our model they will be interpreted by the strategies $\sigma_1 = \{\epsilon, *^{\emptyset}\, n^{\{n \mapsto (\mathsf{Empty}, \emptyset)\}}\}$ and $\sigma_2 = \{\epsilon, *^{\emptyset}\, n^{\{n \mapsto (\mathsf{Point}, \{\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 1\})}\}$ respectively. Using e.g. the casting context $C \equiv (\mathsf{Point})\_; \mathsf{skip}$, we can see that $\Delta|\emptyset \vdash M_2 \not\lesssim M_1 : \mathsf{Empty}$. On the other hand, Theorem 20 will imply $\Delta|\emptyset \vdash M_1 \lesssim M_2 : \mathsf{Empty}$.

On the whole, IMJ is a compact calculus that strips down Middleweight Java to the essentials needed for interface-based interaction. Accordingly, we suppressed the introduction of explicit class hierarchy, as it would remain invisible to the environment anyway and any class-based internal computations can be represented using standard object encodings [1].

At the moment the calculus allows for single inheritance for interfaces only, but extending it to multiple inheritance is not problematic. The following semantic developments only rely on the assumption that $\leq$ must not give rise to circularities.

3. The game model

In our discussion below we assume a fixed interface table Δ.

The game model will be constructed using mathematical objects (moves, plays, strategies) that feature names drawn from the set $\mathbb{A}$. 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, 26]. Note that we do not

need the full power of the theory but mainly the basic notion of name-permutation. 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 $x = \pi \cdot y$.

We proceed to define a category of games. The objects of our category will be ***arenas***, which are nominal sets carrying specific type information.

Definition 4. An ***arena*** is a pair $A = (M_A, \xi_A)$ where:

- M_A is a nominal set of ***moves***;
- $\xi_A : M_A \to (\mathbb{A} \rightharpoonup \mathit{Ints})$ is a nominal *typing function*;

such that, for all $m \in M_A$, $\mathsf{dom}(\xi_A(m)) = \nu(m)$.

We start by defining the following basic arenas,

$$1 = (\{*\}, \{(*, \emptyset)\}, \quad \mathbb{Z} = (\mathbb{Z}, \{(i, \emptyset)\},$$
$$\mathcal{I} = (\mathbb{A} \cup \{\mathtt{nul}\}, \{(\mathtt{nul}, \emptyset)\} \cup \{(a, a, \mathcal{I})\}),$$

for all interfaces $\mathcal{I}$. Given arenas A and B, we can form the arena $A \times B$ by:

$$M_{A\times B} = \{(m, n) \in M_A \times M_B \mid a \in \nu(m) \cap \nu(n) \implies \xi_A(m, a) \leq \xi_B(n, a) \vee \xi_B(n, a) \leq \xi_A(m, a)\}$$

$$\xi_{A\times B}((m, n), a) = \begin{cases} \xi_A(m, a) & \text{if } a \notin \nu(n) \vee \xi_A(m, a) \leq \xi_B(n, a) \\ \xi_B(n, a) & \text{otherwise} \end{cases}$$

Another important arena is $\#(\mathcal{I}_1, \cdots, \mathcal{I}_n)$, with:

$$M_{\#(\vec{\mathcal{I}})} = \{(a_1, \cdots, a_n) \in \mathbb{A}^n \mid a_i\text{'s distinct}\}$$
$$\xi_{\#(\vec{\mathcal{I}})}((a_1, \cdots, a_n), a_i) = \mathcal{I}_i$$

for all $n \in \mathbb{N}$. In particular, $\mathbb{A}^{\#0} = 1$.

For each type θ, we set Val_θ to be the set of *semantic values* of type θ, given by:

$$\mathit{Val}_{\mathsf{void}} = M_1, \quad \mathit{Val}_{\mathsf{int}} = M_{\mathbb{Z}}, \quad \mathit{Val}_{\mathcal{I}} = M_{\mathcal{I}}.$$

For each type sequence $\vec{\theta} = \theta_1, \cdots, \theta_n$, we set $\mathit{Val}_{\vec{\theta}} = \mathit{Val}_{\theta_1} \times \cdots \times \mathit{Val}_{\theta_n}$.

We let a ***store*** Σ be a type-preserving finite partial function from names to object types and field assignments, that is, $\Sigma : \mathbb{A} \rightharpoonup \mathit{Ints} \times (\mathit{Flds} \rightharpoonup \mathit{Val})$ such that $|\Sigma|$ is finite and

$$\Sigma(a) : \mathcal{I} \wedge \Delta(\mathcal{I}).\mathsf{f} = \theta \implies \Sigma(a).\mathsf{f} = v \wedge \Sigma \vdash v \leq \theta,$$

where the new notation is explained below. First, assuming $\Sigma(a) = (\mathcal{I}', \phi)$, the judgement $\Sigma(a) : \mathcal{I}$ holds iff $\mathcal{I} = \mathcal{I}'$ and $\Sigma(a).\mathsf{f}$ stands for $\phi(\mathsf{f})$. Next we define typing rules for values in store contexts:

$$\frac{v \in \mathit{Val}_{\mathsf{void}}}{\Sigma \vdash v : \mathsf{void}} \qquad \frac{v \in \mathit{Val}_{\mathsf{int}}}{\Sigma \vdash v : \mathsf{int}} \qquad \frac{\Sigma(v) : \mathcal{I} \vee v = \mathtt{nul}}{\Sigma \vdash v : \mathcal{I}}$$

and write $\Sigma \vdash v \leq \theta$ for $\Sigma \vdash v : \theta \vee (\Sigma \vdash v : \mathcal{I}' \wedge \mathcal{I}' \leq \theta)$.

We let Sto be the set of all stores. We write $\mathsf{dom}(\Sigma(a))$ for the set of all f such that $\Sigma(a).\mathsf{f}$ is defined. We let Sto_0 contain all stores Σ such that:

$$\forall a \in \mathsf{dom}(\Sigma), \mathsf{f} \in \mathsf{dom}(\Sigma(a)).\ \Sigma(a).\mathsf{f} \in \{*, 0, \mathtt{nul}\}$$

and we call such a Σ a *default store*.

Given arenas A and B, plays in AB will consist of sequences of moves (with store) which will be either moves from $M_A \cup M_B$, or moves representing method calls and returns. Formally, we define:

$$M_{AB} = M_A \cup M_B \cup \mathit{Calls} \cup \mathit{Retns}$$

where we set $\mathit{Calls} = \{\mathsf{call}\ a.\mathsf{m}(\vec{v}) \mid a \in \mathbb{A} \wedge \vec{v} \in \mathit{Val}^*\}$ and $\mathit{Retns} = \{\mathsf{ret}\ a.\mathsf{m}(v) \mid a \in \mathbb{A} \wedge v \in \mathit{Val}\}$.

Definition 5. A ***legal sequence*** in AB is a sequence of moves from M_{AB} that adheres to the following grammar (*Well-Bracketing*), where m_A and m_B range over M_A and M_B respectively.

$$\begin{aligned} L_{AB} &::= \epsilon \mid m_A X \mid m_A Y m_B X \\ X &::= Y \mid Y\,(\mathsf{call}\ a.\mathsf{m}(\vec{v}))\,X \\ Y &::= \epsilon \mid YY \mid (\mathsf{call}\ a.\mathsf{m}(\vec{v}))\,Y\,(\mathsf{ret}\ a.\mathsf{m}(v)) \end{aligned}$$

We write L_{AB} for the set of legal sequences in AB. In the last clause above, we say that $\mathsf{call}\ a.\mathsf{m}(\vec{v})$ ***justifies*** $\mathsf{ret}\ a.\mathsf{m}(v)$.

To each $s \in L_{AB}$ we assign a ***polarity*** function p from move occurrences in s to the set $Pol_1 = \{O, P\}$. Polarities represent the two players in our game reading of programs: O is the *Opponent* and P is the *Proponent* in the game. The latter corresponds to the modelled program, while the former models the possible computational environments surrounding the program. Polarities are complemented via $\overline{O} = \{P\}$ and $\overline{P} = \{O\}$. In addition, the polarity function must satisfy the condition:

- For all $m_X \in M_X$ $(X = A, B)$ occurring in s we have $p(m_A) = O$ and $p(m_B) = P$; (***O-starting***)
- If mn are consecutive moves in s then $p(n) \in \overline{p(m)}$. (***Alternation***)

It follows that there is a unique p for each legal sequence s, namely the one which assigns O precisely to those moves appearing in odd positions in s.

A ***move-with-store*** in AB is a pair m^Σ with $\Sigma \in \mathit{Sto}$ and $m \in M_{AB}$. For each sequence s of moves-with-store we define the set of *available names* of s by:

$$\mathsf{Av}(\epsilon) = \emptyset, \quad \mathsf{Av}(sm^\Sigma) = \Sigma^*(\mathsf{Av}(s) \cup \nu(m))$$

where, for each $X \subseteq \mathbb{A}$, we let $\Sigma^*(X) = \bigcup_i \Sigma^i(X)$, with

$$\Sigma^0(X) = X, \quad \Sigma^{i+1}(X) = \nu(\Sigma(\Sigma^i(X))).$$

That is, a name is available in s just if it appears inside a move in s, or it can be reached from an available name through some store in s. We write $\underline{s}$ for the underlying sequence of moves of s (i.e. $\pi_1(s)$), and let $\sqsubseteq$ denote the prefix relation between sequences. If $s'm^\Sigma \sqsubseteq s$ and $a \in \nu(m^\Sigma) \setminus \nu(s')$ then we say a ***is introduced*** by m^Σ in s.[1] In such a case, we define the *owner* of the name a in s, written $o(a)$, to be $p(m)$ (where p is the polarity associated with $\underline{s}$). For each polarity $X \in \{O, P\}$ we let

$$X(s) = \{a \in \nu(s) \mid o(a) = X\}$$

be the set of names in s owned by X.

Definition 6. A ***play*** in AB is a sequence of moves-with-store s such that $\underline{s}$ is a legal sequence and, moreover, for all $s'm^\Sigma \sqsubseteq s$:

- It holds that $\mathsf{dom}(\Sigma) = \mathsf{Av}(s'm^\Sigma)$. (*Frugality*)
- If $a \in \mathsf{dom}(\Sigma)$ with $\Sigma(a) : \mathcal{I}$ then:
  - if $m \in M_X$, for $X \in \{A, B\}$, then $\mathcal{I} \leq \xi_X(m, a)$;
  - for all n^T in s', if $a \in \mathsf{dom}(T)$ then $T(a) : \mathcal{I}$;
  - if $\Delta(\mathcal{I}).\mathsf{m} = \vec{\theta} \to \theta$ then:
    - if $m = \mathsf{call}\ a.\mathsf{m}(\vec{v})$ then $\Sigma \vdash \vec{v} : \vec{\theta'}$ for some $\vec{\theta'} \leq \vec{\theta}$,
    - if $m = \mathsf{ret}\ a.\mathsf{m}(v)$ then $\Sigma \vdash v : \theta'$ for some $\theta' \leq \theta$.

  (*Well-classing*)
- If $m = \mathsf{call}\ a.\mathsf{m}(\vec{v})$ then $o(a) \in \overline{p(m)}$. (*Well-calling*)

We write P_{AB} for the set of plays in AB.

[1] By abuse of notation, we frequently write instead "a is introduced by m in s". Recall also that $\nu(s)$ collects all names appearing in s; in particular, $\nu(m_1^{\Sigma_1} \cdots m_i^{\Sigma_i}) = \nu(m_1) \cup \nu(\Sigma_1) \cup \cdots \cup \nu(m_i) \cup \nu(\Sigma_i)$.

Note above that, because of well-bracketing and alternation, if $m = \mathsf{ret}\, a.\mathsf{m}(v)$ then well-calling implies $o(a) = p(m)$. Thus, the frugality condition stipulates that names cannot appear in a play in unreachable parts of a store (cf. [17]). Moreover, well-classing ensures that the typing information in stores is consistent and adheres to the constraints imposed by Δ and the underlying arenas. Finally, well-calling implements the specification that each player need only call the other player's methods. This is because calls to each player's own methods cannot in general be observed and so should not be accounted for in plays.

Given arenas A, B, C, next we define interaction sequences, which show how plays from AB and BC can interact to produce a play in AC. The sequences will rely on moves with stores, where the moves come from the set:

$$M_{ABC} = M_A \cup M_B \cup M_C \cup \mathit{Calls} \cup \mathit{Retns}\,.$$

The moves will be assigned polarities from the set:

$$\mathit{Pol}_2 = \{\, O_L, P_L, O_L P_R, P_L O_R, O_R, P_R \,\}\,.$$

The index L stands for "left", while R means "right". The indices indicate which part of the interaction (A, B or C) a move comes from, and what polarity it has therein. We also consider an auxiliary notion of *pseudo-polarities*:

$$OO = \{O_L, O_R\},\ \ PO = \{P_L, P_L O_R\},\ \ OP = \{P_R, O_L P_R\}.$$

Each polarity has an opposite pseudo-polarity determined by:

$$\overline{O_L} = \overline{O_L P_R} = PO,\ \ \overline{O_R} = \overline{P_L O_R} = OP,\ \ \overline{P_L} = \overline{P_R} = OO.$$

Finally, each $X \in \{AB, BC, AC\}$ has a designated set of polarities given by:

$$\begin{aligned} p(AB) &= \{O_L, P_L, O_L P_R, P_L O_R\},\\ p(BC) &= \{O_R, P_R, O_L P_R, P_L O_R\},\\ p(AC) &= \{O_L, P_L, O_R, P_R\}. \end{aligned}$$

Note the slight abuse of notation with p, as it is also used for move polarities.

Suppose $X \in \{AB, BC, AC\}$. Consider a sequence s of moves-with-store from ABC (i.e. a sequence with elements m^Σ with $m \in M_{ABC}$) along with an assignment p of polarities from Pol_2 to moves of s. Let $s \upharpoonright X$ be the subsequence of s containing those moves-with-store m^Σ of s for which $p(m) \in p(X)$. Additionally, we define $s \upharpoonright_\gamma X$ to be $\gamma(s \upharpoonright X)$, where the function γ acts on moves-with-store by restricting the domains of stores to available names:

$$\gamma(\epsilon) = \epsilon, \quad \gamma(s m^\Sigma) = \gamma(s)\, m^{\Sigma \upharpoonright \mathsf{Av}(s m^\Sigma)}.$$

Definition 7. An ***interaction sequence*** in ABC is a sequence s of moves-with-store in ABC satisfying the following conditions.

- For each $s' m^\Sigma \sqsubseteq s$, $\mathsf{dom}(\Sigma) = \mathsf{Av}(s' m^\Sigma)$. (*Frugality*)
- If $s' m^\Sigma \sqsubseteq s$ and $a \in \mathsf{dom}(\Sigma)$ with $\Sigma(a) : \mathcal{I}$ then:
  - if $m \in M_X$, for $X \in \{A, B, C\}$, then $\mathcal{I} \leq \xi_X(m, a)$;
  - for all n^T in s', if $a \in \mathsf{dom}(T)$ then $T(a) : \mathcal{I}$;
  - if $\Delta(\mathcal{I}).\mathsf{m} = \vec{\theta} \to \theta$ then:
    - if $m = \mathsf{call}\, a.\mathsf{m}(\vec{v})$ then $\Sigma \vdash \vec{v} : \vec{\theta'}$ for some $\vec{\theta'} \leq \vec{\theta}$,
    - if $m = \mathsf{ret}\, a.\mathsf{m}(v)$ then $\Sigma \vdash v : \theta'$ for some $\theta' \leq \theta$.

  (*Well-classing*)
- There is a polarity function p from move occurrences in $\underline{s}$ to Pol_2 such that:
  - For all $m_X \in M_X$ ($X = A, B, C$) occurring in $\underline{s}$ we have $p(m_A) = O_L$, $p(m_B) = P_L O_R$ and $p(m_C) = P_R$;
  - If mn are consecutive moves in $\underline{s}$ then $p(n) \in \overline{p(m)}$.

  (*Alternation*)

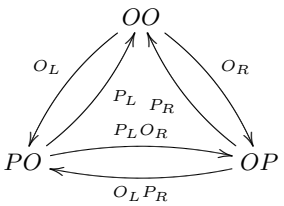

Figure 3. Interaction diagram for $\mathit{Int}(ABC)$. The diagram specifies the alternation of polarities in interaction sequences. Transitions are labelled by move polarities, while OO is the initial state.

- If $s' m^\Sigma \sqsubseteq s$ then $m = \mathsf{call}\, a.\mathsf{m}(\overline{v})$ implies $o(a) \in \overline{p(m)}$. (*Well-calling*)
- For each $X \in \{AB, BC, AC\}$, $\underline{s \upharpoonright X} \in L_X$. (*Projecting*)
- If $s' m^\Sigma \sqsubseteq s$ and $m = \mathsf{ret}\, a.\mathsf{m}(v)$ then there is a move n^T in s' such that, for all X such that $p(m) \in p(X)$, n is the justifier of m in $\underline{s \upharpoonright X}$. (*Well-returning*)
- *Laird's conditions* [17]:
  - $P(s \upharpoonright_\gamma AB) \cap P(s \upharpoonright_\gamma BC) = \emptyset$;
  - $(P(s \upharpoonright_\gamma AB) \cup P(s \upharpoonright_\gamma BC)) \cap O(s \upharpoonright_\gamma AC) = \emptyset$;
  - For each $s' \sqsubseteq s$ ending in $m^\Sigma n^T$ and each $a \in \mathsf{dom}(T)$, if
    - $p(m) \in PO$ and $a \notin \nu(s' \upharpoonright_\gamma AB)$,
    - or $p(m) \in OP$ and $a \notin \nu(s' \upharpoonright_\gamma BC)$,
    - or $p(m) \in OO$ and $a \notin \nu(s' \upharpoonright_\gamma AC)$,

    then $\Sigma(a) = T(a)$.

We write $\mathit{Int}(ABC)$ for the set of interaction sequences in ABC.

Note that, by projecting and well-returning, each return move in s has a unique justifier. Next we show that the polarities of moves inside an interaction sequence are uniquely determined by the ***interaction diagram*** of Figure 3. The diagram can be seen as an automaton accepting $\underline{s}$, for each $s \in \mathit{Int}(ABC)$. The edges represent moves by their polarities, while the labels of vertices specify the polarity of the next (outgoing) move. For example, from OO we can only have a move m with $p(m) \in \{O_L, O_R\}$, for any p.

Lemma 1. *Each $s \in \mathit{Int}(ABC)$ has a unique polarity function p.*

Proof. Suppose $s \in \mathit{Int}(ABC)$. We claim that the alternation, well-calling, projecting and well-returning conditions uniquely specify p. Consider the interaction diagram of Figure 3, which we read as an automaton accepting $\underline{s}$, call it $\mathcal{A}$. The edges represent moves by their polarities, while the labels of vertices specify the polarity of the next (outgoing) move. By projecting we obtain that the first element of $\underline{s}$ is some m_A and, by alternation, its polarity is O_L. Thus, OO is the initial state.
We now use induction on $|s|$ to show that $\mathcal{A}$ has a unique run on $\underline{s}$. The base case is trivial, so suppose $\underline{s} = s'm$. By induction hypothesis, $\mathcal{A}$ has a unique run on s', which reaches some state X. We do a case analysis on m. If $m \in M_A \cup M_B \cup M_C$ then there is a unique edge accepting m and, by alternation, this edge must depart from X. If, on the other hand, $m = \mathsf{call}\, a.\mathsf{m}(\vec{v})$ then the fact that $o(a) \in \overline{p(m)}$ gives two possible edges for accepting m. But observe that no combination of such edges can depart from X. Finally, let $m = \mathsf{ret}\, a.\mathsf{m}(v)$ be justified by some n in s'. Then, by well-bracketing, n is the justifier of m in all projections, and hence the edge accepting m must be the opposite of the one accepting n (e.g. if m is accepted by O_L then n is accepted by P_L). □

Next we show that interaction sequences project to plays. The projection of interaction sequences in ABC on AB, BC and AC

leads to the following definition of projections of polarities,

$$\pi_{AB}(X_L) = X \qquad \pi_{AB}(X_LY_R) = X \qquad \pi_{AB}(Y_R) = \text{undef.}$$
$$\pi_{BC}(X_L) = \text{undef.} \qquad \pi_{BC}(X_LY_R) = Y \qquad \pi_{BC}(Y_R) = Y$$
$$\pi_{AC}(X_L) = X \qquad \pi_{AC}(X_LY_R) = \text{undef.} \qquad \pi_{AC}(Y_R) = Y$$

where $X, Y \in \{O, P\}$. We can now show the following.

Lemma 2. *Let $s \in Int(ABC)$. Then, for each $X \in \{AB, BC, AC\}$ and each m^{Σ} in s, if $p(m) \in p(X)$ then $\pi_X(p(m)) = p_X(m)$, where p_X is the polarity function of $s \upharpoonright X$.*

Proof. We show this for $X = AB$, the other cases are proven similarly, by induction on $|s| \geq 0$; the base case is trivial. For the inductive case, if m is the first move in s with polarity in $p(AB)$ then, by projecting, $m \in M_A$ and therefore $p(m) = O_L$ and $p_{AB}(m) = O$, as required. Otherwise, let n be the last move in s with polarity in $p(AB)$ before m. By IH, $p_{AB}(n) = \pi_{AB}(p(n))$. Now, by projecting, $p_{AB}(m) = p_{AB}(n)$ and observe that, for all $X \in p(n)$, $\pi_{AB}(X) = \pi_{AB}(p(n))$, so in particular $\pi_{AB}(p(m)) = \pi_{AB}(p(n)) = p_{AB}(n) = p_{AB}(m)$. □

The following lemma formulates a taxonomy on names appearing in interaction sequences.

Lemma 3. *Let $s \in Int(ABC)$. Then,*

1. *$\nu(s) = O(s \upharpoonright_\gamma AC) \uplus P(s \upharpoonright_\gamma AB) \uplus P(s \upharpoonright_\gamma BC)$;*
2. *if $s = tm^{\Sigma}$ and:*
   - *$p(m) \in OO$ and $s \upharpoonright_\gamma AC = t'm^{\Sigma'}$,*
   - *or $p(m) \in PO$ and $s \upharpoonright_\gamma AB = t'm^{\Sigma'}$,*
   - *or $p(m) \in OP$ and $s \upharpoonright_\gamma BC = t'm^{\Sigma'}$,*

   then $\nu(t) \cap \nu(m^{\Sigma'}) \subseteq \nu(t')$ and, in particular, if m introduces name a in $t'm^{\Sigma'}$ then m introduces a in s.

Proof. For 1, by definition of interactions we have that these sets are disjoint. It therefore suffices to show the left-to-right inclusion. Suppose that $a \in \nu(s)$ is introduced in some m^{Σ} in s, with $p(m) \in PO$, and let $s \upharpoonright_\gamma AB = \cdots m^{\Sigma'} \cdots$. If $a \in \nu(m^{\Sigma'})$ then $a \in P(s \upharpoonright_\gamma AB)$, as required. Otherwise, by Laird's last set of conditions, a is copied from the store of the move preceding m^{Σ} in s, a contradiction to its being introduced at m^{Σ}. Similarly if $p(m) \in OP$. Finally, if $p(m) \in OO$ then we work similarly, considering $O(s \upharpoonright_\gamma AC)$.
For 2, we show the first case, and the other cases are similar. It suffices to show that $(\nu(m^{\Sigma'}) \setminus \nu(t')) \cap \nu(t) = \emptyset$. So suppose $a \in \nu(m^{\Sigma'}) \setminus \nu(t')$, therefore $a \in O(s \upharpoonright_\gamma AC)$. But then we cannot have $a \in \nu(t)$ as the latter, by item 1, would imply $a \in P(s \upharpoonright_\gamma AB) \cup P(s \upharpoonright_\gamma BC)$. □

Proposition 4. *For all $s \in Int(ABC)$, the projections $s \upharpoonright_\gamma AB$, $s \upharpoonright_\gamma BC$ and $s \upharpoonright_\gamma AC$ are plays in AB, BC and AC respectively.*

Proof. By frugality of s and application of γ, all projections satisfy frugality. Moreover, well-classing is preserved by projections. For well-calling, let $m = \mathsf{call}\, a.\mathsf{m}(\vec{v})$ be a move in s and let n^T be the move introducing a in s. Suppose $p(m) \in p(AB)$ and let us assume $p_{AB}(m) = O$. We need to show that $o_{AB}(m) = P$. By $p_{AB}(m) = O$ we obtain that $p(m) \in \{O_L, O_LP_R\}$ and, by well-calling of s, we have that $o(a) \in PO$. Thus, $p(n) \in PO$ and, by Lemma 3, n introduces a in $s \upharpoonright_\gamma AB$ and therefore $o_{AB}(n) = P$, as required. If, on the other hand, $p_{AB}(m) = P$ then we obtain $p(n) \in OO \cup OP$ and therefore, by Lemma 3, $a \in P(s \upharpoonright_\gamma BC) \cup O(s \upharpoonright_\gamma AC)$. Thus, by the same lemma, $a \notin P(s \upharpoonright_\gamma AB)$ and hence $o_{AB}(a) = O$. The cases for the other projections are shown similarly. □

In our setting programs will be represented by *strategies* between arenas. We shall introduce them next after a few auxiliary definitions. Intuitively, strategies capture the observable computational patterns produced by a program.

Let us define the following notion of subtyping between stores. For $\Sigma, \Sigma' \in Sto$, $\Sigma \leq \Sigma'$ holds if, for all names a,

$$\Sigma'(a) : \mathcal{I}' \implies \Sigma(a) \leq \mathcal{I}' \wedge \forall \mathsf{f} \in \mathsf{dom}(\Sigma'(a)).\Sigma(a).\mathsf{f} = \Sigma'(a).\mathsf{f}$$

In particular, if a is in the domain of Σ', Σ may contain more information about a because of assigning to a a larger interface. Accordingly, for plays $s, s' \in P_{AB}$, we say that s is an ***O-extension*** of s' if s and s' agree on their underlying sequences, while their stores may differ due to subtyping related to O-names. Where such subtyping leads to s having stores with more fields than those in s', P is assumed to copy the values of those fields. Formally, $s \leq_O s'$ is defined by the rules:

$$\frac{}{\epsilon \leq_O \epsilon} \qquad \frac{s \leq_O s' \quad \Sigma \leq \Sigma' \quad \Sigma \upharpoonright P(sm^{\Sigma}) \subseteq \Sigma'}{sm^{\Sigma} \leq_O s'm^{\Sigma'}}\, p(m)=O$$

$$\frac{sn^T \leq_O s' \quad \Sigma \leq \Sigma' \quad \Sigma \text{ extends } \Sigma' \text{ by } T}{sn^Tm^{\Sigma} \leq_O s'm^{\Sigma'}}\, p(m)=P$$

where Σ extends Σ' by T if:

- for all $a \in \mathsf{dom}(\Sigma) \setminus \mathsf{dom}(\Sigma')$, $\Sigma(a) = T(a)$;
- for all a and $\mathsf{f} \in \mathsf{dom}(\Sigma(a)) \setminus \mathsf{dom}(\Sigma'(a))$, $\Sigma(a).\mathsf{f} = T(a).\mathsf{f}$.

The utility of O-extension is to express semantically the fact that the environment of a program may use up-casting to inject in its objects additional fields (and methods) not accessible to the program.

Definition 8. A ***strategy*** σ in AB is a non-empty set of even-length plays from P_{AB} satisfying the conditions:

- If $sm^{\Sigma}n^T \in \sigma$ then $s \in \sigma$. (*Even-prefix closure*)
- If $sm^{\Sigma}, sn^T \in \sigma$ then $sm^{\Sigma} \sim sn^T$. (*Determinacy*)
- If $s \in \sigma$ and $s \sim t$ then $t \in \sigma$. (*Equivariance*)[2]
- If $s \in \sigma$ and $t \leq_O s$ then $t \in \sigma$. (*O-extension*)

We write $\sigma : A \to B$ when σ is a strategy in AB. If $\sigma : A \to B$ and $\tau : B \to C$, we define their ***composition*** $\sigma; \tau$ by:

$$\sigma; \tau = \{s \upharpoonright_\gamma AC \mid s \in \sigma \| \tau\}$$

where $\sigma \| \tau = \{s \in Int(ABC) \mid s \upharpoonright_\gamma AB \in \sigma \wedge s \upharpoonright_\gamma BC \in \tau\}$.

In definitions of strategies we may often leave the presence of the empty sequence implicit, as the latter is a member of every strategy. For example, for each arena A, we define the strategy:

$$\mathtt{id}_A : A \to A = \{m^{\Sigma}_A m^{\Sigma}_A \in P_{AA}\}$$

The next series of lemmata allow us to show that strategy composition is well defined.

Lemma 5. *If $sm^{\Sigma}, sn^T \in \sigma \| \tau$ with $p(m) \notin OO$ then $sm^{\Sigma} \sim sn^T$. Hence, if $s_1m^{\Sigma}, s_2n^T \in \sigma \| \tau$ with $p(m) \notin OO$ and $s_1 \sim s_2$ then $s_1m^{\Sigma} \sim s_2n^T$.*

Proof. For the latter part, if $s_1 = \pi \cdot s_2$ then, since $\pi \cdot (s_2n^T) \in \sigma \| \tau$, by former part of the claim we have $s_1m^{\Sigma} \sim \pi \cdot (s_2n^T)$ so $s_1m^{\Sigma} \sim s_2n^T$.
Now, for the former part, suppose WLOG that $p(m) \in PO$. Then, by the interaction diagram, we also have $p(n) \in PO$. As $sm^{\Sigma}, sn^T \upharpoonright_\gamma AB \in \sigma$, by determinacy of σ we get $s'm^{\Sigma'} \sim s'n^{T'}$,

[2] Recall that, for any nominal set X and $x, y \in X$, we write $x \sim y$ just if there is a permutation π such that $x = \pi \cdot y$.

with $s'm^{\Sigma'} = sm^{\Sigma} \restriction_\gamma AB$ and $s'n^{T'} = sn^{T} \restriction_\gamma AB$. We therefore have $(s', m^{\Sigma'}) \sim (s', n^{T})$ and, trivially, $(s, s') \sim (s, s')$. Moreover, by Lemma 3, $\nu(m^{\Sigma'}) \cap \nu(s) \subseteq \nu(s')$ and $\nu(n^{T'}) \cap \nu(s) \subseteq \nu(s')$ hence, by Strong Support Lemma [26], $sm^{\Sigma'} \sim sn^{T'}$. By Laird's last set of conditions, the remaining values of Σ, T are determined by the last store in s, hence $sm^{\Sigma} \sim sn^{T}$. □

Lemma 6. *If $s_1, s_2 \in \sigma \| \tau$ end in moves with polarities in $p(AC)$ and $s_1 \restriction_\gamma AC = s_2 \restriction_\gamma AC$ then $s_1 \sim s_2$.*

Proof. By induction on $|s_1 \restriction_\gamma AC| > 0$. The base case is encompassed in $s_i = s_i' m^{\Sigma_i}$ with $p(m) \in OO$, $i = 1, 2$, where note that by IH m will have the same polarity in s_1, s_2. Then, by IH we get $s_1' = \pi \cdot s_2'$, for some π. Let $s_i'' m^{\Sigma'} = s_i \restriction_\gamma AC$, for $i = 1, 2$, so in particular $s_1'' = \pi \cdot s_2''$ and therefore $(s_1', s_1'') \sim (s_2', s_2'')$. Moreover, by hypothesis, we trivially have $(m^{\Sigma'}, s_1'') \sim (m^{\Sigma'}, s_2'')$ and hence, by Lemma 3 and Strong Support Lemma [26], we obtain $s_1' m^{\Sigma'} \sim s_2' m^{\Sigma'}$ which implies $s_1 \sim s_2$ by Laird's conditions. Suppose now $s_i = s_i' s_i'' m^{\Sigma_i}$, $i = 1, 2$, with $p(m) \in P(AC) \setminus OO$ and the last move in s_i' being the last move in $s_i' s_i''$ having polarity in $p(AC)$. By IH, $s_1' \sim s_2'$. Then, by consecutive applications of Lemma 5, we obtain $s_1 \sim s_2$. □

Proposition 7. *If $\sigma : A \to B$ and $\tau : B \to C$ then $\sigma; \tau : A \to C$.*

Proof. We show that $\sigma; \tau$ is a strategy. Even-prefix closure and equivariance are clear. Moreover, since each $s \in \sigma \| \tau$ has even-length projections in AB and BC, we can show that its projection in AC is even-length too. For O-extension, if $s \in \sigma; \tau$ and $t \leq_O s$ with $s = u \restriction_\gamma AC$ and $u \in \sigma \| \tau$, we can construct $v \in Int(ABC)$ such that $t = v \restriction_\gamma AC$ and $v \leq_O u$, where $\leq_O$ is defined for interaction sequences in an analogous way as for plays (with condition $p(m) = O$ replaced by $p(m) \in OO$, and $p(m) = P$ by $p(m) \in PO \cup OP$). Moreover, $v \restriction_\gamma AB \leq_O u \restriction_\gamma AB$ and $v \restriction_\gamma BC \leq_O u \restriction_\gamma BC$, so $t \in \sigma; \tau$. Finally, for determinacy, let $sm^{\Sigma}, sn^{T} \in \sigma; \tau$ be due to $s_1 s_1' m^{\Sigma'}, s_2 s_2' n^{T'} \in \sigma \| \tau$ respectively, where s_1, s_2 both end in the last move of s. By Lemma 6, we have $s_1 \sim s_2$ and thus, by consecutive applications of Lemma 5, we get $s_1 s_1' m^{\Sigma'} \sim s_2 s_2' n^{T'}$, so $sm^{\Sigma} \sim sn^{T}$. □

The above result shows that strategies are closed under composition. We can prove that composition is associative and, consequently, obtain a category of games.

Proposition 8. *For all $\rho : A \to B$, $\sigma : B \to C$ and $\tau : C \to D$, $(\rho; \sigma); \tau = \rho; (\sigma; \tau)$.*

Definition 9. Given a class table Δ, we define the category $\mathcal{G}_\Delta$ having arenas as objects and strategies as morphisms. Identity morphisms are given by $\mathtt{id}_A$, for each arena A.

Note that neutrality of identity strategies easily follows from the definitions and, hence, $\mathcal{G}_\Delta$ is well defined. In the sequel, when Δ can be inferred from the context, we shall write $\mathcal{G}_\Delta$ simply as $\mathcal{G}$. As a final note, for class tables $\Delta \subseteq \Delta'$, we can define a functor

$$\Delta/\Delta' : \mathcal{G}_\Delta \to \mathcal{G}_{\Delta'}$$

which acts as the identity map on arenas, and sends each $\sigma : A \to B$ of $\mathcal{G}_\Delta$ to:

$$(\Delta/\Delta')(\sigma) = \{s \in P_{AB}^{\Delta'} \mid \exists t \in \sigma.\, s \leq_O t\}$$

where $P_{AB}^{\Delta'}$ refers to plays in $\mathcal{G}_{\Delta'}$. In the other direction, we can define a strategy transformation:

$$(\Delta'/\Delta)(\sigma) = \sigma \cap P_{AB}^{\Delta}$$

which satisfies $\Delta'/\Delta(\Delta/\Delta'(\sigma)) = \sigma$.

4. Soundness

Here we introduce constructions that will allow us to build a model of IMJ. We begin by defining a special class of strategies. A strategy $\sigma : A \to B$ is called ***evaluated*** if there is a function $f_\sigma : M_A \to M_B$ such that:

$$\sigma = \{m_A^{\Sigma} m_B^{\Sigma} \in P_{AB} \mid m_B = f_\sigma(m_A)\}.$$

Note that equivariance of σ implies that, for all $m_A \in M_A$ and permutations π, it holds that $\pi \cdot f_\sigma(m_A) = f_\sigma(\pi \cdot m_A)$. Thus, in particular, $\nu(f_\sigma(m_A)) \subseteq \nu(m_A)$.

Recall that, for arenas A and B, we can construct a product arena $A \times B$. We can also define projection strategies:

$$\pi_1 : A \times B \to A = \{(m_A, m_B)^{\Sigma} m_A^{\Sigma} \in P_{(A \times B)A}\}$$

and, analogously, $\pi_2 : A \times B \to B$. Note that the projections are evaluated. Moreover, for each object A,

$$!_A = \{m_A^{\Sigma} *^{\Sigma} \mid m_A^{\Sigma} \in P_{A1}\}$$

is the unique evaluated strategy of type $A \to 1$.

Given strategies $\sigma : A \to B$ and $\tau : A \to C$, with τ evaluated, we define:

$$\langle \sigma, \tau \rangle : A \to B \times C = \{m_A^{\Sigma} s[(m_B, f_\tau(m_A))/m_B] \mid m_A^{\Sigma} s \in \sigma\}$$

where we write $s[m'/m_B]$ for the sequence obtained from s by replacing any occurrences of m_B in it by m' (note that there can be at most one occurrence of m_B in s).

The above structure yields products for evaluated strategies.

Lemma 9. *Evaluated strategies form a wide subcategory of $\mathcal{G}$ which has finite products, given by the above constructions. Moreover, for all $\sigma : A \to B$ and $\tau : A \to C$ with τ evaluated, $\langle \sigma, \tau \rangle; \pi_1 = \sigma$ and $\langle \sigma, \tau \rangle = \langle \sigma, \mathtt{id}_A \rangle; \langle \pi_1, \pi_2; \tau \rangle$.*

Using the above result, we can extend pairings to general $\sigma : A \to B$ and $\tau : A \to C$ by:

$$\langle \sigma, \tau \rangle \;=\; A \xrightarrow{\langle \sigma, \mathtt{id}_A \rangle} B \times A \xrightarrow{\langle \pi_2; \tau, \pi_1 \rangle} C \times B \xrightarrow{\cong} B \times C$$

where $\cong$ is the isomorphism $\langle \pi_2, \pi_1 \rangle$. The above represents a notion of *left-pairing* of σ and τ, where the effects of σ precede those of τ. We can also define a *left-tensor* between strategies:

$$\sigma \times \tau \;=\; A \times B \xrightarrow{\langle \pi_1; \sigma, \pi_2 \rangle} A' \times B \xrightarrow{\langle \pi_1, \pi_2; \tau \rangle} A' \times B'$$

for any $\sigma : A \to A'$ and $\tau : B \to B'$.

Lemma 10. *Let $\tau' : A' \to A$, $\sigma : A \to B_1$, $\tau : A \to B_2$, $\sigma_1 : B_1 \times B_2 \to C_1$ and $\sigma_2 : B_2 \to C_2$, with τ and τ' evaluated. Then $\tau'; \langle \sigma, \tau \rangle; \langle \sigma_1, \pi_2; \sigma_2 \rangle = \langle \tau'; \langle \sigma, \tau \rangle; \sigma_1, \tau'; \tau; \sigma_2 \rangle$.*

Proof. The result follows from the simpler statements:

$$\tau; \langle \sigma, \mathtt{id} \rangle = \langle \tau; \sigma, \tau \rangle, \quad \langle \sigma, \mathtt{id} \rangle; \langle \sigma', \pi_2 \rangle = \langle \langle \sigma; \mathtt{id} \rangle; \sigma', \mathtt{id} \rangle,$$

for all appropriately typed σ, σ', τ, with τ evaluated, and Lemma 9. □

An immediate consequence of the above is:

$$A \xrightarrow{\langle \sigma; \tau \rangle} B_1 \times B_2 \xrightarrow{\sigma_1 \times \sigma_2} C_1 \times C_2 \;=\; A \xrightarrow{\langle \sigma; \sigma_1, \tau; \sigma_2 \rangle} C_1 \times C_2$$

More generally, Lemma 10 provides us with naturality conditions similar to those present in *Freyd categories* [25] or, equivalently, categories with monadic products [22].

We also introduce the following weak notion of coproduct. Given strategies $\sigma, \tau : A \to B$, we define:

$$[\sigma, \tau] : \mathbb{Z} \times A \to B = \{(1, m_A)^{\Sigma} s \mid m_A^{\Sigma} s \in \sigma\} \cup \{(0, m_A)^{\Sigma} s \mid m_A^{\Sigma} s \in \tau\}$$

Setting $\hat{i} : 1 \to \mathbb{Z} = \{*\, i\}$, for each $i \in \mathbb{Z}$, we can show the following.

Lemma 11. *For all strategies $\sigma' : A' \to A$ and $\sigma, \tau : A \to B$,*

- $\langle !; \hat{0}, \mathtt{id} \rangle; [\sigma, \tau] = \tau$ *and* $\langle !; \hat{1}, \mathtt{id} \rangle; [\sigma, \tau] = \sigma$*;*
- *if σ' is evaluated then* $(\mathtt{id}_{\mathbb{Z}} \times \sigma'); [\sigma, \tau] = [\sigma'; \sigma, \sigma'; \tau]$.

Method definitions in IMJ induce a form of exponentiation:

$$\frac{\bigwedge_{i=1}^{n} (\Delta | \Gamma \uplus \{\vec{x}_i : \vec{\theta}_i\} \vdash M_i : \theta_i)}{\Delta | \Gamma \vdash \mathcal{M} : \Theta} \quad \begin{array}{l} \Theta = \{\mathsf{m}_i : \vec{\theta}_i \to \theta_i \mid 1 \leq i \leq n\} \\ \wedge\, \mathcal{M} = \{\mathsf{m}_i : \lambda \vec{x}_i . M_i \mid 1 \leq i \leq n\} \end{array}$$

the modelling of which requires some extra semantic machinery. Traditionally, in call-by-value game models, exponentiation leads to 'effectless' strategies, corresponding to higher-order value terms. In our case, higher-order values are methods, manifesting themselves via the objects they may inhabit. Hence, exponentiation necessarily passes through generation of fresh object names containing these values. These considerations give rise to two classes of strategies introduced below.

We say that an even-length play $s \in P_{AB}$ is ***total*** if it is either empty or $s = m_A^{\Sigma} m_B^{\Sigma \uplus T} s'$ and:

- $T \in Sto_0$ and $\nu(m_B) \cap \nu(\Sigma) \subseteq \nu(m_A)$,
- if $s' = s'' m^{\Sigma'} n^{T'}$ and $a \in \mathsf{dom}(\Sigma) \setminus \nu(\gamma(m_A^{\Sigma_0} m_B^{\Sigma_0 \uplus T} s'))$, for $\Sigma_0 \in Sto_0$ such that $\gamma(m_A^{\Sigma_0} m_B^{\Sigma_0 \uplus T} s') \in P_{AB}$, then $a \notin \nu(n)$ and $T'(a) = \Sigma'(a)$.

We write P_{AB}^{t} for the set of total plays in AB. Thus, in total plays, the initial move m_A is immediately followed by a move m_B, and the initial store Σ is *invisible* to P in the sense that P cannot use its names nor their values. A strategy $\phi : A \to B$ is called ***single-threaded*** if it consists of total plays and satisfies the conditions:[3]

- for all $m_A^{\Sigma} \in P_{AB}$ there is $m_A^{\Sigma} m_B^{T} \in \phi$;
- if $m_A^{\Sigma}\, m_B^{\Sigma \uplus T} s \in \phi$ then $\gamma(m_A^{\Sigma_0} m_B^{\Sigma_0 \uplus T} s) \in \phi$, for $\Sigma_0 \in Sto_0$;
- if $m_A^{\Sigma} m_B^{\Sigma \uplus T} s\, \mathsf{call}\, a.\mathsf{m}(\vec{v})^{\Sigma'} s' \in \phi$ and $a \in \nu(T)$ then $s = \epsilon$.

Thus, single-threaded strategies reply to every initial move m_A^{Σ} with a move m_B^{T} which depends only on m_A (i.e. P does not *read* before playing). Moreover, m_B^{T} does not change the values of Σ (P does not *write*) and may introduce some fresh objects, albeit with default values. Finally, plays of single-threaded strategies consist of just one *thread*, where a thread is a total play in which there can be at most one call to names introduced by its second move.

Conversely, given a total play starting with $m_A^{\Sigma} m_B^{\Sigma \uplus T}$, we can extract its threads by tracing back for each move in s the method call of the object $a \in \nu(T)$ it is related to. Formally, for each total play $s = m_A^{\Sigma} m_B^{\Sigma \uplus T} s'$ with $|s'| > 0$, the *threader* move of s, written $\mathsf{thrr}(s)$, is given by induction:

- $\mathsf{thrr}(s' m^{\Sigma'}) = \mathsf{thrr}(s')$, if $p(m) = P$;
- $\mathsf{thrr}(s' \mathsf{call}\, a.\mathsf{m}(\vec{v})^{\Sigma'}) = \mathsf{call}\, a.\mathsf{m}(\vec{v})^{\Sigma'}$, if $a \in \nu(T)$;
- $\mathsf{thrr}(s' n^{T'} s'' \mathsf{call}\, a.\mathsf{m}(\vec{v})^{\Sigma'}) = \mathsf{thrr}(s' n^{T'})$, if $a \in P(s) \setminus \nu(T)$ and n introduces a.
- $\mathsf{thrr}(s' n^{T'} s'' m^{\Sigma'}) = \mathsf{thrr}(s' n^{T'})$, if $p(m) = O$ and n justifies m.

If $s = s' n^{T'} s''$ with $|s'| \geq 2$, we set $\mathsf{thrr}(n^{T'}) = \mathsf{thrr}(s' n^{T'})$. Then, the ***current thread*** of s is the subsequence of s containing only moves with the same threader move as s, that is, if $\mathsf{thrr}(s) = m^{\Sigma'}$ and $s = m_A^{\Sigma} m_B^{\Sigma \uplus T} s'$ then

$$\lceil s \rceil = m_A^{\Sigma} m_B^{\Sigma \uplus T} (s' \upharpoonright m^{\Sigma'})$$

where the restriction retains only those moves $n^{T'}$ of s' such that $\mathsf{thrr}(n^{T'}) = m^{\Sigma'}$. We extend this to the case of $|s| \leq 2$ by setting $\lceil s \rceil = s$. Finally, we call a total play $s \in P_{AB}$ ***thread-independent*** if for all $s' m^{\Sigma'} \sqsubseteq^{even} s$ with $|s'| > 2$:

- if $\gamma(\lceil s' m^{\Sigma'} \rceil) = s'' m^{\Sigma''}$ then $\nu(\Sigma'') \cap \nu(s') \subseteq \nu(s'')$;
- if s' ends in some $n^{T'}$ and $a \in \mathsf{dom}(\Sigma') \setminus \nu(\gamma(\lceil s' m^{\Sigma'} \rceil))$ then $\Sigma'(a) = T'(a)$.

We write P_{AB}^{ti} for the set of thread-independent plays in AB.

We can now define strategies which occur as interleavings of single-threaded ones. Let $\phi : A \to B$ be a single-threaded strategy. We define: $\phi^{\dagger} = \{s \in P_{AB}^{\mathrm{ti}} \mid \forall s' \sqsubseteq^{even} s.\ \gamma(\lceil s' \rceil) \in \phi\}$.

Lemma 12. *$\phi^{\dagger}$ is a strategy, for each single-threaded ϕ.*

Proof. Equivariance, Even-prefix closure and O-extension follow from the corresponding conditions on ϕ. For determinacy, if $sm^{\Sigma}, sn^{T} \in \phi^{\dagger}$ with $|s| > 0$ then, using determinacy of ϕ and the fact that P-moves do not change the current thread, nor do they modify or use names from other threads, we can show that $sm^{\Sigma} \sim sn^{T}$. □

We say that a strategy σ is ***thread-independent*** if $\sigma = \tau^{\dagger}$ for some single-threaded strategy τ. Thus, thread-independent strategies do not depend on initial stores and behave in each of their threads in an independent manner. Note in particular that evaluated strategies are thread-independent (and single-threaded).

Lemma 13. *Let $\sigma : A \to B$ and $\tau : A \to C$ be strategies with τ thread-independent. Then, $\langle \sigma, \tau \rangle; \pi_1 = \sigma$ and:*

$$\langle \sigma, \tau \rangle \;=\; A \xrightarrow{\langle \tau, \sigma \rangle} C \times B \xrightarrow{\cong} B \times C\,.$$

Proof. The former claim is straightforward. For the latter, we observe that the initial effects of σ and τ commute: on initial move m_A^{Σ}, τ does not read the store updates that σ includes in its response $m_B^{\Sigma'}$, while σ cannot access the names created by τ in its second move $m_C^{\Sigma' \uplus T}$. □

It is worth noting that the above lemma does not suffice for obtaining categorical products. Allowing thread-independent strategies to create fresh names in their second move breaks universality of pairings. Considering, for example, the strategy:

$$\sigma : 1 \to \mathcal{I} \times \mathcal{I} = \{*\, (a, a)^{\Sigma} \in P_{1(\mathcal{I} \times \mathcal{I})} \mid \Sigma \in Sto_0\}$$

we can see that $\sigma \neq \langle \sigma; \pi_1, \sigma; \pi_2 \rangle$, as the right-hand-side contains plays of the form $*\, (a, b)^{T}$ with $a \neq b$.

We can now define an appropriate notion of exponential for our games. Let us assume a translation assigning an arena $[\![\vec{\theta}]\!]$ to each type sequence $\vec{\theta}$. Moreover, let $\mathcal{I}$ be an interface such that

$$\Delta(\mathcal{I}) \upharpoonright Meths = \{\mathsf{m}_1 : \vec{\theta}_1 \to \theta_1, \cdots, \mathsf{m}_n : \vec{\theta}_n \to \theta_n\}$$

where $\vec{\theta}_i = \theta_{i1}, \cdots, \theta_{im_i}$, for each i. For any arena A, given single-threaded strategies $\phi_1, \cdots, \phi_n : A \to \mathcal{I}$ such that, for each i, if $m_A^{\Sigma} a^{\Sigma \uplus T} s \in \phi_i$ then

$$a \notin \nu(\Sigma) \wedge T(a) : \mathcal{I} \wedge (\mathsf{call}\, a.\mathsf{m}(\vec{v}) \in s \implies \mathsf{m} = \mathsf{m}_i),$$

we can group them into one single-threaded strategy:

$$\langle\langle \phi_1, \ldots, \phi_n \rangle\rangle : A \to \mathcal{I} = \bigcup\nolimits_{i=1}^{n} \phi_i\,.$$

Note that the a above is fresh for each m_A^{Σ} (i.e. $a \notin \nu(m_A^{\Sigma})$).

[3] Note that the use of the term "thread" here is internal to game semantics parlance and in particular should not be confused with Java threads.

Let now $\sigma_1, \cdots, \sigma_n$ be strategies with $\sigma_i : A \times [\![\vec{\theta_i}]\!] \to [\![\theta_i]\!]$. For each i, we define the single-threaded strategy $\Lambda(\sigma_i) : A \to \mathcal{I}$:

$$\Lambda(\sigma_i) = \{m_A^{\Sigma} a^{\Sigma \uplus T} \mathsf{call}\, a.m_i(\vec{v})^{\Sigma'} s \in P_{A\mathcal{I}}^{\mathrm{t}} \mid \gamma((m_A, \vec{v})^{\Sigma'} s) \in \sigma_i\}$$
$$\cup \{m_A^{\Sigma} a^{\Sigma \uplus T} \mathsf{call}\, a.m_i(\vec{v})^{\Sigma'} s\, \mathsf{ret}\, a.\mathsf{m}_i(v)^{T'} s' \in P_{A\mathcal{I}}^{\mathrm{t}} \mid$$
$$\gamma((m_A, \vec{v})^{\Sigma'} s\, v^{T'} s') \in \sigma_i\} \cup \{m_A^{\Sigma} a^{\Sigma \uplus T} \in P_{A\mathcal{I}}^{\mathrm{t}}\}$$

where $a \notin \nu(\Sigma, \vec{v}, v, s, s', \Sigma', T')$ and $T(a) : \mathcal{I}$. By definition, $\Lambda(\sigma_i)$ is single-threaded. Therefore, setting

$$\Lambda(\sigma_1, \ldots, \sigma_n) = \langle\langle \Lambda(\sigma_1), \ldots, \Lambda(\sigma_n) \rangle\rangle^{\dagger} : A \to \mathcal{I},$$

we obtain a thread-independent strategy implementing a simultaneous currying of $\sigma_1, \cdots, \sigma_n$. In particular, given translations $[\![M_i]\!]$ for each method in a method-set implementation $\mathcal{M}$, we can construct:

$$[\![\mathcal{M}]\!] : [\![\Gamma]\!] \to \mathcal{I} = \Lambda([\![M_1]\!], \cdots, [\![M_n]\!]).$$

Finally, we define *evaluation strategies* $\mathtt{ev}_{\mathsf{m}_i} : \mathcal{I} \times [\![\vec{\theta_i}]\!] \to [\![\theta_i]\!]$ by (taking even-length prefixes of):

$$\mathtt{ev}_{\mathsf{m}_i} = \{(a, \vec{v})^{\Sigma} \mathsf{call}\, a.\mathsf{m}_i(\vec{v})^{\Sigma} \mathsf{ret}\, a.\mathsf{m}_i(v)^{T} v^{T} \in P_{A_i} \mid \Sigma(a) \leq \mathcal{I}\}$$

where $A_i = (\mathcal{I} \times [\![\vec{\theta_i}]\!])[\![\theta_i]\!]$. We can now show the following natural mapping from groups of strategies in $A \times [\![\vec{\theta_i}]\!] \to [\![\theta_i]\!]$ to thread-independent ones in $A \to \mathcal{I}$.

Lemma 14. *Let $\sigma_1, \cdots, \sigma_n$ be as above, and let $\tau : A' \to A$ be evaluated. Then,*

- $\Lambda(\sigma_1, \ldots, \sigma_n) \times \mathtt{id}; \mathtt{ev}_{\mathsf{m}_i} = \sigma_i$,
- $\tau; \Lambda(\sigma_1, \ldots, \sigma_n) = \Lambda((\tau \times \mathtt{id}); \sigma_1, \ldots, (\tau \times \mathtt{id}); \sigma_n)$.

Apart from dealing with exponentials, in order to complete our translation we need also to address the appearance of $x : \mathcal{I}$ in the rule[4]

$$\frac{\Gamma, x : \mathcal{I}, \Delta \vdash \mathcal{M} : \Theta}{\Gamma, \Delta \vdash \mathsf{new}(x : \mathcal{I}; \mathcal{M}) : \mathcal{I}}\ \Delta(\mathcal{I}) \restriction Meths = \Theta.$$

Recall that

$$[\![\mathcal{M}]\!] : [\![\Gamma]\!] \times \mathcal{I} \to \mathcal{I} \tag{1}$$

is obtained using exponentiation. Thus, the second move of $[\![\mathcal{M}]\!]$ will appear in the right-hand-side $\mathcal{I}$ above and will be a fresh name b which will serve as a handle to the methods of $\mathcal{M}$: in order to invoke $\mathsf{m} : \lambda\vec{x}.M$ on input $\vec{v}$, the Opponent would have to call $b.\mathsf{m}(\vec{v})$. The remaining challenge is to merge the two occurrences of $\mathcal{I}$ in (1). We achieve this as follows. Let us assume a well-formed extension Δ' of Δ:

$$\Delta' = (\mathcal{I}' : (\mathsf{f}' : \mathcal{I})), \Delta$$

that is, $\mathcal{I}'$ contains a single field f' of type $\mathcal{I}$. We next define the strategy $\kappa_{\mathcal{I}} : 1 \to \mathcal{I}' \times \mathcal{I}$ of $\mathcal{G}_{\Delta'}$:

$$\kappa_{\mathcal{I}} = \{*\,(a', a)^{\Sigma_0} \mathsf{call}\, a.\mathsf{m}(\vec{v})^{\Sigma} \mathsf{call}\, b.\mathsf{m}(\vec{v})^{\Sigma} \mathsf{ret}\, b.\mathsf{m}(v)^{T} \mathsf{ret}\, a.\mathsf{m}(v)^{T}\}^{\dagger}$$

where $\mathsf{m} \in \mathsf{dom}(\Delta(\mathcal{I}))$, $b = \Sigma(a').\mathsf{f}'$, and $\Sigma_0 \in Sto_0$ is such that $\Sigma_0(a) : \mathcal{I}$ and $\Sigma_0(a') : \mathcal{I}'$. We let $[\![\mathsf{new}(x : \mathcal{I}; \mathcal{M})]\!]$ be the strategy:[5]

$$[\![\Gamma]\!] \xrightarrow{\langle \mathtt{id}, !; \kappa_{\mathcal{I}} \rangle; \cong} \mathcal{I}' \times [\![\Gamma]\!] \times \mathcal{I} \xrightarrow{\mathtt{id} \times \langle [\![\mathcal{M}]\!], \pi_2 \rangle} \mathcal{I}' \times \mathcal{I} \times \mathcal{I} \xrightarrow{(\mathtt{asn}_{\mathsf{f}'} \times \mathtt{id}); \pi_2} \mathcal{I}$$

and $\mathtt{asn}_{\mathsf{f}}$ is the assignment strategy:

$$\mathtt{asn}_{\mathsf{f}} : \mathcal{I} \times [\![\theta]\!] \to 1 = \{(a, v)^{\Sigma} *^{\Sigma[a.\mathsf{f} \mapsto v]} \in P_{(\mathcal{I} \times [\![\theta]\!])1}\},$$

for each field f. Thus, object creation involves creating a pair of names (a', a) with $a : \mathcal{I}$ and $a' : \mathcal{I}'$, where a is the name of the object we want to return. The name a' serves as a store where the handle of the method implementations, that is, the name created by the second move of $[\![\mathcal{M}]\!]$, will be passed. The strategy $\kappa_{\mathcal{I}}$, upon receiving a request $\mathsf{call}\, a.\mathsf{m}(\vec{v})^{\Sigma}$, simply forwards it to the respective method of $a'.\mathsf{f}'$ and, once it receives a return value, copies it back as the return value of the original call.

Let $\#(\vec{\mathcal{I}}) : \overrightarrow{\mathcal{I}} \to \#(\overrightarrow{\mathcal{I}}) = \{\vec{a}^{\Sigma}\vec{a}^{\Sigma} \mid a_i\text{s distinct}\}$, for each sequence of interfaces $\overrightarrow{\mathcal{I}}$. The latter has a right inverse $\#(\vec{\mathcal{I}})^{-r} : \#(\overrightarrow{\mathcal{I}}) \to \overrightarrow{\mathcal{I}}$ with the same plays. We can now define the semantic translation of terms.

Definition 10. The semantic translation is given as follows.

- Contexts $\Gamma = \{x_1 : \theta_1, \cdots, x_n : \theta_n\} \cup \{a_1 : \mathcal{I}_1, \cdots, a_m : \mathcal{I}_m\}$ are translated into arenas by
$$[\![\Gamma]\!] = [\![\theta_1]\!] \times \cdots \times [\![\theta_n]\!] \times \#(\mathcal{I}_1, \cdots, \mathcal{I}_m),$$
where $[\![\mathsf{void}]\!] = 1$, $[\![\mathsf{int}]\!] = \mathbb{Z}$ and $[\![\mathcal{I}]\!] = \mathcal{I}$.
- Terms are translated as in Figure 4 (top part).

In order to prove that the semantics is sound, we will also need to interpret terms inside state contexts. Let $\Gamma \vdash M : \theta$, with $\Gamma = \Gamma_1 \cup \Gamma_2$, where Γ_1 contains only variables and $\mathsf{dom}(\Gamma_2) = \mathsf{dom}(S)$. A term-in-state-context (S, M) is translated into the strategy:

$$[\![\Gamma_1 \vdash (S, M)]\!] = [\![\Gamma_1]\!] \xrightarrow{[\![S]\!]} [\![\Gamma_1]\!] \times \overrightarrow{\mathcal{I}} \xrightarrow{\mathtt{id} \times \#(\vec{\mathcal{I}})} [\![\Gamma]\!] \xrightarrow{[\![M]\!]} [\![\theta]\!].$$

The semantic translation of states (Figure 4, lower part), comprises two stages:

$$[\![\Gamma_1 \vdash S]\!] = [\![\Gamma_1]\!] \xrightarrow{[\![S]\!]_1} [\![\Gamma_1]\!] \times \overrightarrow{\mathcal{I}} \xrightarrow{[\![S]\!]_2} [\![\Gamma_1]\!] \times \overrightarrow{\mathcal{I}}.$$

The first stage, $[\![S]\!]_1$, creates the objects in $\mathsf{dom}(S)$ and implements their methods. The second stage of the translation, $[\![S]\!]_2$, initialises the fields of the newly created objects.

In the rest of this section we show soundness of the semantics. Let us call NEW, FIELDUP, FIELDAC and METHODCL respectively the transition rules in Figure 2 which involve state. Given a rule $\mathtt{r}$, we write $(S, M) \xrightarrow{\mathtt{r}} (S', M')$ if the transition $(S, M) \longrightarrow (S', M')$ involves applying $\mathtt{r}$ and context rules.

Proposition 15 (Correctness)**.** *Let (S, M) be a term-in-state-context and suppose $(S, M) \xrightarrow{\mathtt{r}} (S', M')$.*

1. *If the transition $\mathtt{r}$ is not stateful then $[\![M]\!] = [\![M']\!]$.*
2. *If $\mathtt{r}$ is one of* FIELDAC *or* FIELDUP *then $[\![S]\!]_2; (\mathtt{id} \times \#(\vec{\mathcal{I}})); [\![M]\!] = [\![S']\!]_2; (\mathtt{id} \times \#(\vec{\mathcal{I}})); [\![M']\!]$.*
3. *If $\mathtt{r}$ is one of* METHODCL *or* NEW *then $[\![(S, M)]\!] = [\![(S', M')]\!]$.*

Thus, in every case, $[\![(S, M)]\!] = [\![(S', M')]\!]$.

Proof. Claim 1 is proved by using the naturality results of this section. For the let construct, we show by induction on M that $[\![M[v/x]]\!] = \langle \mathtt{id}, [\![v]\!] \rangle; [\![M]\!]$. For 2 we use the following properties of field assignment and access:

$$\langle \mathtt{asn}_{\mathsf{f}}, \pi_1 \rangle; \pi_2; \mathtt{drf}_{\mathsf{f}} = \langle \mathtt{asn}_{\mathsf{f}}, \pi_2 \rangle; \pi_2 : \mathcal{I} \times [\![\theta]\!] \to [\![\theta]\!]$$
$$\langle \mathtt{asn}_{\mathsf{f}}, \pi_1 \rangle \times \mathtt{id}; \pi_2; \mathtt{asn}_{\mathsf{f}} = \mathtt{id} \times \pi_2; \mathtt{asn}_{\mathsf{f}} : \mathcal{I} \times [\![\theta]\!] \times [\![\theta]\!] \to 1$$

which are easily verifiable (the former one states that assigning a field value and accessing it returns the same value; the latter that two assignments in a row have the same effect as just the last one). The final claim follows by showing that the diagrams below

[4] Note that x may appear free in $\mathcal{M}$; it stands for the keyword this of Java.

[5] Here we omit wrapping $[\![\mathcal{M}]\!]$ inside Δ/Δ', as well as wrapping the whole $[\![\mathsf{new}(x : \mathcal{I}; \mathcal{M})]\!]$ in Δ'/Δ, for conciseness.

- $[\![\Gamma \vdash x_i : \theta_i]\!] = [\![\Gamma]\!] \xrightarrow{\pi_i} [\![\theta_i]\!]$;
- $[\![\Gamma \vdash a_i : \mathcal{I}_i]\!] = [\![\Gamma]\!] \xrightarrow{\pi_{n+1}} \#(\overrightarrow{\mathcal{I}}) \xrightarrow{\#(\vec{\mathcal{I}})^{-r}} \overrightarrow{\mathcal{I}} \xrightarrow{\pi_i} \mathcal{I}_i$;
- $[\![\Gamma \vdash \mathsf{skip} : \mathsf{void}]\!] = [\![\Gamma]\!] \xrightarrow{!} 1$;
- $[\![\Gamma \vdash \mathsf{null} : \mathcal{I}]\!] = [\![\Gamma]\!] \xrightarrow{!;\hat{\mathtt{nul}}} \mathcal{I}$, where $\hat{\mathtt{nul}} : 1 \to \mathcal{I} = \{* \, \mathtt{nul}\}$;
- $[\![\Gamma \vdash i : \mathsf{int}]\!] = [\![\Gamma]\!] \xrightarrow{!;\hat{i}} \mathbb{Z}$;
- $[\![\Gamma \vdash \mathsf{let}\, x = M' \,\mathsf{in}\, M : \theta]\!] = [\![\Gamma]\!] \xrightarrow{\langle \mathtt{id}, [\![M']\!] \rangle} [\![\Gamma]\!] \times [\![\theta']\!] \xrightarrow{[\![M]\!]} [\![\theta]\!]$;
- $[\![\Gamma \vdash (\mathcal{I})M : \mathcal{I}]\!] = [\![\Gamma]\!] \xrightarrow{[\![M]\!]} \mathcal{I}' \xrightarrow{\mathtt{stp}_{\mathcal{I}'\mathcal{I}}} \mathcal{I}$, where $\mathtt{stp}_{\mathcal{I}'\mathcal{I}} : \mathcal{I}' \to \mathcal{I} = \{\mathtt{nul}\,\mathtt{nul}\} \cup \{a^\Sigma a^\Sigma \in P_{\mathcal{I}'\mathcal{I}} \mid \Sigma(a) \leq \mathcal{I}\}$;
- $[\![\Gamma \vdash M \oplus M' : \mathsf{int}]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![M]\!], [\![M']\!] \rangle} \mathbb{Z} \times \mathbb{Z} \xrightarrow{\oplus} \mathbb{Z}$, where $\oplus : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z} = \{(i, j)\,(i \oplus j)\}$;
- $[\![\Gamma \vdash M = M' : \mathsf{int}]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![M]\!], [\![M']\!] \rangle} \mathcal{I} \times \mathcal{I} \xrightarrow{\mathtt{eq}} \mathbb{Z}$, where $\mathtt{eq} = \{(a, a)^\Sigma\, 1^\Sigma \in P_{(\mathcal{I}\times\mathcal{I})\mathbb{Z}}\} \cup \{(a, b)^\Sigma\, 0^\Sigma \in P_{(\mathcal{I}\times\mathcal{I})\mathbb{Z}} \mid a \neq b\}$;
- $[\![\Gamma \vdash \mathsf{if}\, M \,\mathsf{then}\, M' \,\mathsf{else}\, M'' : \theta]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![M]\!], \mathtt{id} \rangle} \mathbb{Z} \times [\![\Gamma]\!] \xrightarrow{[[\![M']\!], [\![M'']\!]]} [\![\theta]\!]$;
- $[\![\Gamma \vdash \mathsf{new}(x{:}\mathcal{I}; \mathcal{M}) : \mathcal{I}]\!] = [\![\Gamma]\!] \xrightarrow{\langle \mathtt{id}, !; \kappa_{\mathcal{I}} \rangle; \cong} \mathcal{I}' \times [\![\Gamma]\!] \times \mathcal{I} \xrightarrow{\mathtt{id} \times \langle [\![\mathcal{M}]\!], \pi_2 \rangle} \mathcal{I}' \times \mathcal{I} \times \mathcal{I} \xrightarrow{\mathtt{asn}_{\mathsf{f}'} \times \mathtt{id}} 1 \times \mathcal{I} \xrightarrow{\pi_2} \mathcal{I}$,
  where $[\![\mathcal{M}]\!] = [\![\Gamma]\!] \times \mathcal{I} \xrightarrow{\Lambda([\![M_1]\!], \ldots, [\![M_n]\!])} \mathcal{I}$ if $\mathcal{M} = \{\mathsf{m}_1 : \lambda \vec{x}_1.M_1, \cdots, \mathsf{m}_n : \lambda \vec{x}_n.M_n\}$;
- $[\![\Gamma \vdash M.\mathsf{f} := M' : \mathsf{void}]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![M]\!], [\![M']\!] \rangle} \mathcal{I} \times [\![\theta]\!] \xrightarrow{\mathtt{asn}_{\mathsf{f}}} 1$;
- $[\![\Gamma \vdash M.\mathsf{f} : \theta]\!] = [\![\Gamma]\!] \xrightarrow{[\![M]\!]} \mathcal{I} \xrightarrow{\mathtt{drf}_{\mathsf{f}}} [\![\theta]\!]$, where $\mathtt{drf}_{\mathsf{f}} : \mathcal{I} \to [\![\theta]\!] = \{a^\Sigma v^\Sigma \in P_{\mathcal{I}[\![\theta]\!]} \mid \Sigma(a).\mathrm{f} = v\}$;
- $[\![\Gamma \vdash M.\mathsf{m}(\overrightarrow{M}) : \theta]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![M]\!], [\![\vec{M}]\!] \rangle} \mathcal{I} \times [\![\vec{\theta}]\!] \xrightarrow{\mathtt{ev}_{\mathsf{m}}} [\![\theta]\!]$, where $[\![\vec{M}]\!] = \langle\langle [\![M_1]\!], [\![M_2]\!] \rangle, \cdots \rangle, [\![M_n]\!] \rangle$.

- $[\![\Gamma_1 \vdash S]\!] = [\![\Gamma_1]\!] \xrightarrow{\langle \mathtt{id}, \overrightarrow{\kappa_{\mathcal{I}}} \rangle} [\![\Gamma_1]\!] \times \overrightarrow{(\mathcal{I}' \times \mathcal{I})} \xrightarrow{\cong} \overrightarrow{\mathcal{I}'} \times ([\![\Gamma_1]\!] \times \overrightarrow{\mathcal{I}}) \xrightarrow{\mathtt{id} \times \langle \pi_2; [\![\overrightarrow{\mathcal{M}}]\!], \mathtt{id} \rangle} \overrightarrow{\mathcal{I}'} \times \overrightarrow{\mathcal{I}} \times ([\![\Gamma_1]\!] \times \overrightarrow{\mathcal{I}}) \xrightarrow{\cong \times \mathtt{id}} \overrightarrow{(\mathcal{I}' \times \mathcal{I})} \times ([\![\Gamma_1]\!] \times \overrightarrow{\mathcal{I}})$
  $\xrightarrow{(\overrightarrow{\mathtt{asn}_{\mathsf{f}'}} \times \mathtt{id}); \pi_2} [\![\Gamma_1]\!] \times \overrightarrow{\mathcal{I}} \xrightarrow{\langle \mathtt{id}, \langle \overrightarrow{\mathtt{id}}, [\![\vec{V}]\!] \rangle \rangle} ([\![\Gamma_1]\!] \times \overrightarrow{\mathcal{I}}) \times \overrightarrow{\mathcal{I}} \times \overrightarrow{[\![\theta]\!]} \xrightarrow{\mathtt{id} \times \cong} ([\![\Gamma_1]\!] \times \overrightarrow{\mathcal{I}}) \times \overrightarrow{(\mathcal{I} \times [\![\theta]\!])} \xrightarrow{(\mathtt{id} \times \overrightarrow{\mathtt{asn}_{\mathsf{f}}}); \pi_1} [\![\Gamma_1]\!] \times \overrightarrow{\mathcal{I}}$,

  where $\mathsf{dom}(S) = \{a_1, \cdots, a_n\}$, $\overrightarrow{\kappa_{\mathcal{I}}} = \langle \kappa_{\mathcal{I}_1}, \cdots, \kappa_{\mathcal{I}_n} \rangle$, $S(a_i) : \mathcal{I}_i$, $\overrightarrow{\mathcal{I}'} = \mathcal{I}'_1 \times \cdots \times \mathcal{I}'_n$, $\overrightarrow{\mathcal{I}} = \mathcal{I}_1 \times \cdots \times \mathcal{I}_n$, $\overrightarrow{\mathcal{M}} = (\mathcal{M}_1, \cdots, \mathcal{M}_n)$, $\mathcal{M}_i = S(a_i) \upharpoonright \mathit{MImps}$, $[\![\overrightarrow{\mathcal{M}}]\!] = \langle [\![\mathcal{M}_1]\!], \cdots, [\![\mathcal{M}_n]\!] \rangle$, $\mathtt{asn}_{\mathsf{f}'} = \mathtt{asn}_{\mathsf{f}'_1} \times \cdots \times \mathtt{asn}_{\mathsf{f}'_n}$, $\overrightarrow{V} = (V_1, \cdots, V_n)$, $V_i = S(a_i) \upharpoonright \mathit{HCnfs}$, $[\![\overrightarrow{V}]\!] = \langle [\![V_1]\!], \cdots, [\![V_n]\!] \rangle$, $V_i = (\mathsf{f}_i^1 : v_i^1, \cdots, \mathsf{f}_i^{n_i} : v_i^{n_i})$, $[\![V_i]\!] = \langle [\![v_i^1]\!], \cdots, [\![v_i^{n_i}]\!] \rangle$, $\overrightarrow{\mathtt{asn}_{\mathsf{f}}} = \mathtt{asn}_{\mathsf{f}_i} \times \cdots \times \mathtt{asn}_{\mathsf{f}_n}$, $\mathtt{asn}_{\mathsf{f}_i} = \mathtt{asn}_{\mathsf{f}_i^1} \times \cdots \times \mathtt{asn}_{\mathsf{f}_i^{n_i}}$.

Figure 4. The semantic translation of IMJ.

commute (we write A for $[\![\Gamma]\!] \times \overrightarrow{\mathcal{I}}$),

$$\begin{array}{ccccc}
\overrightarrow{\mathcal{I}'} \times A \times [\![\vec{\theta}]\!] & \xrightarrow{\mathtt{id} \times [\![\overrightarrow{\mathcal{M}}]\!] \times \mathtt{id}} & \overrightarrow{\mathcal{I}'} \times \overrightarrow{\mathcal{I}} \times [\![\vec{\theta}]\!] & \xrightarrow{\langle \cong, \pi_2; \pi_i \rangle \times \mathtt{id}} & \overrightarrow{(\mathcal{I}' \times \mathcal{I})} \times \mathcal{I}_i \times [\![\vec{\theta}]\!] \\
\downarrow{\scriptstyle \mathtt{id} \times \langle [\![\overrightarrow{\mathcal{M}}]\!], [\![\mathcal{M}_i]\!] \rangle \times \mathtt{id}} & & & & \downarrow{\scriptstyle (\overrightarrow{\mathtt{asn}_{\mathsf{f}'}} \times \mathtt{ev}_{\mathsf{m}}); \pi_2} \\
\overrightarrow{\mathcal{I}'} \times \overrightarrow{\mathcal{I}} \times \mathcal{I}_i \times [\![\vec{\theta}]\!] & \xrightarrow{\cong \times \mathtt{id}} & \overrightarrow{(\mathcal{I}' \times \mathcal{I})} \times \mathcal{I}_i \times [\![\vec{\theta}]\!] & \xrightarrow{(\overrightarrow{\mathtt{asn}_{\mathsf{f}'}} \times \mathtt{ev}_{\mathsf{m}}); \pi_2} & [\![\theta]\!]
\end{array}$$

$$\begin{array}{ccccc}
[\![\Gamma_1]\!] & & & & \\
\downarrow{\scriptstyle \chi} & & & & \\
\overrightarrow{\mathcal{I}'} \times A \times A & \xrightarrow{\mathtt{id} \times \langle \pi_2; \pi_i, \sigma' \rangle} & \overrightarrow{\mathcal{I}'} \times A \times \mathcal{I}_i \times [\![\vec{\theta}]\!] & \xrightarrow{\mathtt{id} \times [\![\overrightarrow{\mathcal{M}}]\!] \times \mathtt{id}} & \overrightarrow{\mathcal{I}'} \times \overrightarrow{\mathcal{I}} \times \mathcal{I}_i \times [\![\vec{\theta}]\!] \\
\downarrow{\scriptstyle \mathtt{id} \times \sigma'} & & & & \cong \times \mathtt{id} \downarrow \\
\overrightarrow{\mathcal{I}'} \times A \times [\![\vec{\theta}]\!] & & & & \overrightarrow{(\mathcal{I}' \times \mathcal{I})} \times \mathcal{I}_i \times [\![\vec{\theta}]\!] \\
\downarrow{\scriptstyle \mathtt{id} \times [\![\overrightarrow{\mathcal{M}}]\!] \times \mathtt{id}} & & & & (\overrightarrow{\mathtt{asn}_{\mathsf{f}'}} \times \mathtt{ev}_{\mathsf{m}}); \pi_2 \downarrow \\
\overrightarrow{\mathcal{I}'} \times \overrightarrow{\mathcal{I}} \times [\![\vec{\theta}]\!] & \xrightarrow{\langle \cong, \pi_2; \pi_i \rangle \times \mathtt{id}} & \overrightarrow{(\mathcal{I}' \times \mathcal{I})} \times \mathcal{I}_i \times [\![\vec{\theta}]\!] & \xrightarrow{(\overrightarrow{\mathtt{asn}_{\mathsf{f}'}} \times \mathtt{ev}_{\mathsf{m}}); \pi_2} & [\![\theta]\!]
\end{array}$$

where $\sigma' : A \to [\![\vec{\theta}]\!]$ a combination of values and assignments, and

$$\chi = [\![\Gamma_1]\!] \xrightarrow{\langle \mathtt{id}, \overrightarrow{\kappa_{\mathcal{I}}} \rangle} [\![\Gamma_1]\!] \times \overrightarrow{\mathcal{I}'} \times \overrightarrow{\mathcal{I}} \xrightarrow{(\delta \times \mathtt{id} \times \delta); \cong} \overrightarrow{\mathcal{I}'} \times A \times A$$

with $\delta = \langle \mathtt{id}, \mathtt{id} \rangle$. The former diagram says that, assigning method implementations $\overrightarrow{\mathcal{M}}$ to object stores $\vec{a}'$ and calling $\mathcal{M}_i$ on some method m is the same as assigning $\overrightarrow{\mathcal{M}}$ to $\vec{a}'$ and evaluating instead a new copy of $\mathcal{M}_i$ on m. The reason the diagram commutes is that the copy of $\mathcal{M}_i$ differs from the original just in the handle name (the one returned in the codomain of $[\![\mathcal{M}_i]\!]$), but the latter is hidden via composition with $\mathtt{ev}_{\mathsf{m}}$. The latter diagram stipulates that if we create $\vec{a}$ with methods $\overrightarrow{\mathcal{M}}$, then calling a_i on m is the same as calling $\mathcal{M}_i$ on m. The latter holds because of the way that $\kappa_{\mathcal{I}_i}$ manipulates calls inside the interaction, by delegating calls to methods of a_i to $\mathcal{M}_i$. □

Proposition 16 (Computational Soundness)**.** *For all* $\vdash M : \mathsf{void}$*, if* $M \Downarrow$ *then* $[\![M]\!] = \{**\}$ *(i.e.* $[\![M]\!] = [\![\mathsf{skip}]\!]$*).*

Proof. This directly follows from Correctness. □

Proposition 17 (Computational Adequacy)**.** *For all* $\vdash M : \mathsf{void}$*, if* $[\![M]\!] = \{**\}$ *then* $M \Downarrow$*.*

Proof. Suppose, for the sake of contradiction, that $[\![M]\!] = \{**\}$ and $M \not\Downarrow$. We notice that, by definition of the translation for blocking constructs (castings and conditionals may block) and due to Correctness, if $M \not\Downarrow$ were due to some reduction step being blocked then the semantics would also block. Thus, $M \not\Downarrow$ must be due to divergence. Now, the reduction relation restricted to all rules but METHODCL is strongly normalising, as each transition decreases the size of the term. Hence, if M diverges then it must involve infinitely many METHODCL reductions and our argument below shows that the latter would imply $[\![M]\!] = \{\epsilon\}$.
For any term $\Gamma \vdash N : \theta$ and $a \in \mathbb{A} \setminus \mathsf{dom}(\Gamma)$, construct $\Gamma_a \vdash N_a$, where $\Gamma_a = \Gamma \uplus \{a : \mathsf{Var}_I\}$, by recursively replacing each subterm of N of the shape $N'.\mathsf{m}(\vec{N})$ with $a.\mathsf{f} := (a.\mathsf{f} + 1); N'.\mathsf{m}(\vec{N})$.

Var_I is an interface with a sole field $\mathsf{f} : \mathsf{int}$. Observe that each $s \in [\![\Gamma \vdash N]\!]$ induces some $s' \in [\![\Gamma_a \vdash N_a]\!]$ 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 $a.\mathsf{f}$, while P never decreases the value of $a.\mathsf{f}$. We write $[\![\Gamma_a \vdash N_a]\!]_a$ for the subset of $[\![\Gamma_a \vdash N_a]\!]$ containing precisely these plays. Then, take M_0 to be the term $\mathsf{let}\, x = \mathsf{new}(x : \mathsf{Var}_I;)\, \mathsf{in}\, (M_a[x/a]; x.\mathsf{f})$, where x a fresh variable. Because $** \in [\![M]\!]$, we get $*j \in [\![M_0]\!]$ for some $j \in \mathbb{Z}$. Consider now the infinite reduction sequence of $(\emptyset, M)$. It must have infinitely many METHODCL steps, so suppose $(\emptyset, M) \longrightarrow^* (S, M')$ contains $j+1$ such steps. Then, we obtain $(\emptyset, M_0) \longrightarrow^* (S_a, (M')_a; a.\mathsf{f})$, with $S_a(a).\mathsf{f} = j+1$. By Correctness, we have that $*j \in [\![S_a, (M')_a; a.\mathsf{f}]\!] = [\![S_a]\!]; (\mathtt{id}\times\#); [\![(M')_a; a.\mathsf{f}]\!]_a$. Since in $[\![(M')_a]\!]_a$ the value of a cannot decrease, and its initial value is $j+1$ (as stipulated by S_a), we reach a contradiction. □

5. Full Abstraction

Recall that, given plays s, s', we call s an O-extension of s' (written $s \leq_O s'$) if s, s' are identical except the type information regarding O-names present in stores: the types of O-names in s may be subtypes of those in s'. We shall write $s \leq_P s'$ for the dual notion involving P-names, i.e., $s \leq_P s'$ if s, s' are the same, but the types of P-names in s' may be subtypes of those in s. Then, given $X \in \{O, P\}$ and fixed A, B, let us define $\mathsf{cl}_X(s) = \{s' \in P_{AB} \mid s' \leq_X s\}$ and $\mathsf{cl}_X(\sigma) = \bigcup_{s\in\sigma} \mathsf{cl}_X(s)$. We write $P_{\Delta|\Gamma\vdash\theta}$ for $P_{[\![\Gamma]\!][\![\theta]\!]}$. A play will be called *complete* if it is of the form $m_A Y m_B Y$.

Next we establish a definability result stating that any complete play (together with other plays implied by O-closure) originates from a term.

Lemma 18 (Definability)**.** *Let $s \in P_{\Delta|\Gamma\vdash\theta}$ be a complete play. There exists $\Delta' \supseteq \Delta$ and $\Delta'|\Gamma \vdash M : \theta$ such that $[\![\Delta'|\Gamma \vdash M : \theta]\!] = \mathsf{cl}_O(s)$.*

Proof. The argument proceeds by induction on $|s|$. For $s = \epsilon$, any divergent term suffices. For example, one can take $\Delta' = \Delta \oplus \{\mathsf{Div} \mapsto (\mathsf{m} : \mathsf{void} \to \mathsf{void})\}$, and pre-compose any term of type θ with $\mathsf{new}(x : \mathsf{Div};\, \mathsf{m} : \lambda().\mathsf{m}()\,).\mathsf{m}()$.

Suppose $s \neq \epsilon$. Then the second move can be a question or an answer. We first show how to reduce the former case to the latter, so that only the latter needs to be attacked directly.

Suppose

$$s = q^{\Sigma_q}\ \mathsf{call}\, o.\mathsf{m}(\vec{u})^{\Sigma_1}\ s_1\ \mathsf{ret}\, o.\mathsf{m}(v)^{\Sigma_2}\ s_2\ w^{\Sigma_3}\ s_3,$$

where $o : \mathcal{I}'$ and $\Delta(\mathcal{I}')(\mathsf{m}) : \overrightarrow{\mathcal{I}_L} \to \mathcal{I}_R$. Consider $\Delta' = \Delta \oplus \{\mathcal{I}'' \mapsto (\overrightarrow{\mathsf{f}' : \mathcal{I}_L},\ \mathsf{m}' : \mathcal{I}_R \to \theta)\}$ and the following play from $P_{\Delta'|\Gamma\vdash\mathcal{I}''}$:

$$s' = q^{\Sigma_q}\ p^{\Sigma'_1}\ s'_1\ \mathsf{call}\, p.\mathsf{m}'(v)^{\Sigma'_2}\ s'_2\ \mathsf{ret}\, p.\mathsf{m}'(v)^{\Sigma'_3}\ s'_3,$$

where $p \notin \nu(s)$, $\Sigma'_i = \Sigma_i \oplus \Sigma$, $\Sigma = \{p \mapsto (\mathcal{I}'', \overrightarrow{f' \mapsto u})\}$ and s'_j is the same as s_j except that each store is extended by Σ. If $\Delta'|\Gamma \vdash M' : \mathcal{I}'$ satisfies the Lemma for s' then, for s, one can take $\mathsf{let}\, x_p = M'\, \mathsf{in}\, x_p.\mathsf{m}'(y.\mathsf{m}(\overrightarrow{x_p.f'}))$, where y refers to o, i.e., y is of the shape $x.\overrightarrow{f}$, where $x \in \mathsf{dom}\,\Gamma$ and $\overrightarrow{f}$ is a sequence of fields that points at o in Σ_q.

Thanks to the reduction given above we can now assume that $s \in P_{\Delta|\Gamma\vdash\theta}$ is non-empty and

$$s = q^{\Sigma_q}\ m_0^{\Sigma_0}\ m_1^{\Sigma_1} \cdots m_{2k}^{\Sigma_{2k}},$$

where m_0 is an answer. We are going to enrich s in two ways so that it is easier to decompose. Ultimately, the decomposition of s will be based on the observation that the $m_1^{\Sigma_1} \cdots m_{2k}^{\Sigma_{2k}}$ segment can be viewed as an interleaving of threads, each of which is started by a move of the form $\mathsf{call}\, p$ for some P-name p. A thread consists of the starting move and is generated according to the following two rules: m_{2i} belongs to the thread of m_{2i-1} and every answer-move belongs to the same thread as the corresponding question-move.

- The first transformation of s brings forward the point of P-name creation to the second move. In this way, threads will never create objects and, consequently, it will be possible to compose them without facing the problem of object fusion.
  Suppose $P(s) = \overrightarrow{p_i}$ and $p_i : \mathcal{I}_{p_i}$. Let $\Delta' = \Delta \oplus \{\mathcal{I}_P \mapsto \overrightarrow{f_i : \mathcal{I}_{p_i}}\}$. Consider $s' = (n, q)^{\Sigma'_q}\ m_0^{\Sigma'_0}\ m_1^{\Sigma'_1} \cdots m_{2k}^{\Sigma'_{2k}}$, where $\Sigma'_q = \Sigma_q \oplus \{n \mapsto (\mathcal{I}_P, \overrightarrow{\mathsf{null}})\}$ and $\Sigma'_i = \Sigma_i \oplus \{n \mapsto (\mathcal{I}_P, \overrightarrow{p_i})\} \oplus \{p_i \mapsto (\mathcal{I}_{p_i}, \overrightarrow{\mathsf{null}}) \mid \Sigma_i(p_i)\ \text{undefined},\ p_i \in P(s)\}$. Let $\Gamma' = \{x_n : \mathcal{I}_P\} \oplus \Gamma$. Observe that $s' \in P_{\Delta'|\Gamma'\vdash\theta}$.
- The second transformation consists in storing the unfolding play in a global variable. It should be clear that the recursive structure of types along with the ability to store names is sufficient to store plays in objects. Let $\mathcal{I}_{play}$ be a signature that makes this possible. This will be used to enforce the intended interleaving of threads after their composition (in the style of Innocent Factorization [5]). Let $\Delta'' = \Delta' \oplus \{\mathsf{History} \mapsto \mathsf{play} : I_{play}\}$ and $\Gamma'' = \{x_h : \mathsf{History}\} \oplus \Gamma$. Consider

  $$s'' = (h, n, q)^{\Sigma''_q}\ m_0^{\Sigma''_0}\ m_1^{\Sigma''_1} \cdots m_{2k}^{\Sigma''_{2k}}$$

  with

  $$\begin{array}{rcl} \Sigma''_q & = & \Sigma'_q \oplus \{h \mapsto (\mathsf{History}, \mathsf{play} \mapsto \mathsf{null})\}, \\ \Sigma''_{2i} & = & \Sigma'_{2i} \oplus \{h \mapsto (\mathsf{History}, \mathsf{play} \mapsto s'_{\leq m_{2i}})\}, \\ \Sigma''_{2i+1} & = & \Sigma'_{2i+1} \oplus \{h \mapsto (\mathsf{History}, \mathsf{play} \mapsto s'_{\leq m_{2i}})\}. \end{array}$$

  Observe that $s'' \in P_{\Delta''|\Gamma'\vdash\theta}$.

Now we shall decompose $m_1^{\Sigma''_1} \cdots m_{2k}^{\Sigma''_{2k}}$ into threads. Recall that each of them is a subsequence of s'' of the form

$$\mathsf{call}\, p.\mathsf{m}(\overrightarrow{u})^{\Sigma_c} \quad t \quad \mathsf{ret}\, p.\mathsf{m}(v)^{\Sigma_r}$$

where the segment t contains moves of the form $\mathsf{call}\, o$ or $\mathsf{ret}\, o$ for some $o \in O(s)$. We would now like to invoke the IH for each thread but, since a thread is not a play, we do so for the closely related play $(h, n, q, \overrightarrow{u})^{\Sigma_c}\ t'\ v^{\Sigma_r}$. Let us call the resultant term $M_{p,\mathsf{m},\overrightarrow{u},\Sigma_c}$. Next we combine terms related to the same $p : \mathcal{I}_p$ into an object definition by

$$M_p \equiv \mathsf{new}(x : I_p; \mathsf{m} : \lambda\overrightarrow{u}.\mathsf{case}(\overrightarrow{u}, \Sigma_c)[M_{p,\mathsf{m},\overrightarrow{u},\Sigma_c}]).$$

The case statement, which can be implemented in IMJ using nested if's, is needed to recognize instances of $\overrightarrow{u}$ and Σ_c that really occur in threads related to p. In such cases the corresponding term $M_{p,\mathsf{m},\overrightarrow{u},\Sigma_c}$ will be run. Otherwise, the statement leads to divergence.

The term M for s can now be obtained by taking

$$\begin{array}{l} \mathsf{let}\, x_n = \mathsf{new}(x : \mathcal{I}_P;)\, \mathsf{in} \\ \mathsf{let}\, x_h = \mathsf{new}(x : \mathsf{History};)\, \mathsf{in} \\ \mathsf{let}\, \overrightarrow{x_{p_i} = M_{p_i}}\, \mathsf{in} \\ \qquad \mathsf{assert}(q^{\Sigma_q}); \overrightarrow{x_n.f_i = x_{p_i}}; \mathsf{make}(\Sigma''_0); \mathsf{play}(m_0) \end{array}$$

where $\overrightarrow{x_{p_i} = M_{p_i}}$ represents a series of bindings (one for each P-name $p_i \in P(s)$), $\mathsf{assert}((h, n, q)^{\Sigma''_q})$ is a conditional that converges if and only if the initial values of free Γ identifiers as well as values accessible through them are consistent with q and Σ_q respectively, $\mathsf{make}(\Sigma''_0)$ is a sequence of assignments that set values to those specified in Σ''_0 (up-casts need to be performed to ensure typability) and $\mathsf{play}(m_0)$ is skip, i, null or, if m_0 is a name,

it is a term of the form $(\theta)y.\vec{f}$, where y is x_n or $(x : I_x) \in \Gamma$ such that $y.\vec{f}$ gives an access path to m_0 in Σ_0''. □

We conclude with full abstraction results both in inequational and equational forms. For technical convenience, we shall use a modified (but equivalent) definition of contextual approximation.

Lemma 19. *Let* $\Gamma = \{x_1 : \mathcal{I}_1, \cdots, x_k : \mathcal{I}_k\}$, $\Delta|\Gamma \vdash M_i : \theta$ *$(i = 1, 2)$, and* $\Delta' = \Delta \cup \{\mathsf{Wrap}_{\Gamma,\mathcal{I}} \mapsto (f : (\mathcal{I}_1, \cdots, \mathcal{I}_k) \to \theta)\}$. *Then* $\Delta|\Gamma \vdash M_1 \precsim M_2$ *if and only if, for all* $\Delta'' \supseteq \Delta'$ *and* $\Delta'', z : \mathsf{Wrap}_{\Gamma,\mathcal{I}} \vdash \mathsf{test} : \mathsf{void}$, *if* $C_{\mathsf{test}}[M_1] \Downarrow$ *then* $C_{\mathsf{test}}[M_2] \Downarrow$, *where* $C_{\mathsf{test}}[-] \equiv \mathsf{let}\ z = \mathsf{new}(x : \mathsf{Wrap}_{\Gamma,\mathcal{I}}; f : \lambda\overrightarrow{x_i}.[-])\ \mathsf{in}\ \mathsf{test}$.

Proof. The Lemma holds because, on the one hand, it relies on contexts of a specific shape and, on the other hand, any closing context $C[-]$ for M_i can be presented in the above form with $\mathsf{test} \equiv C[z.f(x_1, \cdots, x_k)]$. □

Given a term $\Delta|\Gamma \vdash M : \theta$, let us write $[\![\Delta|\Gamma \vdash M : \theta]\!]_{comp}$ for the set of complete plays from $[\![\Delta|\Gamma \vdash M : \theta]\!]$. In what follows, we shall often omit $\Delta|\Gamma, \vdash$ for brevity.

Theorem 20 (Inequational full abstraction)**.** *Given* $\Delta|\Gamma \vdash M_i : \theta$ *$(i = 1, 2)$, we have* $\Delta|\Gamma \vdash M_1 \precsim M_2 : \theta$ *if and only if*

$$\mathsf{cl}_P([\![\Delta|\Gamma \vdash M_1 : \theta]\!]_{comp}) \subseteq \mathsf{cl}_P([\![\Delta|\Gamma \vdash M_2 : \theta]\!]_{comp}).$$

Proof. The proof uses the following play transformation. Given $t = q^{\Sigma_q} s_1 a^{\Sigma_a} s_2 \in P_{\Delta|\Gamma\vdash\theta}$, we define $\bar{t} \in P_{\Delta',\mathsf{Wrap}_{\Gamma,\mathcal{I}}\vdash\mathsf{void}}$ as

$$n^{\Sigma_n}\ \mathsf{call}\, n.f(q)^{\Sigma_q \oplus \Sigma_n}\ s_1^{\oplus\Sigma_n}\ \mathsf{ret}\, n.f(a)^{\Sigma_a\oplus\Sigma_n}\ s_2^{\oplus\Sigma_n}\ *^{\Sigma\oplus\Sigma_n},$$

where $\Delta', \mathsf{Wrap}_{\Gamma,\mathcal{I}}$ are the same as in the above Lemma, $\Sigma_n = \{n \mapsto (\mathsf{Wrap}_{\Gamma,\mathcal{I}}, \emptyset)\}$, $s^{\oplus\Sigma_n}$ stands for s in which each store was augmented by Σ_n and Σ is the store of the last move in t. Intuitively, $\bar{t}$ is the play that $C_{\mathsf{test}}[-]$ needs to provide for a terminating interaction with t.

(⇒) Let $s \in \mathsf{cl}_P([\![M_1]\!]_{comp})$. Then there exists $s' \in [\![M_1]\!]_{comp}$ with $s \in \mathsf{cl}_P(s')$. Apply Definability to $\overline{s'}$ to obtain $\Delta'', z : \mathsf{Wrap}_{\Gamma,\mathcal{I}} \vdash \mathsf{test} : \mathsf{void}$ such that $[\![\mathsf{test}]\!] = \mathsf{cl}_O(\overline{s'})$. Because $s' \in [\![M_1]\!]_{comp}$ and Adequacy holds, we must have $C_{\mathsf{test}}[M_1] \Downarrow$. From $M_1 \precsim M_2$ we obtain $C_{\mathsf{test}}[M_2] \Downarrow$. Hence, because of Soundness, there exists $s'' \in [\![M_2]\!]_{comp}$ such that $\overline{s''} \in [\![\mathsf{test}]\!]$. Since $[\![\mathsf{test}]\!] = \mathsf{cl}_O(\overline{s'})$, it follows that $\overline{s''} \in \mathsf{cl}_O(\overline{s'})$ and, consequently, $s' \in \mathsf{cl}_P(s'')$. Thus, $s \in \mathsf{cl}_P(s')$ and $s' \in \mathsf{cl}_P(s'')$. Hence, $s \in \mathsf{cl}_P(s'')$ and, because $s'' \in [\![M_2]\!]_{comp}$, we can conclude $s \in \mathsf{cl}_P([\![M_2]\!]_{comp})$.

(⇐) Let $C_{\mathsf{test}}[-]$ be such that $C_{\mathsf{test}}[M_1] \Downarrow$. By Soundness, there exists $s \in [\![M_1]\!]_{comp}$ such that $\overline{s} \in [\![\mathsf{test}]\!]$. Because $[\![M_1]\!]_{comp} \subseteq \mathsf{cl}_P([\![M_1]\!]_{comp})$ and $\mathsf{cl}_P([\![M_1]\!]_{comp}) \subseteq \mathsf{cl}_P([\![M_2]\!]_{comp})$, we also have $s \in \mathsf{cl}_P([\![M_2]\!]_{comp})$. Thus, there exists $s' \in [\![M_2]\!]_{comp}$ such that $s \in \mathsf{cl}_P(s')$. Consequently, $\overline{s'} \in \mathsf{cl}_O(\overline{s})$. Since $\overline{s} \in [\![\mathsf{test}]\!]$, we also have $\overline{s'} \in [\![\mathsf{test}]\!]$. Because $s' \in [\![M_2]\!]_{comp}$ and $\overline{s'} \in [\![\mathsf{test}]\!]$, by Adequacy, we can conclude that $C_{\mathsf{test}}[M_2] \Downarrow$. □

Example 3. Let us revisit Example 2. We have $\mathsf{cl}_P(\sigma_1) = \sigma_1$ and $\mathsf{cl}_P(\sigma_2) = \sigma_2 \cup \{*^{\emptyset}, n^{\{n\mapsto(\mathsf{Empty},\emptyset)\}}\}$, i.e. $\mathsf{cl}_P(\sigma_1) \subsetneq \mathsf{cl}_P(\sigma_2)$. Thus, it follows from Theorem 20 that $\Delta|\emptyset \vdash M_1 \precsim M_2$ and $\Delta|\emptyset \vdash M_1 \not\cong M_2$.

Theorem 21 (Equational full abstraction)**.** *Given* $\Delta|\Gamma \vdash M_i : \theta$ *$(i = 1, 2)$,* $\Delta|\Gamma \vdash M_1 \cong M_2 : \theta$ *if and only if*

$$[\![\Delta|\Gamma \vdash M_1 : \theta]\!]_{comp} = [\![\Delta|\Gamma \vdash M_2 : \theta]\!]_{comp}.$$

Proof. The preceding result implies that $M_1 \cong M_2$ if and only if $\mathsf{cl}_P([\![M_1]\!]_{comp}) = \mathsf{cl}_P([\![M_2]\!]_{comp})$. We show that this implies $[\![M_1]\!]_{comp} = [\![M_2]\!]_{comp}$. Let $s \in [\![M_1]\!]_{comp}$. By $\mathsf{cl}_P([\![M_1]\!]_{comp}) = \mathsf{cl}_P([\![M_2]\!]_{comp})$, it must be the case that $s \in \mathsf{cl}_P([\![M_2]\!]_{comp})$, i.e., there exists $s' \in [\![M_2]\!]_{comp}$ such that $s \in \mathsf{cl}_P(s')$. Again, by $\mathsf{cl}_P([\![M_1]\!]_{comp}) = \mathsf{cl}_P([\![M_2]\!]_{comp})$, it follows that $s' \in \mathsf{cl}_P([\![M_1]\!]_{comp})$, i.e., there exists $s'' \in [\![M_1]\!]_{comp}$ such that $s' \in \mathsf{cl}_P(s'')$. So, we have $s \in \mathsf{cl}_P(s')$ and $s' \in \mathsf{cl}_P(s'')$, which implies $s \in \mathsf{cl}_P(s'')$. However, $s, s'' \in [\![M_1]\!]_{comp}$, so $s \in \mathsf{cl}_P(s'')$ entails $s = s''$. Hence, $s \in \mathsf{cl}_P(s')$ and $s' \in \mathsf{cl}_P(s)$, and $s = s'$ follows. Because $s' \in [\![M_2]\!]_{comp}$, we showed $s \in [\![M_2]\!]_{comp}$. The other inclusion is derived analogously. □

References

[1] M. Abadi and L. Cardelli. *A theory of objects*. Springer Verlag, 1996.

[2] E. Ábraham, M. M. Bonsangue, F. S. de Boer, A. Gruener, and M. Steffen. Observability, connectivity, and replay in a sequential calculus of classes. In Proceedings of *FMCO*, 2004.

[3] S. Abramsky, D. R. Ghica, A. S. Murawski, C.-H. L. Ong, and I. D. B. Stark. Nominal games and full abstraction for the nu-calculus. In Proceedings of *LICS*, 2004.

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

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

[6] S. Abramsky and G. McCusker. Game semantics. In *Logic and Computation: Marktoberdorf Proceedings*. Springer-Verlag, 1998.

[7] J. Alves-Foss, editor. *Formal Syntax and Semantics of Java*, volume 1523 of *Lecture Notes in Computer Science*. Springer, 1999.

[8] J. Alves-Foss and F. S. Lam. Dynamic denotational semantics of Java. In *Formal Syntax and Semantics of Java*, pages 201–240. 1999.

[9] G.M. Bierman, M.J. Parkinson, and A.M. Pitts. MJ: An imperative core calculus for Java and Java with effects. Technical Report 563, Computer Laboratory, University of Cambridge, 2002.

[10] H. Björklund and T. Schwentick. On notions of regularity for data languages. *Theor. Comput. Sci.*, 411(4-5):702–715, 2010.

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

[12] J. M. E. Hyland and C.-H. L. Ong. On Full Abstraction for PCF. *Information and Computation*, 163(2):285–408, 2000.

[13] A. Jeffrey and J. Rathke. Java Jr: Fully abstract trace semantics for a core Java language. In Proceedings of *ESOP*, 2003.

[14] A. Jeffrey and J. Rathke. A fully abstract may testing semantics for concurrent objects. *Theor. Comput. Sci.*, 338(1-3):17–63, 2005.

[15] S. N. Kamin and U. S. Reddy. Two semantic models of object-oriented languages. In *Theoretical Aspects of Object Oriented Programming*, pages 463–495. MIT Press, 1994.

[16] V. Koutavas and M. Wand. Reasoning about class behavior. In Proceedings of *FOOL/WOOD*, 2007.

[17] J. Laird. A game semantics of local names and good variables. In *Proceedings of FOSSACS*, 2004.

[18] J. Laird. Game semantics for higher-order concurrency. In Proceedings of *FSTTCS*, 2006.

[19] J. Laird. Game semantics for call-by-value polymorphism. In *Proceedings of ICALP*, 2010.

[20] S. B. Lassen and P. B. Levy. Typed normal form bisimulation for parametric polymorphism. In *Proceedings of LICS*, 2008.

[21] R. Milner. Fully abstract models of typed lambda-calculi. *Theoretical Computer Science*, 4(1):1–22, 1977.

[22] E. Moggi. Computational lambda-calculus and monads. In *Proceedings of LICS*, 1989.

[23] A. S. Murawski and N. Tzevelekos. Algorithmic nominal game semantics. In *Proceedings of ESOP*, 2011.

[24] A. S. Murawski and N. Tzevelekos. Game semantics for good general references. In *Proceedings of LICS*, 2011.

[25] J. Power and E. Robinson. Premonoidal categories and notions of computation. *Math. Struct. in Comput. Sci.*, 7:453–468, 1997.

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

[27] N. Tzevelekos. Fresh-register automata. In *Proc. of POPL*, 2011.