Typed Interpretations of Extensible Objects

VIVIANA BONO
Università di Torino
MICHELE BUGLIESI
Università "Ca' Foscari" di Venezia
SILVIA CRAFA
Università "Ca' Foscari" di Venezia

Finding typed encodings of object-oriented into procedural or functional programming sheds light on the theoretical foundations of object-oriented languages and their specific typing constructs and techniques. This paper describes a type preserving and computationally adequate interpretation of a full-fledged object calculus that supports message passing and primitives for object update and extension. The target theory is a higher-order λ-calculus with polymorphic types, recursive types and subtyping. The interpretation specializes to calculi of nonextensible objects, and validates the expected subtyping relationships.

Categories and Subject Descriptors: D.3.1 [**Programming Languages**]: Formal Definitions and Theory—*Semantics*; F.3.1 [**Logics and Meaning of Programs**]: Specifying and Verifying and Reasoning about Programs—*Logics of Programs*; F.3.3 [**Logics and Meaning of Programs**]: Studies of Program Constructs—*Object Oriented Constructs*; *Type Structure*

Additional Key Words and Phrases: object calculus, extensible object, type system, type specialization, subtyping, typed encoding, computational adequacy

1. INTRODUCTION

Object-oriented languages have introduced a number of ideas, concepts and techniques: the concepts of *self*, class and subclassing, as well as the techniques for code reuse based on inheritance and subsumption have all emerged with the advent of this paradigm. Although some of these ideas and techniques are yet to be fully formalized, all of them have proved to be very useful and effective in programming.

Interpretations of object-oriented programming are typically defined in terms of reductions to procedural or functional programming, and help provide sound and formal foundations to object-oriented languages and their specific constructs and

Authors' address
Viviana Bono: Università di Torino, Dipartimento di Informatica, C.so Svizzera 185, I-10149 Torino, Italy. e-mail: bono@di.unito.it, http://www.di.unito.it/~bono
Michele Bugliesi and Silvia Crafa: Università Ca' Foscari di Venezia, Dipartimento di Informatica, Via Torino 155, 30172, Mestre (VE), Italy. E-mail:{michele,silvia}@dsi.unive.it, http://www.dsi.unive.it/~{michele,silvia}

techniques. The reduction is not straightforward: difficulties arise principally at the level of types, when trying to validate the subtyping properties of the source languages. The first contributions to this research date back to Cook's important early work [Cook 1987; 1989] on the *generator model*, and to Kamin's interpretation of objects as records of functions (the *self-application semantics* of [Kamin 1988]). Refined formulations of the generator model were later proposed by Bruce in [Bruce 1994] to give direct interpretations for *class-based* object calculi.

A number of object encodings for the so-called *object-based* calculi have subsequently been proposed by Pierce and Turner [Pierce and Turner 1994], Abadi, Cardelli and Viswanathan [Abadi and Cardelli 1996b; Abadi et al. 1996], Bruce, Pierce and Cardelli [Bruce et al. 1999], and recently by Crary [Crary 1999]. These interpretations apply to a rich variety of object calculi with primitives of object formation, message send and (functional) method override: they succeed in validating the operational semantics of these calculi as well as the expected subtyping relations over object types; finally they extend nicely to the case of *Self Types* and other object-oriented constructs.

None of these proposals, however, appear to scale to calculi of extensible objects, where primitives are provided for modifying the size of an object with the addition of new methods. There are two major difficulties in dealing with interpretations of extensible objects. The first is the presence of the mechanism known as *MyType* method specialization [Bruce 1994; Fisher et al. 1994], a typing technique that allows the types of methods to be specialized when they are inherited from an object (or *prototype*) to its *derivates*. The second difficulty arises from the coexistence of subtyping and object extension, two mechanisms that are essentially incompatible [Fisher and Mitchell 1995], and hence difficult to combine in sound and flexible type systems.

In this paper we present an interpretation of extensible objects that addresses both these problems. Our source calculus supports extensible objects in ways similar to the *Lambda Calculus of Objects* [Fisher et al. 1994] and subsequent calculi [Fisher and Mitchell 1995; Bono and Liquori 1995; Bono and Bugliesi 1999b]. *MyType* polymorphism is rendered in the type system by means of *match*-bounded polymorphism, as in the system we developed in [Bono and Bugliesi 1999b]. Subtyping, is accounted for by distinguishing between extensible and nonextensible objects, as proposed by Fisher and Mitchell in [Fisher and Mitchell 1995].

As in most of the existing papers on the subject, we define our interpretation as an encoding: the target calculus is a polymorphic λ-calculus with records, recursive types and (higher-order) subtyping. Within this calculus, an extensible object is interpreted as a pair of two components: the object *generator*, which is made available to contexts where the structure of the object is extended with new methods or fields, and the *interface*, a recursive record that provides direct access to the methods and fields of the object, for its clients. Technically, the two components are collectively grouped into a single recursive record by a technique which is inspired by, and generalizes, the *split-method* interpretation of [Abadi et al. 1996].

The resulting interpretation is faithful to the source calculus in that (*i*) it preserves the validity of typing judgements, and (*ii*) it validates the operational semantics, that is shown by proving that the encoding is computationally adequate.

Besides providing a fully formal interpretation of extensible objects, the interpretation also clarifies the relationship between calculi of extensible and nonextensible objects presented in the recent literature. In fact, the encoding specializes smoothly to the case of nonextensible objects and object types, validating the expected subtyping relationships. Although we focus on one particular calculus – specifically, on one approach to combining object extension with subtyping – the translation is sufficiently general to capture other notions of subtyping over object types (a notable example are the rules for covariant subtyping of [Abadi and Cardelli 1996b]).

An interpretation of extensible objects with essentially the same properties as ours has recently, and independently, been obtained by Boudol and Dal Zilio [Boudol and Dal Zilio 1999][1]. We postpone the comparison between the two solutions to Section 8, and organize the rest of the paper as follows. In Section 2 we introduce the calculus of extensible objects and describe its type system. In Section 3, we briefly review the target theory for the encoding. In Section 4 we describe the encoding, and then, in Sections 5 and 6 we prove it sound and computationally adequate. In Section 7 we discuss the interpretation of nonextensible objects and various forms of subtyping relationships. We conclude in Section 8 with comparisons with related work. The typing rules of the source and target calculi are collected in Appendices B, C and D.

2. EXTENSIBLE OBJECTS AND OBJECT TYPES

The source calculus of our translation, named $\mathtt{Ob}^+$, is a typed variant of the *Lambda Calculus of Objects* of [Fisher et al. 1994], the first calculus that provided a formal type system for extensible objects. There are a few differences from the original proposal of [Fisher et al. 1994]: firstly, we rely on an explicitly typed syntax, secondly we use ς-binders for method bodies instead of the λ-binders of [Fisher et al. 1994], and finally we distinguish methods from fields, both syntactically and semantically. The typed syntax ensures that well-typed objects have unique types, a property that was missing in [Fisher et al. 1994]. The use of ς-binders eases the comparisons between our translation and related translations in the literature (with ς-binders, the syntax of $\mathtt{Ob}^+$ is a proper extension of Abadi and Cardelli's typed ς-calculus [Abadi and Cardelli 1996b]). As for the distinction of methods and fields, besides being a common practice in object-oriented languages and calculi, it arises as a result of a retrospective analysis of the interpretation of objects and object types. As we shall see, the qualitative nature of the target theory used in the interpretation changes significantly depending on the kind of updates supported by the source calculus: specifically, recursive types suffice for the interpretation of both *external* and *self-inflicted* field updates and *external* method overrides, while *self-inflicted* method overrides require a non-trivial extension of the target theory, one where recursion and least fixed points are available not only for types, but also for type operators.

[1]Interestingly enough, [Boudol and Dal Zilio 1999] and the preliminary version of the present paper [Bono and Bugliesi 1999a] have been presented at (the same session of) the same conference.

2.1 Terms and Operational Semantics

The syntax of terms is defined by the following productions:

$$
\begin{array}{lll}
A, B, C ::= & \textit{Types} & \\
& X, U & \textit{variable} \\
& \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i \in I}, m_j : B_j\{X\}^{j \in J}\rangle\!\rangle & \textit{object type} \\
& & (v_i\textit{'s and } m_j\textit{'s distinct})
\end{array}
$$

$$
\begin{array}{lll}
a, b, c ::= & \textit{Terms} & \\
& x & \textit{variable} \\
& \varsigma(X{=}A)\langle v_i = c_i{}^{i \in I}, m_j = \varsigma(x : X)b_j\{X, x\}^{j \in J}\rangle & \textit{object} \\
& & (v_i\textit{'s and } m_j\textit{'s distinct}) \\
& a.v & \textit{field selection} \\
& a.v \leftarrow_A b & \textit{field update} \\
& a.v \leftarrow\!\!+_A b & \textit{field addition} \\
& a \circ m & \textit{method invocation} \\
& a \circ m \leftarrow \varsigma(X{=}A)\varsigma(x : X)b\{X, x\} & \textit{method override} \\
& a \circ m \leftarrow\!\!+ \varsigma(X{=}A)\varsigma(x : X)b\{X, x\} & \textit{method addition}
\end{array}
$$

Terms of the form $\varsigma(X{=}A)\langle v_i = c_i{}^{i \in I}, m_j = \varsigma(x : X)b_j\{X, x\}^{j \in J}\rangle$ denote objects, collections of named fields (or instance variables) and methods that can be selected, updated, or added. The notation $b\{X, x\}$ emphasizes that the type variable X and the term variable x may occur free in b. Given $b\{X, x\}$, we write $b\{A, a\}$ (or, equivalently $b\{X := A, x := a\}$) for the term that results from substituting the type A and the term a for every free occurrence of X and x in b.

Each of the primitive operations on objects comes in two versions, for fields and methods. $a \circ m$ invokes the method associated with the label m in a, while $a.v$ selects the field v; $a \circ m \leftarrow \varsigma(X{=}A)\varsigma(x : X)b\{X, x\}$ replaces the current body of m in a with $\varsigma(X{=}A)\varsigma(x : X)b\{X, x\}$, while $a.v \leftarrow_A b$ performs the corresponding operation on fields; finally, $a \circ m \leftarrow\!\!+ \varsigma(X{=}A)\varsigma(x : X)b\{X, x\}$ extends a by adding a new label m with associated body $\varsigma(X{=}A)\varsigma(x : X)b\{X, x\}$, and $a.v \leftarrow\!\!+_A b$ does the same with the field v.

Terms that differ only for renaming of bound variables, or for the relative order of method and field labels are considered equal: we write $a \equiv b$ to state that a and b are equal. To ease the notation, whenever the distinction between methods and fields may soundly be disregarded, we use ℓ to denote method or field labels, and adopt the following shorthands:

$$
\begin{array}{rcl}
\varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i \in I \cup J}\rangle & \triangleq & \varsigma(X{=}A)\langle v_i = c_i{}^{i \in I}, m_j = \varsigma(x : X)b_j\{X, x\}^{j \in J}\rangle \\
a \cdot \ell & \triangleq & a \circ \ell \text{ or } a.\ell \\
a \cdot \ell \leftarrow_{(X,A)} \mathbf{b}\{X\} & \triangleq & a \circ \ell \leftarrow \varsigma(X{=}A)\varsigma(x : X)b\{X, x\} \text{ or } a.\ell \leftarrow {}_A b \\
a \cdot \ell \leftarrow\!\!+_{(X,A)} \mathbf{b}\{X\} & \triangleq & a \circ \ell \leftarrow\!\!+ \varsigma(X{=}A)\varsigma(x : X)b\{X, x\} \text{ or } a.\ell \leftarrow\!\!+ {}_A b
\end{array}
$$

The evaluation relation $\Downarrow_o$ over closed terms is defined in Figure 1 as an extension of the corresponding relation in [Abadi and Cardelli 1996b] that handles field and method addition. A *result* r is defined to be a term in object form. We say that a closed term a converges – written $a \Downarrow_o$ – if there exists a result r such that $a \Downarrow_o r$.

$$(\text{Select}_\text{v}) \quad \frac{a \Downarrow_o \varsigma(X{=}A)\langle \ldots, v_j{=}c_j, \ldots\rangle \quad c_j \Downarrow_o r}{a.v_j \Downarrow_o r}$$

$$(\text{Select}_\text{m}) \quad \frac{a \Downarrow_o \widehat{a} \quad b_j\{A, \widehat{a}\} \Downarrow_o r \quad (\widehat{a} \equiv \varsigma(X{=}A)\langle \ldots, m_j = \varsigma(x:X)b_j\{X,x\}, \ldots\rangle)}{a \circ m_j \Downarrow_o r}$$

$$(\text{Override}) \quad \frac{a \Downarrow_o \varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J}\rangle \quad k \in I \cup J}{a \cdot \ell_k \leftarrow_{(X,A)} \mathbf{b}\{X\} \Downarrow_o \varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J-\{k\}}, \ell_k =_{(X)} \mathbf{b}\{X\}\rangle}$$

$$(\text{Extend}) \quad \frac{a \Downarrow_o \varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J}\rangle \quad \ell \notin \{\ell_i\}^{i\in I\cup J}}{a \cdot \ell \longleftarrow\!\!\!+_{(X,A^+)} \mathbf{b}\{X\} \Downarrow_o \varsigma(X{=}A^+)\langle \ell =_{(X)} \mathbf{b}\{X\}, \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J}\rangle}$$

Fig. 1. Operational Semantics for $\mathtt{Ob}^+$

2.2 Type System

An object type $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$ is the type of all the objects with fields v_i ($i \in I$), and methods m_j ($j \in J$). When invoked, each field v_i returns a value of type C_i, and each method m_j a value of type $B_j\{A\}$, that is the type B_j with every free occurrence of the variable X substituted by the type A itself: this type substitution reflects the *self-substitution* semantics of method invocation defined by the (Select_m) rule introduced above. As for the syntax of terms, we use the notation $\mathtt{pro}(X)\langle\!\langle \ell_i : B_i\{X\}^{i\in I}\rangle\!\rangle$ for pro-types, whenever there is no reason to distinguish methods from fields. Object types that differ in the order of the component labels, or for the names of bound variables are considered equal: we write $A \equiv B$ to state that the types A and B are equal.

The typing rules for $\mathtt{Ob}^+$ rely on the form of match-bounded polymorphism we studied in [Bono and Bugliesi 1999b] for the *Lambda Calculus of Objects* [Fisher et al. 1994]. Polymorphic types arise in the typing of methods as a result of (i) the fact that method bodies depend on *self*, and (ii) that they may be invoked on extensions of the object where they are first installed. As a consequence, to ensure sound typings of method invocation, method bodies are typed in a context that assumes the so-called *MyType* for the *self* variable, i.e., a match-bounded type variable that represents the types of all the objects arising as a result of the possible extensions of the host object with new methods and fields. The use of *matching* [Bruce 1994] in place of the more standard relation of subtyping is central to the type system, as matching, unlike subtyping, does not support subsumption: since objects are extensible, absence of subsumption on pro-types is crucial for type soundness [Bono and Bugliesi 1999b].

MyType polymorphism is illustrated in the typing rule for object formation, given below, where we assume $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$:

$$\frac{\Gamma \vdash c_i : C_i, \quad \Gamma, U \lessdot\!\# A, x : U \vdash b_j\{U, x\} : B_j\{U\} \qquad \forall\, i \in I, i \in J}{\Gamma \vdash \varsigma(X{=}A)\langle v_i = c_i{}^{i \in I}, m_j = \varsigma(x : X) b_j\{X, x\}^{j \in J}\rangle : A}$$

MyType is the type variable U match-bounded in the context of the typing judgements of method bodies. The rule also emphasizes the distinction between methods and fields: since the latter do not depend on *self*, their type need not depend on *MyType*. As a further remark, note that the return type of each method depends polymorphically on *MyType*: this allows the (return) type of methods to be soundly specialized upon method addition. The role of *MyType* for method specialization is further clarified in the typing rule for method invocation:

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash A \lessdot\!\# \, \mathtt{pro}(X)\langle\!\langle \ell : B\{X\}\rangle\!\rangle}{\Gamma \vdash a \cdot \ell : B\{A\}}$$

An invocation for (the field or) method ℓ on an object a requires a to have a `pro`-type containing the label ℓ. The result of the call has the type B listed in the `pro`-type of a, with A substituted for X (if ℓ is a field, this substitution is vacuous). Note that A may either be a `pro`-type matching $\mathtt{pro}(X)\langle\!\langle \ell : B\{X\}\rangle\!\rangle$, or else an unknown type (i.e., a type variable) occurring (match-bounded) in the context Γ. Rules like the one above are sometimes referred to as *structural rules* [Abadi and Cardelli 1996b], and their use is critical for an adequate rendering of *MyType* polymorphism: it is the ability to refer to possibly unknown types that allows methods to act parametrically over any $U \lessdot\!\# A$, where U is the type of *self*, and A is a given `pro`-type.

The typing rules for $\mathtt{Ob}^{+}$ are collected in Appendix B: comments on the most interesting rules are given below.

(Val Field Update) is a structural rule: as in (Val Select), the type A of the object a being updated may either be a type variable, or a `pro`-type. When it is a `pro`-type, the update is *external*, when it is a type variable, the update is *self-inflicted*. The judgement $\Gamma \vdash A \lessdot\!\# \, \mathtt{pro}(X)\langle\!\langle \ell : C\rangle\!\rangle$ requires A (hence $a : A$) to have a field ℓ with type C, and the remaining judgement ensures that the update preserves the type of the object.

(Val Method Override) handles the case of updates for methods: unlike the corresponding rule for fields, (Val Method Override) is non-structural, as the type A of the object being updated is required to be a `pro`-type. As a consequence, method override may *not* be self-inflicted: instead, it is available as an *external* operation which can only be performed from outside the object. This restriction could safely be lifted, without consequences on the operational behavior of the source calculus, or on the operational soundness of the type system[2]. On the other hand, as we mentioned, self-inflicted method overrides do have significant impact on the semantic interpretation.

(Val Extend) is the typing rule for field and method additions. The label ℓ is assumed to be different from all of the ℓ's, $i \in I$, and the type of the object a being extended is required to be a `pro`-type: since no subtyping is available on `pro`-types, this implies that an object extension is typed with *exact* knowledge of the type of

[2]Type soundness in the presence of self-inflicted method overrides can be proved as we did in the calculus described in [Bono and Bugliesi 1999b].

a.

We conclude the discussion on the source calculus stating the main properties of the type system. Proofs for the results below are omitted, as they are essentially the same as those given in full detail in [Bono and Bugliesi 1999b].

THEOREM 2.1 SUBJECT REDUCTION. *If* $\Gamma \vdash a : A$ *in* $\mathtt{Ob}^{+}$, *and* $a \Downarrow_o r$, *then also* $\Gamma \vdash r : A$.

THEOREM 2.2 SOUNDNESS. *Let* c *be a closed expression such that* $\varnothing \vdash c : A$ *for some type* A. *Then:*

(1) *if* $c \equiv a{\cdot}\ell \leftarrow_{(X,A)} \mathbf{b}\{X\}$, *and* $a \Downarrow_o r$, *then* $r \equiv \varsigma(X{=}A)\langle \ldots, \ell =_{(X)} \overline{\mathbf{b}}\{X\}, \ldots\rangle$.
(2) *if* $c \equiv a{\cdot}\ell \longleftarrow\!\!\!+_{(X,A)} \mathbf{b}\{X\}$, *and* $a \Downarrow_o r$, *then* $r \equiv \varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i \in I \cup J}\rangle$ *and* $\ell \notin \{\ell_i\}^{i \in I}$.
(3) *if* $c \equiv a \cdot \ell$ *and* $a \Downarrow_o r$, *then* $r \equiv \varsigma(X{=}A)\langle \ldots, \ell =_{(X)} \mathbf{b}\{X\}, \ldots\rangle$.

3. THE TARGET CALCULUS $F_{\omega<:\mu}$

The target calculus of the translation is equivalent to the polymorphic typed λ-calculus $F^{\omega}_{<:}$ with recursive and record types, and higher-order subtyping. We briefly review the syntax, introducing notation and terminology on type operators and recursive types.

$\mathtt{K} ::=$	*Kinds*	
	$\mathtt{T}$	type
	$\mathtt{K} \Rightarrow \mathtt{K}$	type operator
$\mathbb{A}, \mathbb{B} ::=$	*Constructors*	
	X	constructor variable
	$\mathtt{Top}$	greatest constructor of kind $\mathtt{T}$
	$\mathbb{A} \to \mathbb{B}$	function type
	$[m_1 : \mathbb{B}, \ldots, m_k : \mathbb{B}]$	record type
	$\forall(X <: \mathbb{A} :: \mathtt{K})\mathbb{A}$	bounded universal type
	$\mu(X)\mathbb{A}$	recursive type
	$\lambda(X :: \mathtt{K})\mathbb{B}$	operator
	$\mathbb{B}(\mathbb{A})$	operator application
$M, N ::=$	*Expressions*	
	x	variable
	$\lambda(x : \mathbb{A})M$	abstraction
	$M\,N$	application
	$\Lambda(X <: \mathbb{A} :: \mathtt{K})e$	type-abstraction
	$M\,\mathbb{A}$	type-application
	$[m_1 = M_1, \ldots, m_k = M_k]$	record
	$M.m$	record selection
	$\mathtt{fold}(\mathbb{A}, M)$	recursive fold
	$\mathtt{unfold}(M)$	recursive unfold
	$\mathtt{let}\ x = M\ \mathtt{in}\ N$	local definition
	$\mathtt{letrec}\ f(x : \mathbb{A}) : \mathbb{B} = M\ \mathtt{in}\ N$	recursive local definition

Types and type operators are collectively called *constructors*: a type operator is a function from types to types. The notation $\mathbb{A} :: \mathtt{K}$ indicates that the constructor $\mathbb{A}$ has kind $\mathtt{K}$. The typing rules, found in Appendix D, are standard (see Chapter 20 of [Abadi and Cardelli 1996b]). Type equality is defined by judgements of the form $\Gamma \vdash \mathbb{A} \leftrightarrow \mathbb{B}$, modulo renaming of bound variables. The following notation is used throughout: $Op \equiv \mathtt{T} \Rightarrow \mathtt{T}$ is the kind of type operators, $\mathbb{A} \leq \mathbb{B}$ denotes subtyping over type operators, whereas $\mathbb{A} <: \mathbb{B}$ denotes subtyping over the kind $\mathtt{T}$ of types. We also use the following shorthands to emphasize the relationships between type operators and their fixed points. Given the type operator $\mathbb{A} :: Op$, $\mathbb{A}^* \equiv \mu(X)\mathbb{A}(X)$ is the (least) fixed point of $\mathbb{A}$; dually, given the recursive type $\mathbb{A} :: \mathtt{T} \equiv \mu(X)\mathbb{B}(X)$, $\mathbb{A}^{\text{OP}} \equiv \lambda(X)\mathbb{B}(X) :: Op$ is the type operator whose (least) fixed point is $\mathbb{A}$. As in [Abadi and Cardelli 1996a], $\mathbb{A}^{\text{OP}}$ is defined in terms of the syntactic form $\mu(X)\mathbb{B}(X)$ of $\mathbb{A}$: the notation $\mathbb{A}^{\text{OP}}$ is well-defined because we rely on a weak notion of type equality whereby a recursive type is isomorphic, rather than equal, to its unfoldings.

Results, or values, in this calculus are lambda abstractions, records, and recursive folds, as defined by the following productions:

$$r \ ::= \ \lambda(x : \mathbb{A})\, M \mid [m_1 = M_1, \ldots, m_k = M_k] \mid \mathtt{fold}(\mathbb{A}, M) \mid \Lambda(X <: \mathbb{A} :: \mathtt{K})\, M$$

A standard call-by-name relation of evaluation for the calculus is given in Figure 2. We say that a closed term M converges – written $M \Downarrow_f$ – if there exists a result r such that $M \Downarrow_f r$.

$$(\beta_1) \quad \frac{M \Downarrow_f (\lambda(x : \mathbb{A})M') \quad M'\{x := N\} \Downarrow_f r}{M\,N \Downarrow_f r}$$

$$(\beta_2) \quad \frac{M \Downarrow_f (\Lambda(X <: \mathbb{A} :: \mathtt{K})\, N) \quad N\{X := \mathbb{B}\} \Downarrow_f r}{M\,\mathbb{B} \Downarrow_f r}$$

$$(\text{select}) \quad \frac{M \Downarrow_f [\ldots, m_j = M_j, \ldots] \quad M_j \Downarrow_f r}{M.m_j \Downarrow_f r}$$

$$(\text{unfold}) \quad \frac{M \Downarrow_f \mathtt{fold}(\mathbb{A}, N) \quad N \Downarrow_f r}{\mathtt{unfold}(M) \Downarrow_f r}$$

$$(\text{letrec}) \quad \frac{N\{f := M\{f := \mathtt{letrec}\ f = M\ \mathtt{in}\ f\}\} \Downarrow_f r}{\mathtt{letrec}\ f = M\ \mathtt{in}\ N \Downarrow_f r}$$

Fig. 2. Operational Semantics for $F_{\omega<:\mu}$

4. ENCODING OF EXTENSIBLE OBJECTS AND TYPES

To understand the encoding, and its subtleties, it is useful to proceed by steps, and first discuss solutions that are intuitively simple but do not quite work. We initially keep the discussion informal, and look at a simplified case where objects have no fields, and for which the only available operators are method addition and invocation. Then we extend our analysis to objects with fields, and show how to account for field selection, addition and update. Finally we look at method overrides, distinguishing *external* from *self-inflicted* overrides: we show that the former can be encoded in $F_{\omega<:\mu}$ with no additional machinery, and discuss the extensions to $F_{\omega<:\mu}$ required to handle the latter. In Section 4.5, we summarize the result of the analysis giving the formal definition of the interpretation.

4.1 Failures of Self Application

We first consider objects with no fields. Looking at the reduction rules, it would be tempting to interpret these objects as in the *self-application* semantics of [Kamin 1988]. In this semantics, methods are functions of the self parameter, objects are records of such functions, and method invocation is field selection plus self-application. This semantics was originally proposed as an interpretation of nonextensible objects, and its properties are well-known [Abadi and Cardelli 1996b]: it works well in the untyped case, but fails in the typed case because it does not validate the expected subtyping relationships over object types.

A similar situation arises for our extensible objects, even though no subtyping is available over pro-types. Following the self-application semantics, one would interpret the type $\mathtt{pro}(X)\langle\!\langle m{:}B\rangle\!\rangle$ as the recursive record type $A \equiv \mu(X)[m : X \to B]$ which solves the type equation $A = [m : A{\to}B]$. Now, given (the interpretation of) an object $a : A$, consider extending a with (the interpretation of) a new method $m' = \lambda(s)b$. The extension is interpreted as the formation of the new record $[m = a.m, m' = \lambda(s)b]$, whose type is $A^+ \equiv \mu(X)[m : X{\to}B, m' : X \to B']$. However, typing the new record requires the type of $a.m$ to be subsumed to $A^+ \to B$: the problem is that the subtyping relationship $A{\to}B <: A^+{\to}B$ fails due to the contravariant occurrence of the types A and A^+.

To circumvent the use of subsumption, a seemingly correct solution would be to refine the self-application semantics with polymorphically typed methods where pro-types are recursive types of the form $\mu(X)[m : \forall(U \not\lessdot X)U{\to}B\{U\}]$. Unfortunately this attempt fails when trying to reduce matching to subtyping in $F_{\omega<:\mu}$: the reasons are essentially the same as before, given that the universal quantifier is again contravariant in its bound.

In both the previous attempts, the actual source of the problem is the poor interaction between the subtyping rules for recursive types and the contravariant occurrence of the recursion variable in the types of methods: to find a solution, we need to break this problematic dependency.

4.2 Methods and Split Labels

If we look at the typing rules of $\mathtt{Ob}^+$, we may identify two distinguished views of methods. Consider the rule (Val Object): the premises give the internal view, in which methods are concrete values – abstractions of *self* – while the conclusion provides the external view where methods are seen as "abstract services" that can be invoked by messages. This observation suggests an interpretation that splits methods into two parts, in ways similar to, but different from (see Section 8), the translation of [Abadi et al. 1996], thus making the two views on methods explicit. In the untyped case, the object $\langle m_i = \varsigma(x)b_i{}^{i\in 1..n}\rangle$ can be interpreted as the recursive record that satisfies the following equation:

$$M = [m_i^{gen} = \lambda(x)\,[\![\, b_i \,]\!]^{\,i\in 1..n},\; m_i^{sel} = M.m_i^{gen}(M)^{\;i\in 1..n}]$$

Each method m_i is represented by two components: the *generator* m_i^{gen}, associated with the function that represents the actual body of m_i, and the *selector* m_i^{sel} which is the result of self-applying m_i^{gen} to the host object, and can directly be invoked by selection, without self-application. Thus, *clients* of the object can access the object's methods by means of the selectors, while *derived objects*, obtained by the addition of new methods, inherit the generators and re-install the corresponding selectors, thus rebinding *self* to the extended structure of the host object. Viewed this way, the set of selectors can be thought of as the *abstract interface* that the object provides for its clients, while the generators are available in contexts where the structure of the object needs to be extended with new methods.

This idea works well in the typed case as well. Consider the pro-type $A \equiv \mathtt{pro}(X)\langle\!\langle m_i : B_i\{X\}^{i\in 1..n}\rangle\!\rangle$. The interface associated with A is represented by the type operator $\mathbb{A}^{\text{IN}}(X) \equiv [m_i^{sel} : \mathbb{B}_i\{X\}^{i\in 1..n}]$ that collects the method selectors (here, and below, $\mathbb{B}_i$ is the translation of B_i). The type A, in turn, is interpreted as the (recursive) record type that collects selectors and generators for each of the m_i, defined as follows: $\mathbb{A} \equiv \mu(X)[m_i^{gen} : \forall(U \le \mathbb{A}^{\text{IN}})U^* \to \mathbb{B}_i\{U^*\}^{\;i\in 1..n}, m_i^{sel} : \mathbb{B}_i\{X\}^{\;i\in 1..n}]$. The generators have polymorphic function types corresponding to the match-bounded types used in the typing rules of the source calculus: following [Abadi and Cardelli 1996a], matching is interpreted as higher-order subtyping. The typed translation of terms derives immediately from the untyped translation and the translation of types. The object $\varsigma(X{=}A)\langle m_i = \varsigma(x : X)b_i\{X\}^{i\in 1..n}\rangle$, is interpreted as the following recursive record, where $\mathbb{A}^{\text{OP}}$ is the type operator corresponding to $\mathbb{A}$:

$$M = [m_i^{gen} = \Lambda(U \le \mathbb{A}^{\text{IN}})\lambda(x : U^*)\,[\![\, b_i\{U\} \,]\!]^{\,i\in 1..n},\; m_i^{sel} = M.m_i^{gen}(\mathbb{A}^{\text{OP}})(M)^{\;i\in 1..n}]$$

The (higher-order) subtype constraint $U \le \mathbb{A}^{\text{IN}}$ in the generators ensures that each method may legally invoke its sibling methods via *self*. The use of the interface $\mathbb{A}^{\text{IN}}$ in the bounded quantifier is critical for well-typedness: besides exposing the selectors, for use within each method body, it validates the subtyping relationships

$$\forall(U \le \mathbb{A}^{\text{IN}})U^* \to \mathbb{B}_i\{U^*\} \le \forall(U \le (\mathbb{A}^+)^{\text{IN}})U^* \to \mathbb{B}_i\{U^*\}$$

needed to inherit the generators upon object extension, as well as the relationship $\mathbb{A}^{\text{OP}} \le \mathbb{A}^{\text{IN}}$ needed to type the self-application $M.m_i^{gen}(\mathbb{A}^{\text{OP}})$. Finally, the interface (hence the internal view of *self*), hides the generators to reflect the fact that in the source calculus objects cannot be self-extended.

4.3 Fields and Field Update

Fields are handled quite easily in the interpretation: they need no generators, as they do not depend on *self*, and their evaluation is independent of the structure of the object and of its extensions. On the other hand, field updates require a different treatment (and interpretation) of the recursive nature of *self* and of self reference. The problem is well-known [Abadi and Cardelli 1996b]: defining objects by *direct* recursion, as we did above, does not quite reflect their computational behavior. Specifically, field updates do not work if the recursion freezes *self* to be the object at the time of creation or extension: subsequent updates on a field are not reflected in the invocation of a method that depended on that field through *self.* The solution, as in [Abadi et al. 1996; Abadi and Cardelli 1996b], is to give a recursive definition not of the object itself, but rather of the dependency of the object on its methods. In the untyped case, this correspond to the following interpretation of $\langle m_i = \varsigma(x)b_i{}^{i\in 1..n}\rangle$.

$$\begin{array}{l} \texttt{letrec}\; mkobj(f_1,\ldots,f_n) \;=\; \\ \quad [m_i^{gen} = f_i{}^{i\in 1..n},\; m_i^{sel} = f_i(mkobj(f_1,\ldots,f_n))\; {}^{i\in 1..n}] \\ \texttt{in}\; mkobj(\lambda(x)\,[\![\, b_1 \,]\!]\,,\ldots,\lambda(x)\,[\![\, b_n \,]\!]\,) \end{array}$$

Now it is the definition of $mkobj$, i.e., of the function that creates the object, that is recursive, not the object itself: this leaves room for a correct interpretation of field updates using the *updaters* of [Abadi et al. 1996]. The interpretation of a, now complete, object of the form $\langle v_i = c_i{}^{in\in 1..n}, m_j = \varsigma(x)b_j{}^{j\in 1..m}\rangle$ can be defined as follows:

$$\begin{array}{l} \texttt{letrec}\; mkobj(w_i{}^{i\in 1..n}, f_j{}^{j\in 1..m}) = \\ \quad [v_i^{sel} = w_i{}^{i\in 1..n}, \\ \quad v_i^{upd} = \lambda(z)mkobj(w_1,\ldots,w_{i-1},z,w_{i+1},\ldots,w_n,f_j{}^{j\in 1..m})^{i\in 1..n}, \\ \quad m_j^{gen} = f_j{}^{j\in 1..m}, \\ \quad m_j^{sel} = f_j(mkobj(w_i{}^{i\in 1..n}, f_j{}^{j\in 1..m}))\; {}^{j\in 1..m}] \\ \texttt{in}\; mkobj(\,[\![\, c_i \,]\!]^{i\in 1..n}, \lambda(x)\,[\![\, b_j \,]\!]^{j\in 1..m}) \end{array}$$

Fields are also split into two components: the selector v^{sel} provides access to the contents of the field, the updater v^{upd} takes the new value supplied in the update and returns a new object with the value installed in place of the original. A field update may then be translated by a simple call to the updater associated with that field. Finally, field or method additions, are translated by corresponding calls to the constructor function $mkobj$ (see Section 4.5, Figure 4).

The translation extends smoothly to the typed case: to allow field selection and update, the type constructor and the recursive type that represent, respectively, the interface and the type of the object, are extended with new components corresponding to the selectors and updaters associated with the object's fields. If $A \equiv \texttt{pro}(X)\langle\!\langle v : C, \ldots\rangle\!\rangle$, the type of the selector v^{sel} is $\mathbb{C}$, and the type of the updater v^{upd} is $\mathbb{C} \rightarrow \mathbb{A}$, that is the type of a function that, given an argument with the same type as the value to be updated, returns an object which has the same type as the object prior to the update. As we show in section 4.5, introducing field updaters preserves all the subtyping relationships needed in the typing of method and field addition.

4.4 Method Override

In the untyped case, method overrides can be dealt with in exactly the same way as field updates, by introducing an updater m_i^{upd} for each method, and interpreting a method override as a call to the corresponding updater. Unfortunately, the typing of method updaters poses a nontrivial problem.

Self-inflicted overrides. Consider an object $a : A$, where $A \equiv \texttt{pro}(X)\langle\!\langle m :$

$B\{X\}\rangle\!\rangle$, and consider overriding the method m of a. Using method updaters, the translation of A would be the recursive type:

$$\begin{aligned}\mathbb{A} = \mu(X)[\ & m^{gen} : \forall(U \le \mathbb{A}^{\text{IN}})U^* \to \mathbb{B}\{U^*\},\\ & m^{upd} : (\forall(U \le \mathbb{A}^{\text{IN}})U^* \to \mathbb{B}\{U^*\}) \to X,\\ & m^{sel} : \mathbb{B}\{X\}]\end{aligned}$$

As in the case of fields, the updater m^{upd} expects an argument of the same type as the actual method body –the type of m^{gen}– and returns a modified copy of the object, preserving the original type. The problem arises from self-inflicted method overrides: if a self-inflicted override is to be translated as a call to the updater, the updaters must be exposed in the interface used in the type of the generators. But then, since the generators and the updaters must be typed consistently, the updaters must be exposed in the interface $\mathbb{A}^{\text{IN}}$. This leads to a new definition of the interface associated to the type A, as the type operator that satisfies the following equation:

$$\mathbb{A}^{\text{IN}}(X) = [m_i^{upd} : (\forall(U \le \mathbb{A}^{\text{IN}})U^* \to \mathbb{B}_i\{U^*\}) \to X,\ m_i^{sel} : \mathbb{B}_i\{X\}]$$

The problem with this equation is that it involves type operators rather than types: solving equations of this kind requires a significant extension to the target theory of $F_{\omega<:\mu}$, one that allows fixed points to be taken not only at types, but also at type operators. To the best of our knowledge, this extended theory has not been studied in the literature: more importantly, its soundness is still an open problem. Given this, in the formal treatment that follows, we disregard method updaters, and focus attention to the simplified case in which method override is an operation that may only be performed from outside the object. External overrides, which are legal in the source calculus, can still be accounted for in $F_{\omega<:\mu}$, as we discuss here below.

We remark again that self-inflicted field updates stay within the simpler ambient theory. This is the main reason to distinguish among fields and methods in our setting.

External overrides. The translation of external method overrides relies on essentially the same technique used for method addition. Given the object $a \equiv \langle m_i = \varsigma(x)b_i{}^{i\in 1..n}\rangle$, the interpretation of the override operation $a \circ m_j \leftarrow \varsigma(x)b$ is the call $mkobj(\llbracket a \rrbracket.m_1^{gen}, \ldots, \llbracket a \rrbracket.m_{j-1}^{gen}, \lambda(x)\llbracket b \rrbracket, \llbracket a \rrbracket.m_{j+1}^{gen}, \ldots, \llbracket a \rrbracket.m_n^{gen})$ which forms a new object containing new body for m_j and the methods inherited from a.

Note that this interpretation of method overrides does not work for the self-inflicted case. As we already pointed out, the generators m_i^{gen} cannot be invoked from within a method body, as they are not exposed by the interface of the object: on the other hand, exposing the generators in the interface (i.e. using $\mathbb{A}^{\text{OP}}$ in place of $\mathbb{A}^{\text{IN}}$) would break the subtyping required to type an object extension (for $(\mathbb{A}^+)^{\text{OP}} \le \mathbb{A}^{\text{OP}}$ fails due to the contravariant occurrence of the universal quantifier in the type of the generators).

4.5 The Translation

The translation of types proceeds inductively on the structure of types. As in [Abadi and Cardelli 1996a], the treatment of type variables depends on the context where they are used: in contexts where they are match-bound, they are interpreted as type operators, in other contexts they are interpreted as types.

$$A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$$

INTERFACES

$$\begin{aligned}
[\![\,\Gamma', X \lessdot\!\# A, \Gamma'' \rhd X\,]\!]^{\text{IN}} &\triangleq X\\
[\![\,\Gamma \rhd A\,]\!]^{\text{IN}} &\triangleq \lambda(X)[\, v_i^{sel} : [\![\,\Gamma \rhd C_i\,]\!]^{\text{TY}}{}^{\,i\in I},\ v_i^{upd} : [\![\,\Gamma \rhd C_i\,]\!]^{\text{TY}} \to X^{\ i\in I},\\
&\qquad m_j^{sel} : [\![\,\Gamma, X \rhd B_j\{X\}\,]\!]^{\text{TY}}{}^{\,j\in J}\ ]
\end{aligned}$$

OPERATORS

$$\begin{aligned}
[\![\,\Gamma', X \lessdot\!\# A, \Gamma'' \rhd X\,]\!]^{\text{OP}} &\triangleq X\\
[\![\,\Gamma \rhd A\,]\!]^{\text{OP}} &\triangleq \lambda(X)[\, v_i^{sel} : [\![\,\Gamma \rhd C_i\,]\!]^{\text{TY}}{}^{\,i\in I},\ v_i^{upd} : [\![\,\Gamma \rhd C_i\,]\!]^{\text{TY}} \to X^{\ i\in I},\\
&\qquad m_j^{sel} : [\![\,\Gamma, X \rhd B_j\{X\}\,]\!]^{\text{TY}}{}^{\,j\in J},\\
&\qquad m_j^{gen} : \forall(U \le [\![\,\Gamma \rhd A\,]\!]^{\text{IN}})U^* \to [\![\,\Gamma, X \rhd B_j\{X\}\,]\!]^{\text{TY}}\{X{:=}U^*\}^{j\in J},\\
&\qquad ext : \forall(U \le [\![\,\Gamma \rhd A\,]\!]^{\text{IN}})U^* \to U^*]
\end{aligned}$$

TYPES

$$\begin{aligned}
[\![\,\Gamma', X \lessdot\!\# A, \Gamma'' \rhd X\,]\!]^{\text{TY}} &\triangleq X^*\\
[\![\,\Gamma', X, \Gamma'' \rhd X\,]\!]^{\text{TY}} &\triangleq X\\
[\![\,\Gamma \rhd A\,]\!]^{\text{TY}} &\triangleq \mu(X)[\, v_i^{sel} : [\![\,\Gamma \rhd C_i\,]\!]^{\text{TY}}{}^{\,i\in I},\ v_i^{upd} : [\![\,\Gamma \rhd C_i\,]\!]^{\text{TY}} \to X^{\ i\in I},\\
&\qquad m_j^{sel} : [\![\,\Gamma, X \rhd B_j\{X\}\,]\!]^{\text{TY}}{}^{\,j\in J},\\
&\qquad m_j^{gen} : \forall(U \le [\![\,\Gamma \rhd A\,]\!]^{\text{IN}})U^* \to [\![\,\Gamma, X \rhd B_j\{X\}\,]\!]^{\text{TY}}\{X{:=}U^*\}^{j\in J},\\
&\qquad ext : \forall(U \le [\![\,\Gamma \rhd A\,]\!]^{\text{IN}})U^* \to U^*\ ]
\end{aligned}$$

Fig. 3. Translation of Types

The whole translation depends on contexts. For this reason we introduce the notation $\rhd$, whose meaning is made formally clear in Figure 5.

The translation of terms, in Figure 4, formalizes the ideas we illustrated in the previous sections. To ease the notation, we use the following shorthand, where

$A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i \in 1..n}, m_j : B_j\{X\}^{j \in 1..m}\rangle\!\rangle$:

$$
\begin{array}{l}
\textsc{mkobj}_A(M_1, \ldots, M_n, N_1, \ldots, N_m) \equiv \\
\quad \mathtt{letrec}\ mkobj(w_i : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}\, i \in 1..n}, \\
\qquad\qquad\qquad f_j : \forall(U \leq [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}})U^* \to [\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}}\{X{:=}U^*\}^{j \in 1..m} \\
\qquad\qquad\qquad) : [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{TY}} = \\
\qquad \mathtt{fold}([\![\, \Gamma \triangleright A \,]\!]^{\mathrm{TY}}, \\
\qquad\quad [v_i^{sel} = w_i{}^{\ i \in 1..n} \\
\qquad\quad v_i^{upd} = \lambda(z : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}})mkobj(w_l{}^{\ l \in 1..i-1},\ z,\ w_l{}^{\ l \in i+1..n}, f_1, \ldots, f_m)^{i \in 1..n}, \\
\qquad\quad m_j^{sel} = f_j([\![\, \Gamma \triangleright A \,]\!]^{\mathrm{OP}})(mkobj(w_1, \ldots, w_n, f_1, \ldots, f_m))^{j \in 1..m}, \\
\qquad\quad m_j^{gen} = f_j{}^{j \in 1..m}, \\
\qquad\quad ext = \Lambda(U \leq [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}})\lambda(x : U^*)x] \\
\quad \mathtt{in}\ mkobj(M_1, \ldots, M_n, N_1, \ldots, N_m)
\end{array}
$$

The definition of the constructor function $\textsc{mkobj}_A$ is the typed version of the $mkobj$ function we discussed earlier: in the typed case, we define one such function for each type A. The first n parameters for $\textsc{mkobj}_A$ initialize the object's fields, the subsequent m initialize the object's methods. The ext entry, whose body is the polymorphic identity, is needed to handle the translation of object extension, as we discuss below.

The formation of an object of a given type A is translated directly as a call to the constructor function $\textsc{mkobj}_A$. Typing the selectors m_i^{sel} requires the relation $[\![\, \Gamma \triangleright A \,]\!]^{\mathrm{OP}} \leq [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}}$, which is derived by first unrolling the fixed-point and then applying the rules for constructor subtyping.

Field and method selection are translated as a call to the corresponding selectors: invoking a method or field ℓ on an object a results in a call to the ℓ^{sel} component in the translation of a. A recursive unfold is required prior to accessing the desired component.

An object extension forms a new object by applying $\textsc{mkobj}_{A^+}$ (A^+ is the type of the extended object) to the (translation of) the method bodies of the original object a, and to the newly added method. The call to $\textsc{mkobj}_{A^+}$ is passed as argument to the identity function associated to the ext field: selecting the ext field from $\widehat{a}$, – the object being extended – guarantees that $\widehat{a}$ is evaluated prior to the extension. This is required for the translation to be computationally adequate, as the evaluation rules of $\mathtt{Ob}^+$ do require a to evaluate to an object prior to evaluating an object extension.

To see why computational adequacy requires the ext field and would otherwise fail, consider the term $\Omega \equiv (\varsigma(X{=}P)\langle \ell = \varsigma(x : X)x \circ \ell\rangle) \circ \ell$. If we choose $P \equiv \mathtt{pro}(X)\langle\!\langle \ell : \mathtt{pro}(X)\langle\!\langle\rangle\!\rangle\rangle\!\rangle$, Ω is well-typed in $\mathtt{Ob}^+$, as $\varnothing \vdash \varsigma(X{=}P)\langle \ell = \varsigma(x : X)x \circ \ell\rangle : P$ is derivable, and hence so is $\varnothing \vdash \Omega : A$ for $A \equiv \mathtt{pro}(X)\langle\!\langle\rangle\!\rangle$. Now consider extending Ω with a new method, as in $\Omega^+ \equiv \Omega \circ m \longleftarrow\!\!+\ \varsigma(X{=}A^+)\varsigma(x : X)x$. The new term is also well-typed: taking $A^+ \equiv \mathtt{pro}(X)\langle\!\langle m : X\rangle\!\rangle$, from $\varnothing \vdash \Omega : A$, one derives $\varnothing \vdash \Omega^+ : A^+$.

Now, let us look at how Ω and Ω^+ are interpreted in $F_{\omega<:\mu}$. In the rest of this section we will use the following shorthands: $\mathbb{P}, \mathbb{P}^{\mathrm{IN}}, \mathbb{P}^{\mathrm{OP}}$ stand for, respectively, $[\![\, \varnothing \triangleright P \,]\!]^{\mathrm{TY}}$, $[\![\, \varnothing \triangleright P \,]\!]^{\mathrm{IN}}$, $[\![\, \varnothing \triangleright P \,]\!]^{\mathrm{OP}}$, while $(\mathbb{A}^+)^{\mathrm{IN}}, (\mathbb{A}^+)^{\mathrm{OP}}$ stand for, respec-

Variables.

$\llbracket \Gamma \triangleright x \rrbracket \triangleq x$

Object formation.

$\llbracket \Gamma \triangleright \varsigma(X{=}A)\langle v_i = c_i{}^{i\in I}, m_j = \varsigma(x:X)b_j\{X,x\}^{j\in J}\rangle \rrbracket \triangleq$
$\quad \textsc{mkobj}_A(\llbracket \Gamma \triangleright c_i \rrbracket^{i\in I}, \Lambda(U \le \llbracket \Gamma \triangleright A \rrbracket^{\textsc{in}})\lambda(x:U^*)\llbracket \Gamma, U \lessdot\!\# A, x:U \triangleright b_j\{U,x\} \rrbracket^{j\in J})$
where $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$

Field and method selection.

$\llbracket \Gamma \triangleright a\cdot\ell \rrbracket \triangleq \mathtt{unfold}(\llbracket \Gamma \triangleright a \rrbracket).\ell^{sel}$

Field update.

$\llbracket \Gamma \triangleright a.v \leftarrow_A b \rrbracket \triangleq \mathtt{unfold}(\llbracket \Gamma \triangleright a \rrbracket).v^{upd}(\llbracket \Gamma \triangleright b \rrbracket)$

Field addition.

$\llbracket \Gamma \triangleright a.v_{n+1} \longleftrightarrow_{A^+} b \rrbracket \triangleq$
$\quad \widehat{a}.ext\ (\llbracket \Gamma \triangleright A^+ \rrbracket^{\textsc{op}})\ (\textsc{mkobj}_{A^+}(\widehat{a}.v_i^{sel\ i\in 1..n}, \llbracket \Gamma \triangleright b \rrbracket, \widehat{a}.m_j{}^{gen\ j\in J}))$
where $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in 1..n},\ m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$,
$\quad A^+ \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in 1..n+1},\ m_i : B_i\{X\}^{i\in J}\rangle\!\rangle$.
$\quad \widehat{a} \equiv \mathtt{unfold}(\llbracket \Gamma \triangleright a \rrbracket)$

Method addition.

$\llbracket \Gamma \triangleright a \circ m_{n+1} \longleftrightarrow \varsigma(X{=}A^+)\varsigma(x:X)b\{X,x\} \rrbracket \triangleq$
$\quad \widehat{a}.ext\ (\llbracket \Gamma \triangleright A^+ \rrbracket^{\textsc{op}})\ (\textsc{mkobj}_{A^+}(\widehat{a}.v_i^{sel\ i\in I}, \widehat{a}.m_j^{gen\ j\in 1..n},\ \Lambda(U \le \llbracket \Gamma \triangleright A^+ \rrbracket^{\textsc{in}})\widehat{b}))$
where $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I},\ m_j : B_j\{X\}^{j\in 1..n}\rangle\!\rangle$,
$\quad A^+ \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I},\ m_i : B_i\{X\}^{i\in 1..n+1}\rangle\!\rangle$, .
$\quad \widehat{a} \equiv \mathtt{unfold}(\llbracket \Gamma \triangleright a \rrbracket)$, and $\widehat{b} \equiv \lambda(x:U^*)\llbracket \Gamma, U \lessdot\!\# A^+, x:U \triangleright b\{U,x\} \rrbracket$

Method override.

$\llbracket \Gamma \triangleright a \circ m_k \leftarrow \varsigma(X{=}A)\varsigma(x:X)b\{X,x\} \rrbracket \triangleq$
$\widehat{a}.ext\ (\llbracket \Gamma \triangleright A \rrbracket^{\textsc{op}})\ (\textsc{mkobj}_A(\widehat{a}.v_i^{sel\ i\in I}, \widehat{a}.m_1{}^{gen}, \ldots \Lambda(U \le \llbracket \Gamma \triangleright A \rrbracket^{\textsc{in}})\widehat{b}, \ldots, \widehat{a}.m_n{}^{gen}))$
where $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I},\ m_j : B_j\{X\}^{j\in 1..n}\rangle\!\rangle$, $\widehat{a} \equiv \mathtt{unfold}(\llbracket \Gamma \triangleright a \rrbracket)$.
$\quad \widehat{b} \equiv \lambda(x:U^*)\llbracket \Gamma, U \lessdot\!\# A^+, x:U \triangleright b\{U,x\} \rrbracket$

Fig. 4. Translation of Terms

tively $\llbracket \varnothing \triangleright A^+ \rrbracket^{\textsc{in}}$, $\llbracket \varnothing \triangleright A^+ \rrbracket^{\textsc{op}}$. By definition we have:

$$\begin{aligned}
&\llbracket \varnothing \triangleright \Omega \rrbracket = \mathtt{unfold}(\llbracket \varnothing \triangleright \varsigma(X{=}P)\langle \ell = \varsigma(x:X)x\circ\ell\rangle \rrbracket).\ell^{sel}, \text{ where}\\
&\llbracket \varnothing \triangleright \varsigma(X{=}P)\langle \ell = \varsigma(x:X)x\circ\ell\rangle \rrbracket\\
&\qquad = \textsc{mkobj}_P(\Lambda(U \le \mathbb{P}^{\textsc{in}})\lambda(x:U^*)\mathtt{unfold}(x).\ell^{sel})\\
&\qquad = \mathtt{fold}(\mathbb{P}, [\ell^{gen} = \ldots,\\
&\qquad\qquad \ell^{sel} = \Lambda(U \le \mathbb{P}^{\textsc{in}})\lambda(x:U^*)\mathtt{unfold}(x).\ell^{sel}(\mathbb{P}^{\textsc{op}})(\textsc{self})\\
&\qquad\qquad ext = \ldots])\\
&\text{and } \textsc{self} = \textsc{mkobj}_P(\Lambda(U \le \mathbb{P}^{\textsc{in}})\lambda(x:U^*)\mathtt{unfold}(x).\ell^{sel})
\end{aligned}$$

Then, consider defining the interpretation of method addition as a simple call to the constructor function MKOBJ, thus disregarding the *ext* field. In our example, this would correspond to defining

$$[\![\varnothing \triangleright \Omega^+]\!] = \text{MKOBJ}_{A^+}(\Lambda(U \leq (\mathbb{A}^+)^{\text{IN}})\lambda(x : U^*)x)$$

Note that the generator ℓ^{gen} is not selected from Ω as it is not exposed in the type A given to Ω. This is a problem, because the call to the constructor function converges to the following result:

$$\begin{array}{l}\texttt{fold}(\mathbb{A}^+,\\ \quad [m^{gen} = \Lambda(U \leq (\mathbb{A}^+)^{\text{IN}})\lambda(x : U^*)x,\\ \quad m^{sel} = \Lambda(U \leq (\mathbb{A}^+)^{\text{IN}})\lambda(x : U^*)x\,(\mathbb{A}^+)^{\text{OP}}\,(\text{MKOBJ}_{A^+}(\Lambda(U \leq (\mathbb{A}^+)^{\text{IN}})\lambda(x : U^*))x)\\ \quad ext = \ldots])\end{array}$$

Summarizing, we have $[\![\varnothing \triangleright \Omega^+]\!] \Downarrow_f$. In the source calculus, instead, $\Omega^+ \not\Downarrow_o$, as evaluating Ω^+ requires Ω to evaluate to an object, while Ω diverges when selecting the label ℓ.

Computational adequacy is restored if, as in our translation, one enforces the evaluation of $[\![\varnothing \triangleright \Omega]\!]$ via the selection of the *ext* field. In that case, the translation gives the term:

$$\begin{array}{rcl}[\![\varnothing \triangleright \Omega^+]\!] & = & \texttt{unfold}([\![\varnothing \triangleright \Omega]\!]).ext((\mathbb{A}^+)^{\text{OP}})(\text{MKOBJ}_{A^+}(\cdots))\\ & = & \texttt{unfold}([\ell^{gen} = \ldots,\\ & & \qquad\qquad \ell^{sel} = \Lambda(U \leq \mathbb{P}^{\text{IN}})\lambda(x : U^*)\texttt{unfold}(x).\ell^{sel}(\mathbb{P}^{\text{OP}})(\text{SELF})\\ & & \qquad\qquad ext = \ldots].\ell^{sel}).ext((\mathbb{A}^+)^{\text{OP}})(\text{MKOBJ}_{A^+}(\cdots))\end{array}$$

which diverges as it tries to recursively select ℓ^{sel} prior to accessing the *ext* field from the resulting record.

In section 6 we give a formal proof of computational adequacy, showing that this use of the *ext* field works for arbitrary terms in the source calculus. Now, we conclude the presentation of our interpretation with the translation of contexts and judgements, in Figure 5. The definition derives directly from the type and term translations given above: note, in particular, that the translation of a judgement $\Gamma \vdash a : A$ does not depend on its derivation in $\texttt{Ob}^+$: as in [Abadi et al. 1996], we can thus avoid coherence issue in our soundness proofs.

5. SOUNDNESS OF THE ENCODING

The main result of this section is a proof that the encoding preserves the validity of judgements. The following notation and terminology are used throughout. For every ($\texttt{Ob}^+$ or $F_{\omega<:\mu}$) type A, $FTV(A)$ denotes the set of free type variables in A, defined in the usual way. We often write $\Gamma \vdash \Im$ to state that $\Gamma \vdash \Im$ is derivable, in either system. Also, we write $X \notin \Gamma$ to mean that X does not occur in (the types and declarations of) Γ, that is: $X \notin \Gamma \triangleq X \notin Dom(\Gamma)$ and $X \notin FTV(A)$ for all $x : A \in \Gamma$.

LEMMA 5.1 CORE PROPERTIES OF $\texttt{Ob}^+$.

(Contexts) *If* $\Gamma \vdash \Im$, *then* $\Gamma \vdash *$.

(Weakening) *If* $\Gamma \vdash \Im$ *and* $\Gamma, \Gamma' \vdash *$ *then* $\Gamma, \Gamma' \vdash \Im$.

$$
\begin{array}{rcl}
[\![\,\varnothing\,]\!] & \triangleq & \varnothing \\
[\![\,\Gamma, x{:}A\,]\!] & \triangleq & [\![\,\Gamma\,]\!], x{:}[\![\,\Gamma \triangleright A\,]\!]^{\text{TY}} \\
[\![\,\Gamma, X\,]\!] & \triangleq & [\![\,\Gamma\,]\!], X \\
[\![\,\Gamma, X {\lessdot\!\#} A\,]\!] & \triangleq & [\![\,\Gamma\,]\!], X \leq [\![\,\Gamma \triangleright A\,]\!]^{\text{IN}} \\
\\
[\![\,\Gamma \triangleright \varnothing\,]\!] & \triangleq & \varnothing \\
[\![\,\Gamma \triangleright \Gamma', x{:}A\,]\!] & \triangleq & [\![\,\Gamma \triangleright \Gamma'\,]\!], x{:}[\![\,\Gamma, \Gamma' \triangleright A\,]\!]^{\text{TY}} \\
[\![\,\Gamma \triangleright \Gamma', X\,]\!] & \triangleq & [\![\,\Gamma \triangleright \Gamma'\,]\!], X \\
[\![\,\Gamma \triangleright \Gamma', X {\lessdot\!\#} A\,]\!] & \triangleq & [\![\,\Gamma \triangleright \Gamma'\,]\!], X \leq [\![\,\Gamma, \Gamma' \triangleright A\,]\!]^{\text{IN}} \\
\\
[\![\,\Gamma \vdash *\,]\!] & \triangleq & [\![\,\Gamma\,]\!] \vdash \diamond \\
[\![\,\Gamma \vdash A\,]\!] & \triangleq & [\![\,\Gamma\,]\!] \vdash [\![\,\Gamma \triangleright A\,]\!]^{\text{TY}} \\
[\![\,\Gamma \vdash A {\lessdot\!\#} B\,]\!] & \triangleq & [\![\,\Gamma\,]\!] \vdash [\![\,\Gamma \triangleright A\,]\!]^{\text{IN}} \leq [\![\,\Gamma \triangleright B\,]\!]^{\text{IN}} \\
[\![\,\Gamma \vdash a : A\,]\!] & \triangleq & [\![\,\Gamma\,]\!] \vdash [\![\,\Gamma \triangleright a\,]\!] : [\![\,\Gamma \triangleright A\,]\!]^{\text{TY}}
\end{array}
$$

Fig. 5. Translation of Contexts and Judgements

(Exchange) *If* $\Gamma, X, \Gamma' \vdash \Im$ *and* $X \notin \Gamma'$, *then* $\Gamma, \Gamma', X \vdash \Im$. *Similarly, if* $\Gamma, U {\lessdot\!\#} A, \Gamma' \vdash \Im$ *and* $U \notin \Gamma'$, *then* $\Gamma, \Gamma', U {\lessdot\!\#} A \vdash \Im$. *The opposite implication holds as well, in both cases.*

(Substitution) *If* $\Gamma, X {\lessdot\!\#} A, \Gamma' \vdash \Im$ *and* $\Gamma \vdash B {\lessdot\!\#} A$, *then* $\Gamma, \Gamma'\{X{:=}B\} \vdash \Im\{X{:=}B\}$. *Similarly, if* $\Gamma, X, \Gamma' \vdash \Im$ *and* $\Gamma \vdash B$, *then* $\Gamma, \Gamma'\{X{:=}B\} \vdash \Im\{X{:=}B\}$.

PROOF. *By induction on derivations [Bono and Bugliesi 1999b].* □

LEMMA 5.2 CORE PROPERTIES OF $F_{\omega<:\mu}$.

(Weakening) *If* $\Gamma \vdash \Im$ *and* $\Gamma, \Gamma' \vdash \diamond$ *then* $\Gamma, \Gamma' \vdash \Im$.

(Exchange) *If* $\Gamma, X <: \mathbb{A} :: \mathtt{K}, \Gamma' \vdash \Im$ *and* $X \notin \Gamma'$, *then* $\Gamma, \Gamma', X <: \mathbb{A} :: \mathtt{K} \vdash \Im$.

(Substitution) *If* $\Gamma, X <: \mathbb{A} :: \mathtt{K}, \Gamma' \vdash \Im$ *and* $\Gamma \vdash \mathbb{B} <: \mathbb{A} :: \mathtt{K}$, *then* $\Gamma, \Gamma'\{X{:=}\mathbb{B}\} \vdash \Im\{X{:=}\mathbb{B}\}$.

PROOF. *By induction on derivations [Abadi and Cardelli 1996b].* □

5.1 Basic Properties of the Translation

The first lemma shows that the translation of well-formed $\mathtt{Ob}^+$ types is well defined: to ease the presentation, we often appeal only implicitly to this result throughout this and the following sections.

LEMMA 5.3. *If* $\Gamma \vdash A$ *in* $\mathtt{Ob}^+$, *then* $[\![\,\Gamma \triangleright A\,]\!]^{\text{TY}}$ *is well defined.*

PROOF. *By induction on the structure of* A. *The only interesting case is when* A *is a type variable, say* X, *as* $[\![\,\Gamma \triangleright X\,]\!]$ *depends on how* X *is declared in* Γ. *There are two possible sub-cases: either* $\Gamma \equiv \Gamma', X, \Gamma''$ *or* $\Gamma \equiv \Gamma', X {\lessdot\!\#} B, \Gamma''$ *for appropriate* B, Γ' *and* Γ''. *From the assumption that* $\Gamma \vdash A$, *by Lemma 5.1 (Contexts),* $\Gamma \vdash *$. *Hence, in both the sub-cases above,* $X \notin Dom(\Gamma, \Gamma')$: *now the claim follows by an inspection of the translation of types.* □

Next, we prove a few lemmas that establish some useful identities for the translation.

LEMMA 5.4 DOMAIN AND FREE TYPE VARIABLES.
Assume $\Gamma \vdash *$ *in* $\mathtt{Ob}^+$. *Then* $Dom(\Gamma) = Dom(\llbracket \Gamma \rrbracket)$. *Furthermore, if* $\Gamma \vdash A$, *then* $FTV(A) = FTV(\llbracket \Gamma \triangleright A \rrbracket^{\mathrm{TY}})$.

PROOF. *By induction on the derivations of the* $\mathtt{Ob}^+$ *judgements in the assumption.* □

LEMMA 5.5 CONTEXT SPLIT.
If $\Gamma, \Gamma' \vdash *$ *in* $\mathtt{Ob}^+$, *then* $\llbracket \Gamma, \Gamma' \rrbracket \equiv \llbracket \Gamma \rrbracket, \llbracket \Gamma \triangleright \Gamma' \rrbracket$.

PROOF. *By induction on the length* $|\Gamma'|$ *of* Γ'.

$|\Gamma'| = 0$. *In this case* $\Gamma' \equiv \varnothing$, *and the proof follows immediately since* $\llbracket \Gamma \triangleright \varnothing \rrbracket = \varnothing$ *by definition.*

$|\Gamma'| = n+1$. *In this case* $\Gamma' \equiv \Gamma'', \delta$, *where* δ *is one of* $X \lessdot\!\!\# A$, X, *or* $x : A$. *We take the case when* δ *is* $x : A$ *as representative, being the proof of the other cases similar. We have*

$$\begin{array}{rcll} \llbracket \Gamma, \Gamma'', x : A \rrbracket & \equiv & \llbracket \Gamma, \Gamma'' \rrbracket, x : \llbracket \Gamma, \Gamma'' \triangleright A \rrbracket & \textit{by definition} \\ & \equiv & \llbracket \Gamma \rrbracket, \llbracket \Gamma \triangleright \Gamma'' \rrbracket, x : \llbracket \Gamma, \Gamma'' \triangleright A \rrbracket & \textit{by induction hypothesis} \\ & \equiv & \llbracket \Gamma \rrbracket, \llbracket \Gamma \triangleright \Gamma'', x : A \rrbracket & \textit{by definition} \end{array}$$

LEMMA 5.6 EXCHANGE.
Let δ *be one of* $U \lessdot\!\!\# A$, X *or* $x : A$. *If* $\Gamma, \delta, \Gamma' \vdash A$ *and* $\Gamma, \Gamma', \delta \vdash *$ *in* $\mathtt{Ob}^+$, *then* $\llbracket \Gamma, \delta, \Gamma' \triangleright A \rrbracket^{\mathrm{TY}} \equiv \llbracket \Gamma, \Gamma', \delta \triangleright A \rrbracket^{\mathrm{TY}}$.

PROOF. *From the assumptions it follows that* $\Gamma, \Gamma', \delta \vdash A$ *in* $\mathtt{Ob}^+$*: this follows by induction on the derivation of* $\Gamma, \delta, \Gamma' \vdash A$. *Then the proof follows by an inspection of the clauses that define the translation.* □

LEMMA 5.7 WEAKENING.

(1) *If* $\Gamma \vdash A$ *and* $\Gamma, \Gamma' \vdash *$ *in* $\mathtt{Ob}^+$, *then* $\llbracket \Gamma \triangleright A \rrbracket^{\mathrm{TY}} \equiv \llbracket \Gamma, \Gamma' \triangleright A \rrbracket^{\mathrm{TY}}$.

(2) *If* $\Gamma \vdash a : A$ *and* $\Gamma, \Gamma' \vdash *$ *in* $\mathtt{Ob}^+$, *then* $\llbracket \Gamma \triangleright a \rrbracket \equiv \llbracket \Gamma, \Gamma' \triangleright a \rrbracket$.

PROOF. *The proof of (1) is by induction on the derivation of* $\Gamma \vdash A$. *(2) is also by induction, using (1). For (1), we proceed by cases depending on the last rule applied in the derivation of* $\Gamma \vdash A$.

(TYPE X) $\Gamma \vdash A$ *is the judgement* $\Gamma_1, X, \Gamma_2 \vdash X$ *for some* Γ_1 *and* Γ_2. *From* $\Gamma \vdash X$ *and* $\Gamma, \Gamma' \vdash *$, *by Lemma 5.1(Weakening),* $\Gamma, \Gamma' \vdash X$. *By Lemma 5.3, both* $\llbracket \Gamma \triangleright X \rrbracket^{\mathrm{TY}}$ *and* $\llbracket \Gamma, \Gamma' \triangleright X \rrbracket^{\mathrm{TY}}$ *are well defined, and, by definition,* $\llbracket \Gamma \triangleright X \rrbracket^{\mathrm{TY}} \equiv \llbracket \Gamma, \Gamma' \triangleright X \rrbracket^{\mathrm{TY}} \equiv X$.

(TYPE MATCH U) $\Gamma \vdash A$ *is the judgement* $\Gamma_1, X \lessdot\!\!\# B, \Gamma_2 \vdash X$. *Reasoning as in the previous case, we have, by definition* $\llbracket \Gamma \vdash X \rrbracket^{\mathrm{TY}} \equiv \llbracket \Gamma, \Gamma' \vdash X \rrbracket^{\mathrm{TY}} \equiv X^*$.

(TYPE $\mathtt{pro}$) *A is the type* $\mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i \in I}, m_j : B_j\{X\}^{j \in J} \rangle\!\rangle$, *and* $\Gamma \vdash A$ *is derived* *from* $\Gamma \vdash C_i$ *and* $\Gamma, X \vdash B_j\{X\}$ *for* $i \in I$ *and* $j \in J$. *By definition,*

$$\begin{array}{rcl}
[\![\, \Gamma \triangleright A \,]\!]^{\mathrm{TY}} & \triangleq & \mu(X)[\, v_i^{sel} : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}\ i\in I},\ v_i^{upd} : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}} \to X^{\ i\in I}, \\
 & & m_j^{sel} : [\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}\ j\in J}, \\
 & & m_j^{gen} : \forall(U \leq [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}})U^* \to [\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}}\{X{:=}U^*\}^{\ j\in J}, \\
 & & ext : \forall(U \leq [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}})U^* \to U^* \,]
\end{array}$$

where

$$\begin{array}{rcl}
[\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}} & \triangleq & \lambda(X)[\, v_i^{sel} : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}\ i\in I},\ v_i^{upd} : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}} \to X^{\ i\in I}, \\
 & & m_j^{sel} : [\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}\ j\in J} \,]
\end{array}$$

We may safely assume that $X \notin Dom(\Gamma')$, *as renaming of bound variables may always be used to enforce this assumption. Then, to prove the claim, it is enough to show that* $[\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}} \equiv [\![\, \Gamma, \Gamma' \triangleright C_i \,]\!]^{\mathrm{TY}}$ *for every* $i \in I$, *and that* $[\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}} \equiv [\![\, \Gamma, \Gamma', X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}}$ *for every* $j \in J$. *That* $[\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}} \equiv [\![\, \Gamma, \Gamma' \triangleright C_i \,]\!]^{\mathrm{TY}}$ *follows directly by the induction hypothesis on the judgements* $\Gamma \vdash C_i$. *For the remaining identity, from* $\Gamma, X \vdash B_j\{X\}$ $(j \in J)$, *by induction hypothesis* $[\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}} \equiv [\![\, \Gamma, X, \Gamma' \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}}$. *Furthermore, from* $\Gamma, X \vdash B_j\{X\}$ *by Lemma 5.1 (Contexts),* $\Gamma, X \vdash *$. *Then, from* $\Gamma, \Gamma' \vdash *$ *and* $X \notin Dom(\Gamma')$, *one has* $\Gamma, \Gamma', X \vdash *$. *Now, by Lemma 5.6,* $[\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}} \equiv [\![\, \Gamma, \Gamma', X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}}$, *as desired.*

□

LEMMA 5.8 SUBSTITUTION.

(*1*) *If* $\Gamma, X, \Gamma' \vdash A$ *and* $\Gamma \vdash C$ *in* $\mathtt{Ob}^+$, *then*

$$[\![\, \Gamma, X, \Gamma' \triangleright A \,]\!]^{\mathrm{TY}}\{X{:=}[\![\, \Gamma \triangleright C \,]\!]^{\mathrm{TY}}\} \equiv [\![\, \Gamma, \Gamma'\{X{:=}C\} \triangleright A\{X{:=}C\} \,]\!]^{\mathrm{TY}}$$

(*2*) *If* $\Gamma, U \lessdot\!\# A, \Gamma' \vdash B\{U\}$ *and* $\Gamma \vdash C \lessdot\!\# A$ *in* $\mathtt{Ob}^+$, *then*

$$[\![\, \Gamma, U \lessdot\!\# A, \Gamma' \triangleright B\{U\} \,]\!]^{\mathrm{TY}}\{U := [\![\, \Gamma \triangleright C \,]\!]^{\mathrm{OP}}\} = [\![\, \Gamma, \Gamma\{U := C\}' \triangleright B\{C\} \,]\!]^{\mathrm{TY}}$$

PROOF. *We prove (1), by induction on the derivation of* $\Gamma, X, \Gamma' \vdash A$. *The proof of (2) is similar, also by induction, on the derivation of* $\Gamma, U \lessdot\!\# A, \Gamma' \vdash B\{U\}$.

(TYPE X) *We distinguish two cases depending on whether* X *is the projected variable or not.*

If X *is the projected variable, the judgement in question is* $\Gamma, X, \Gamma' \vdash X$. *For the left-hand side of the identity, by definition,* $[\![\, \Gamma, X, \Gamma' \triangleright X \,]\!]^{\mathrm{TY}} \equiv X$, *from which* $[\![\, \Gamma, X, \Gamma' \triangleright X \,]\!]^{\mathrm{TY}}\{X{:=}[\![\, \Gamma \triangleright C \,]\!]^{\mathrm{TY}}\} \equiv [\![\, \Gamma \triangleright C \,]\!]^{\mathrm{TY}}$. *For the right-hand side, we first note that* $\Gamma, X, \Gamma' \vdash X$ *implies that* $\Gamma, X, \Gamma' \vdash *$. *From this, and from the assumption* $\Gamma \vdash C$, *by Lemma 5.1(Substitution),* $\Gamma, \Gamma'\{X := C\} \vdash *$. *From the last judgement, and from* $\Gamma \vdash C$, *by Lemma 5.7,* $[\![\, \Gamma \triangleright C \,]\!]^{\mathrm{TY}} \equiv [\![\, \Gamma, \Gamma'\{X := C\} \triangleright C \,]\!]^{\mathrm{TY}}$. *This proves the claim as* $C \equiv X\{X := C\}$.

If X *is not the projected variable, the judgement is* $\Gamma, X, \Gamma' \vdash Y$, *for some* $Y \in Dom(\Gamma, \Gamma'), Y \neq X$, *and the identity to be proved is* $[\![\, \Gamma, X, \Gamma' \triangleright Y \,]\!]^{\mathrm{TY}} \equiv [\![\, \Gamma, \Gamma'\{X{:=}C\} \triangleright Y \,]\!]^{\mathrm{TY}}$. *Reasoning as above, we derive* $\Gamma, X, \Gamma' \vdash *$ *and* $\Gamma, \Gamma'\{X := C\} \vdash *$. *Now, if* $Y \in Dom(\Gamma)$, *one has* $\Gamma \vdash Y$ *and, by Lemma 5.7,* $[\![\, \Gamma \triangleright Y \,]\!]^{\mathrm{TY}} \equiv [\![\, \Gamma, X, \Gamma' \triangleright Y \,]\!]^{\mathrm{TY}}$ *and* $[\![\, \Gamma \triangleright Y \,]\!]^{\mathrm{TY}} \equiv [\![\, \Gamma, \Gamma'\{X{:=}C\} \triangleright Y \,]\!]^{\mathrm{TY}}$. *The claim follows by transitivity from be last two identities. The case when* $Y \in Dom(\Gamma')$ *is similar, as* $Y \neq X$ *implies* $Y \in Dom(\Gamma'\{X{:=}C\})$.

(Type Match U) *Similar to the previous case.*

(Type pro) A *is the type* $\mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$. *By α-conversion, there exists Y fresh so that* $A \equiv \mathtt{pro}(Y)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{Y\}^{j\in J}\rangle\!\rangle$, *with* $\Gamma, X, \Gamma' \vdash \mathtt{pro}(Y)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{Y\}^{j\in J}\rangle\!\rangle$ *derived from* $\Gamma, X, \Gamma' \vdash C_i$, $i \in I$, *and from* $\Gamma, X, \Gamma', Y \vdash B_j\{Y\}$ *for* $j \in J$. *Letting* $\overline{\Gamma}$ *be the context* Γ, X, Γ', *by definition:*

$$\begin{aligned}[\![\,\overline{\Gamma} \triangleright A\,]\!]^{\mathrm{TY}} \triangleq \mu(Y)[\, & v_i^{sel} : [\![\,\overline{\Gamma} \triangleright C_i\,]\!]^{\mathrm{TY}}\ {}^{i\in I},\ v_i^{upd} : [\![\,\overline{\Gamma} \triangleright C_i\,]\!]^{\mathrm{TY}} \to Y\ {}^{i\in I},\\ & m_j^{sel} : [\![\,\overline{\Gamma}, Y \triangleright B_j\{Y\}\,]\!]^{\mathrm{TY}}\ {}^{j\in J},\\ & m_j^{gen} : \forall(U \leq [\![\,\overline{\Gamma} \triangleright A\,]\!]^{\mathrm{IN}})U^* \to [\![\,\overline{\Gamma}, Y \triangleright B_j\{Y\}\,]\!]^{\mathrm{TY}}\{Y{:=}U^*\}\ {}^{j\in J},\\ & ext : \forall(U \leq [\![\,\overline{\Gamma} \triangleright A\,]\!]^{\mathrm{IN}})U^* \to U^*\ ]\end{aligned}$$

where

$$\begin{aligned}[\![\,\overline{\Gamma} \triangleright A\,]\!]^{\mathrm{IN}} \triangleq \lambda(Y)[\, & v_i^{sel} : [\![\,\overline{\Gamma} \triangleright C_i\,]\!]^{\mathrm{TY}}\ {}^{i\in I},\ v_i^{upd} : [\![\,\overline{\Gamma} \triangleright C_i\,]\!]^{\mathrm{TY}} \to Y\ {}^{i\in I},\\ & m_j^{sel} : [\![\,\overline{\Gamma}, Y \triangleright B_j\{Y\}\,]\!]^{\mathrm{TY}}\ {}^{j\in J}\ ]\end{aligned}$$

Now the claim follows by the induction hypothesis on the judgements $\overline{\Gamma} \vdash C_i$ *and* $\overline{\Gamma}, Y \vdash B_j\{Y\}$.

□

Corollary 5.9. *If* $\Gamma, X \vdash B$ *and* $\Gamma, U \lessdot\!\# A \vdash *$ *in* $\mathtt{Ob}^+$, *then*

$$[\![\,\Gamma, X \triangleright B\,]\!]^{\mathrm{TY}}\{X{:=}U^*\} \equiv [\![\,\Gamma, U \lessdot\!\# A \triangleright B\{X{:=}U\}\,]\!]^{\mathrm{TY}}$$

Proof. *By Lemma 5.1 (Contexts) from* $\Gamma, X \vdash B$ *one has* $\Gamma, X \vdash *$, *which implies* $X \notin Dom(\Gamma)$. *Then, from* $\Gamma, U \lessdot\!\# A \vdash *$, *by Lemma 5.1(Weakening),* $\Gamma, U \lessdot\!\# A, X \vdash *$, *From this, and from* $\Gamma, X \vdash B$, *again by Lemma 5.1 (Weakening) and (Exchange),* $\Gamma, U \lessdot\!\# A, X \vdash B$. *Furthermore, from* $\Gamma, U \lessdot\!\# A \vdash *$, *by (Type Match U), one derives* $\Gamma, U \lessdot\!\# A \vdash U$. *By Lemma 5.8.1,*

$$[\![\,\Gamma, U \lessdot\!\# A, X \triangleright B\,]\!]^{\mathrm{TY}}\{X{:=}[\![\,\Gamma, U \lessdot\!\# A \triangleright U\,]\!]^{\mathrm{TY}}\} \equiv [\![\,\Gamma, U \lessdot\!\# A \triangleright B\{X{:=}U\}\,]\!]^{\mathrm{TY}}.$$

By definition, $[\![\,\Gamma, U \lessdot\!\# A \triangleright U\,]\!]^{\mathrm{TY}} \equiv U^*$. *Hence the previous identity is*

$$[\![\,\Gamma, U \lessdot\!\# A, X \triangleright B\,]\!]^{\mathrm{TY}}\{X{:=}U^*\} \equiv [\![\,\Gamma, U \lessdot\!\# A \triangleright B\{X{:=}U\}\,]\!]^{\mathrm{TY}}.$$

To conclude, note that $[\![\,\Gamma, X \triangleright B\,]\!]^{\mathrm{TY}} \equiv [\![\,\Gamma, U \lessdot\!\# A, X \triangleright B\,]\!]^{\mathrm{TY}}$ *by Lemma 5.7 and Lemma 5.6.* □

5.2 Main Properties of the Translation

Lemma 5.10 Kinding and Context formation.

(*1*) *If* $\Gamma \vdash *$ *in* $\mathtt{Ob}^+$ *then* $[\![\,\Gamma\,]\!] \vdash \diamond$ *in* $F_{\omega<:\mu}$.
(*2*) *If* $\Gamma \vdash A$ *in* $\mathtt{Ob}^+$ *then* $[\![\,\Gamma\,]\!] \vdash [\![\,\Gamma \triangleright A\,]\!]^{\mathrm{TY}}$ *in* $F_{\omega<:\mu}$.

Proof. *By simultaneous induction on the judgements in the assumptions.*

(*1*) *If* $\Gamma \vdash *$ *in* $\mathtt{Ob}^+$ *then* $[\![\,\Gamma\,]\!] \vdash \diamond$ *in* $F_{\omega<:\mu}$.

(Ctx $\oslash$) $\Gamma \vdash \Im$ *is the judgement* $\varnothing \vdash *$, *its translation is* $\varnothing \vdash \diamond$, *which is derivable in* $F_{\omega<:\mu}$.

(Ctx X) $\Gamma \vdash \Im$ *is the judgement* $\Gamma, X \vdash *$, *derived from* $\Gamma \vdash *$ *and* $X \notin Dom(\Gamma)$. *By induction hypothesis* **(1)** $[\![\,\Gamma\,]\!] \vdash \diamond$ *in* $F_{\omega<:\mu}$, *and* $[\![\,\Gamma\,]\!], X \vdash \diamond$ *(the translation of* $\Gamma \vdash \Im$*) derives by (Env X <:) as* $X \notin Dom([\![\,\Gamma\,]\!])$ *by Lemma 5.4.*

(Ctx Match) *In this case* $\Gamma \vdash \Im$ *is the judgement* $\Gamma, U \lll\!\!\!\# A \vdash *$, *derived from* $\Gamma \vdash A$ *for* $U \notin Dom(\Gamma)$. *An inspection of the type rules shows that* $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$. *By definition,*

$$\begin{aligned}\llbracket \Gamma \triangleright A \rrbracket^{\mathrm{TY}} \triangleq \mu(X)[\, & v_i^{sel} : \llbracket \Gamma \triangleright C_i \rrbracket^{\mathrm{TY}\ i\in I},\ v_i^{upd} : \llbracket \Gamma \triangleright C_i \rrbracket^{\mathrm{TY}} \to X^{\ i\in I}, \\ & m_j^{sel} : \llbracket \Gamma, X \triangleright B_j\{X\} \rrbracket^{\mathrm{TY}\ j\in J}, \\ & m_j^{gen} : \forall(U \leq \llbracket \Gamma \triangleright A \rrbracket^{\mathrm{IN}})U^* \to \llbracket \Gamma, X \triangleright B_j\{X\} \rrbracket^{\mathrm{TY}}\{X{:=}U^*\}^{j\in J}, \\ & ext : \forall(U \leq \llbracket \Gamma \triangleright A \rrbracket^{\mathrm{IN}})U^* \to U^*\ ]\end{aligned}$$

where

$$\begin{aligned}\llbracket \Gamma \triangleright A \rrbracket^{\mathrm{IN}} \triangleq \lambda(X)[\, & v_i^{sel} : \llbracket \Gamma \triangleright C_i \rrbracket^{\mathrm{TY}\ i\in I},\ v_i^{upd} : \llbracket \Gamma \triangleright C_i \rrbracket^{\mathrm{TY}} \to X^{\ i\in I}, \\ & m_j^{sel} : \llbracket \Gamma, X \triangleright B_j\{X\} \rrbracket^{\mathrm{TY}\ j\in J}\ ]\end{aligned}$$

From $\Gamma \vdash A$, *by induction hypothesis* **(2)**, $\llbracket \Gamma \rrbracket \vdash \llbracket \Gamma \triangleright A \rrbracket^{\mathrm{TY}}$. *From this, the judgement* $\llbracket \Gamma \rrbracket \vdash \llbracket \Gamma \triangleright A \rrbracket^{\mathrm{IN}} :: Op$ *derives routinely in* $F_{\omega<:\mu}$. *From* $U \notin Dom(\Gamma)$, *by Lemma 5.4,* $U \notin Dom(\llbracket \Gamma \rrbracket)$. *Then, the* $\llbracket \Gamma \rrbracket, U <: \llbracket \Gamma \triangleright A \rrbracket^{\mathrm{IN}} :: Op \vdash \diamond$ *(the translation of* $\Gamma \vdash \Im$*) derives by (Env* $X <:$ *).*

(2) *If* $\Gamma \vdash A$ *in* $\mathtt{Ob}^+$ *then* $\llbracket \Gamma \rrbracket \vdash \llbracket \Gamma \triangleright A \rrbracket^{\mathrm{TY}}$ *in* $F_{\omega<:\mu}$.

(Type X) $\Gamma \vdash \Im$ *is the judgement* $\Gamma', X, \Gamma'' \vdash X$, *derived from* $\Gamma, X, \Gamma'' \vdash *$, *for appropriate* Γ' *and* Γ''. *From the last judgement, by Lemma 5.5 and the translation of contexts (cf. Figure 5),* $\llbracket \Gamma', X, \Gamma'' \rrbracket \equiv \llbracket \Gamma' \rrbracket, X, \llbracket \Gamma', X \triangleright \Gamma'' \rrbracket$. *Again from* $\Gamma, X, \Gamma'' \vdash *$, *by induction hypothesis* **(1)**, $\llbracket \Gamma', X, \Gamma'' \rrbracket \vdash \diamond$. *From this, by the identity we just proved,* $\llbracket \Gamma' \rrbracket, X, \llbracket \Gamma', X \triangleright \Gamma'' \rrbracket \vdash \diamond$. *From the last judgement, by (Con* X*),* $\llbracket \Gamma' \rrbracket, X, \llbracket \Gamma', X \triangleright \Gamma'' \rrbracket \vdash X$, *which is the translation of* $\Gamma \vdash \Im$.

(Type Match U) *Similar to the previous case.*

(Type $\mathtt{pro}$) *In this case we have that* $\Gamma \vdash \Im$ *is the judgement* $\Gamma \vdash A$, *with* $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$, *and* $\Gamma \vdash A$ *derived from* $\Gamma \vdash C_i$ *and* $\Gamma, X \vdash B_j\{X\}$ *for every* $i \in I$, $j \in J$. *By induction hypothesis* **(2)**, *we have* $\llbracket \Gamma \rrbracket \vdash \llbracket \Gamma \triangleright C_i \rrbracket^{\mathrm{TY}}$ *and* $\llbracket \Gamma \rrbracket, X \vdash \llbracket \Gamma, X \triangleright B_j\{X\} \rrbracket^{\mathrm{TY}}$. *By definition,*

$$\begin{aligned}\llbracket \Gamma \triangleright A \rrbracket^{\mathrm{TY}} \triangleq \mu(X)[\, & v_i^{sel} : \llbracket \Gamma \triangleright C_i \rrbracket^{\mathrm{TY}\ i\in I},\ v_i^{upd} : \llbracket \Gamma \triangleright C_i \rrbracket^{\mathrm{TY}} \to X^{\ i\in I}, \\ & m_j^{sel} : \llbracket \Gamma, X \triangleright B_j\{X\} \rrbracket^{\mathrm{TY}\,j\in J}, \\ & m_j^{gen} : \forall(U \leq \llbracket \Gamma \triangleright A \rrbracket^{\mathrm{IN}})U^* \to \llbracket \Gamma, X \triangleright B_j\{X\} \rrbracket^{\mathrm{TY}}\{X{:=}U^*\}^{\ j\in J}, \\ & ext : \forall(U \leq \llbracket \Gamma \triangleright A \rrbracket^{\mathrm{IN}})U^* \to U^*\ ]\end{aligned}$$

where

$$\begin{aligned}\llbracket \Gamma \triangleright A \rrbracket^{\mathrm{IN}} \triangleq \lambda(X)[\, & v_i^{sel} : \llbracket \Gamma \triangleright C_i \rrbracket^{\mathrm{TY}\ i\in I},\ v_i^{upd} : \llbracket \Gamma \triangleright C_i \rrbracket^{\mathrm{TY}} \to X^{\ i\in I}, \\ & m_j^{sel} : \llbracket \Gamma, X \triangleright B_j\{X\} \rrbracket^{\mathrm{TY}\ j\in J}\ ]\end{aligned}$$

To verify that $\llbracket \Gamma \rrbracket \vdash \llbracket \Gamma \triangleright A \rrbracket^{\mathrm{TY}}$, *it remains to show that the following judgements are derivable in* $F_{\omega<:\mu}$, *where* U *is a fresh variable not in* $Dom(\Gamma)$, *hence not in* $Dom(\llbracket \Gamma \rrbracket)$.

(i) $\llbracket \Gamma \rrbracket \vdash \llbracket \Gamma \triangleright A \rrbracket^{\mathrm{IN}} :: Op$. *This follows from* $\llbracket \Gamma \rrbracket, X \vdash \llbracket \Gamma, X \triangleright B_j\{X\} \rrbracket^{\mathrm{TY}}$, *and from* $\llbracket \Gamma \rrbracket \vdash \llbracket \Gamma \triangleright C_i \rrbracket^{\mathrm{TY}}$ *by an inspection of the typing rules of* $F_{\omega<:\mu}$.

(*ii*) $[\![\Gamma]\!], X, U \leq [\![\Gamma \triangleright A]\!]^{\text{IN}} \vdash U^*$, *where* $U^* \equiv \mu(Y)U(Y)$. *From* $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A]\!]^{\text{IN}} :: Op$ *(proved above), one has* $[\![\Gamma]\!], U \leq [\![\Gamma \triangleright A]\!]^{\text{IN}} \vdash \diamond$. *From this, and* $U \notin Dom(\Gamma)$, $[\![\Gamma]\!], U \leq [\![\Gamma \triangleright A]\!]^{\text{IN}} \vdash U :: Op$. *Then, by By Lemma 5.2 (Weakening), one derives* $[\![\Gamma]\!], U \leq [\![\Gamma \triangleright A]\!]^{\text{IN}}, Y \vdash U :: Op$. *Since* $[\![\Gamma]\!], U \leq [\![\Gamma \triangleright A]\!]^{\text{IN}}, Y \vdash Y$, *by (Con Appl) and (Con* μ*),* $[\![\Gamma]\!], U \leq [\![\Gamma \triangleright A]\!]^{\text{IN}} \vdash U^*$. *Now the claim follows by Lemma 5.2(Weakening) and (Exchange).*

(*iii*) $[\![\Gamma]\!], X, U \leq [\![\Gamma \triangleright A]\!]^{\text{IN}} \vdash [\![\Gamma, X \triangleright B_i\{X\}]\!]^{\text{TY}}\{X{:=}U^*\}$. *From the* $F_{\omega<:\mu}$ *judgements* $[\![\Gamma]\!], X \vdash [\![\Gamma, X \triangleright B_j\{X\}]\!]^{\text{TY}}$, *by Lemmas 5.2 (Weakening) and (Exchange), one derives* $[\![\Gamma]\!], U \leq [\![\Gamma \triangleright A]\!]^{\text{IN}}, X \vdash [\![\Gamma, X \triangleright B_j\{X\}]\!]^{\text{TY}}$. *By Lemma 5.7 and Lemma 5.6 this judgement is* $[\![\Gamma]\!], U \leq [\![\Gamma \triangleright A]\!]^{\text{IN}}, X \vdash [\![\Gamma, U \lessdot\!\!\# A, X \triangleright B_j\{X\}]\!]^{\text{TY}}$. *Reasoning as in* (*ii*) *above,* $[\![\Gamma]\!], U \leq [\![\Gamma \triangleright A]\!]^{\text{IN}} \vdash U^*$. *From this, by Lemma 5.2 (Substitution), we have* $[\![\Gamma]\!], U \leq [\![\Gamma \triangleright A]\!]^{\text{IN}} \vdash [\![\Gamma, U \lessdot\!\!\# A, X \triangleright B_j\{X\}]\!]^{\text{TY}}\{X{:=}U^*\}$. *Again by Lemmas 5.7 and 5.6 this is the judgement we wished to derive.*

□

LEMMA 5.11 MATCHING.
If $\Gamma \vdash A \lessdot\!\!\# B$ *in* $\texttt{Ob}^+$, *then* $[\![\Gamma \vdash A \lessdot\!\!\# B]\!]$ *in* $F_{\omega<:\mu}$.

PROOF. *By induction on the derivation of* $\Gamma \vdash A \lessdot\!\!\# B$.

(MATCH U) *The judgement in question is* $\Gamma', U \lessdot\!\!\# A, \Gamma'' \vdash U \lessdot\!\!\# A$, *derived from* $\Gamma', U \lessdot\!\!\# A, \Gamma'' \vdash *$. *By Lemma 5.10,* $[\![\Gamma', U \lessdot\!\!\# A, \Gamma'']\!] \vdash \diamond$ *is derivable in* $F_{\omega<:\mu}$; *by Lemma 5.5,* $[\![\Gamma', U \lessdot\!\!\# A, \Gamma'']\!] \equiv [\![\Gamma']\!], U \leq [\![\Gamma' \triangleright A]\!]^{\text{IN}}, [\![\Gamma', U \lessdot\!\!\# A \triangleright \Gamma'']\!]$. *By (Con Sub X),*

$$[\![\Gamma']\!], U \leq [\![\Gamma' \triangleright A]\!]^{\text{IN}}, [\![\Gamma', U \lessdot\!\!\# A \triangleright \Gamma'']\!] \vdash U \leq [\![\Gamma' \triangleright A]\!]^{\text{IN}}.$$

By Lemma 5.7, this judgement is

$$[\![\Gamma']\!], U \leq [\![\Gamma' \triangleright A]\!]^{\text{IN}}, [\![\Gamma', U \lessdot\!\!\# A \triangleright \Gamma'']\!] \vdash U \leq [\![\Gamma', X \lessdot\!\!\# A, \Gamma'' \triangleright A]\!]^{\text{IN}},$$

which is the translation of $\Gamma \vdash A \lessdot\!\!\# B$.

(MATCH REFL) (MATCH TRANS) *By induction hypothesis, and by (Cons Sub Trans) to handle the case of (Match Trans).*

(MATCH $\texttt{pro}$) *In this case the judgement in question is* $\Gamma \vdash A \lessdot\!\!\# A'$, *and it is derived from* $\Gamma \vdash A$, *with* $A \equiv \texttt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$, *and* $A' \equiv \texttt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I'}, m : B_i\{X\}^{j\in J'}\rangle\!\rangle$, *for* $I' \subseteq I$ *and* $J' \subseteq J$. *Then also* $\Gamma \vdash C_i$ *and* $\Gamma, X \vdash B_i\{X\}$ *in* $\texttt{Ob}^+$ *for every* $i \in I$ *and* $j \in J$. *By Lemma 5.10,* $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright C_i]\!]^{\text{TY}}$ *and* $[\![\Gamma]\!], X \vdash [\![\Gamma, X \triangleright B_i\{X\}]\!]^{\text{TY}}$ *in* $F_{\omega<:\mu}$, *with* i *and* j *ranging over* I *and* J. *From these judgements,* $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A]\!]^{\text{IN}} \leq [\![\Gamma \triangleright A']\!]^{\text{IN}}$ *derives in* $F_{\omega<:\mu}$ *routinely. This concludes the proof, as the last judgement is the translation of* $\Gamma \vdash A \lessdot\!\!\# A'$.

□

We need two more lemmas to prove that the translations preserve the validity of typing judgements.

LEMMA 5.12 TYPES VS OPERATORS.
Assume $\Gamma \vdash A$ *in* $\mathtt{Ob}^+$. *If* $[\![\,\Gamma \triangleright A\,]\!]^{\mathrm{OP}}$ *is defined, then* $[\![\,\Gamma \triangleright A\,]\!]^{\mathrm{TY}} \equiv ([\![\,\Gamma \triangleright A\,]\!]^{\mathrm{OP}})^*$.

PROOF. *By induction on the derivation of* $\Gamma \vdash A$.

(TYPE X) $\Gamma \vdash A$ *is the judgement* $\Gamma', X, \Gamma'' \vdash X$, *and the claim follows vacuously, since* $[\![\,\Gamma \triangleright A\,]\!]^{\mathrm{OP}}$ *is not defined, as* Γ *is well formed.*

(TYPE MATCH U) (TYPE $\mathtt{pro}$) *Directly by the definition.*

□

LEMMA 5.13 OPERATORS VS INTERFACES.
If $\Gamma \vdash A$ *is derivable in* $\mathtt{Ob}^+$, *and* $[\![\,\Gamma \triangleright A\,]\!]^{\mathrm{OP}}$ *is defined, then* $[\![\,\Gamma\,]\!] \vdash [\![\,\Gamma \triangleright A\,]\!]^{\mathrm{OP}} \leq [\![\,\Gamma \triangleright A\,]\!]^{\mathrm{IN}}$ *is derivable in* $F_{\omega<:\mu}$.

PROOF. *By induction on the derivation of* $\Gamma \vdash A$.

(TYPE X) *We have* $\Gamma \vdash A$ *is the judgement* $\Gamma', X, \Gamma'' \vdash X$, *and the claim follows vacuously since* $[\![\,\Gamma \triangleright A\,]\!]^{\mathrm{OP}}$ *is not defined.*

(TYPE MATCH U) $\Gamma \vdash A$ *is the judgement* $\Gamma', U \lessdot\!\!\# C, \Gamma'' \vdash U$, *for appropriate* Γ, Γ' *and* C. *By definition,* $[\![\,\Gamma', U \lessdot\!\!\# C, \Gamma'' \triangleright U\,]\!]^{\mathrm{OP}} \equiv [\![\,\Gamma', U \lessdot\!\!\# C, \Gamma'' \triangleright U\,]\!]^{\mathrm{IN}} \equiv U$. *By Lemma 5.5,* $[\![\,\Gamma', U \lessdot\!\!\# C, \Gamma''\,]\!] \equiv [\![\,\Gamma'\,]\!], U \leq [\![\,\Gamma' \triangleright C\,]\!]^{\mathrm{IN}}, [\![\,\Gamma', U \lessdot\!\!\# C \triangleright \Gamma''\,]\!]$. *Now the desired judgement derives by (Con X) and (Con Sub Refl).*

(TYPE $\mathtt{pro}$) *We have* $A \equiv \mathtt{pro}(X)\langle\!\langle \ell_i{:}B_i\{X\}^{i \in I}\rangle\!\rangle$, *and* $\Gamma \vdash A$ *derives from* $\Gamma, X \vdash B_i\{X\}$, *for* $i \in I$. *By Lemma 5.10, we know that* $[\![\,\Gamma\,]\!], X \vdash [\![\,\Gamma, X \triangleright B_i\{X\}\,]\!]^{\mathrm{TY}}$, *(*$i \in I$*). From these judgements,* $[\![\,\Gamma\,]\!] \vdash [\![\,\Gamma \triangleright A\,]\!]^{\mathrm{OP}} \leq [\![\,\Gamma \triangleright A\,]\!]^{\mathrm{IN}}$ *derives by (Con Sub Abs) and (Con Sub Record).*

□

THEOREM 5.14 VALIDATION OF TYPING.
If the judgement $\Gamma \vdash a : A$ *is derivable in* $\mathtt{Ob}^+$, *then the judgement* $[\![\,\Gamma\,]\!] \vdash [\![\,\Gamma \triangleright a\,]\!] : [\![\,\Gamma \triangleright A\,]\!]^{\mathrm{TY}}$ *is derivable in* $F_{\omega<:\mu}$.

PROOF. *By induction on the derivation of* $\Gamma \vdash a : A$.

(VAL x) *The judgement in question is* $\Gamma', x : A, \Gamma'' \vdash x : A$, *derived from* $\Gamma', x : A, \Gamma'' \vdash *$. *By Lemma 5.10, the judgement* $[\![\,\Gamma', x : A, \Gamma''\,]\!] \vdash \diamond$ *is derivable in* $F_{\omega<:\mu}$. *By Lemma 5.5* $[\![\,\Gamma', x : A, \Gamma''\,]\!] \equiv [\![\,\Gamma\,]\!], x : [\![\,\Gamma' \triangleright A\,]\!]^{\mathrm{TY}}, [\![\,\Gamma, x : A \triangleright \Gamma''\,]\!]$. *Hence, by (Val x)* $[\![\,\Gamma', x : A, \Gamma''\,]\!] \vdash x : [\![\,\Gamma' \triangleright A\,]\!]^{\mathrm{TY}}$ *in* $F_{\omega<:\mu}$. *This concludes the proof, as the last judgement is the translation of* $\Gamma', x : A, \Gamma'' \vdash x : A$. *In fact,* $[\![\,\Gamma \triangleright x\,]\!] \equiv x$ *and, by Lemma 5.7* $[\![\,\Gamma \triangleright A\,]\!]^{\mathrm{TY}} \equiv [\![\,\Gamma', x : A, \Gamma'' \triangleright A\,]\!]^{\mathrm{TY}}$.

(VAL OBJECT) *The judgement is* $\Gamma \vdash \varsigma(X{=}A)\langle v_i = c_i{}^{i \in I}, m_j = \varsigma(x : X)b_j\{X, x\}^{j \in J}\rangle : A$ *derived from the judgements* $\Gamma \vdash c_i : C_i$, $i \in I$ *and* $\Gamma, U \lessdot\!\!\# A, x : U \vdash b_j : B_j\{U, x\}$, $j \in J$, *for* $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i \in I}, m_j : B_j\{X\}^{j \in J}\rangle\!\rangle$. *By induction hypothesis, the following judgements are all derivable in* $F_{\omega<:\mu}$ *(for* $i \in I$ *and* $j \in J$*):*

$$[\![\,\Gamma\,]\!] \vdash [\![\,\Gamma \triangleright c_i\,]\!] : [\![\,\Gamma \triangleright C_i\,]\!]^{\mathrm{TY}} \tag{1}$$

$$[\![\,\Gamma, U \lessdot\!\!\# A, x : U\,]\!] \vdash [\![\,\Gamma, U \lessdot\!\!\# A, x : U \triangleright b_j\{U, x\}\,]\!] : [\![\,\Gamma, U \lessdot\!\!\# A, x : U \triangleright B_j\{U\}\,]\!]^{\mathrm{TY}} \tag{2}$$

By Lemma 5.5, and the translation of contexts, $[\![\Gamma, U \lll\# A, x{:}U]\!] \equiv [\![\Gamma]\!], U \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}}, x{:}U^*$. *Then, from the judgements in (2), by (Val Abs) and (Val Abs2):*

$$\begin{array}{rl} [\![\Gamma]\!] \ \vdash & \Lambda(U \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}})\lambda(x : U^*)[\![\Gamma, U \lll\# A, x : U \triangleright b_j\{U, x\}]\!] \\ : & \forall(U \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}})U^* \to [\![\Gamma, U \lll\# A, x : U \triangleright B_j\{U\}]\!]^{\mathrm{TY}} \qquad j \in J. \end{array}$$

By Lemma 5.7 and Corollary 5.9, we have that $[\![\Gamma, U \lll\# A, x : U \triangleright B_j\{U\}]\!]^{\mathrm{TY}} \equiv [\![\Gamma, X \triangleright B_j\{X\}]\!]^{\mathrm{TY}}\{X{:=}U^*\}$. *Now the proof follows by an inspection of the recursive definition of* MKOBJ_A, *and the typing rules of* $F_{\omega<:\mu}$.

(Val Extend) *We take the case of method addition as representative: the case of field addition is similar. For the case in question, the judgement is* $\Gamma \vdash a \circ m \longleftrightarrow \varsigma(X{=}A^+)\varsigma(s : X)b\{X, x\} : A^+$, *that must have been derived from the judgements* $\Gamma \vdash a : A$, *and from* $\Gamma, U \lll\# A^+, x : U \vdash b\{U, x\} : B\{U\}$, *with* $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$, $A^+ \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m : B\{X\}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$. *By induction hypothesis, the following judgements are both derivable in* $F_{\omega<:\mu}$*:*

$$[\![\Gamma]\!] \ \vdash \ [\![\Gamma \triangleright a]\!] : [\![\Gamma \triangleright A]\!]^{\mathrm{TY}} \tag{1}$$

$$[\![\Gamma, U \lll\# A^+, x : U]\!] \ \vdash \ [\![\Gamma, U \lll\# A^+, x : U \triangleright b\{U, x\}]\!] : [\![\Gamma, U \lll\# A^+, x : U \triangleright B\{U\}]\!]^{\mathrm{TY}} \tag{2}$$

Form (1), by (val Unfold) $[\![\Gamma]\!] \vdash \mathtt{unfold}([\![\Gamma \triangleright a]\!]) : [\![\Gamma \triangleright A]\!]^{\mathrm{OP}}([\![\Gamma \triangleright A]\!]^{\mathrm{TY}})$. *From this judgement, using the definition of* $[\![\Gamma \triangleright A]\!]^{\mathrm{OP}}$, *by (Con eval Beta), (Con Sub Refl) and (Val Subsumption):*

$$\begin{array}{rl} [\![\Gamma]\!] \vdash \mathtt{unfold}([\![\Gamma \triangleright a]\!]) : & [v_i^{sel} : [\![\Gamma \triangleright C_i]\!]^{\mathrm{TY}\ i\in I},\ v_i^{upd} : [\![\Gamma \triangleright C_i]\!]^{\mathrm{TY}} \to X^{\ i\in I}, \\ & m_j^{sel} : [\![\Gamma, X \triangleright B_j\{X\}]\!]^{\mathrm{TY}\ j\in J}, \\ & m_j^{gen} : \forall(U \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}})U^* \to [\![\Gamma, X \triangleright B_j\{X\}]\!]^{\mathrm{TY}}\{X{:=}U^*\}^{j\in J}, \\ & ext : \forall(U \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}})U^* \to U^*\ ]\{X := [\![\Gamma \triangleright A]\!]^{\mathrm{TY}}\} \end{array} \tag{3}$$

Applying the substitution $\{X := [\![\Gamma \triangleright A]\!]^{\mathrm{TY}}\}$, *and then Lemma 5.8.1, (3) above is the judgement*

$$\begin{array}{rl} [\![\Gamma]\!] \vdash \mathtt{unfold}([\![\Gamma \triangleright a]\!]) : & [v_i^{sel} : [\![\Gamma \triangleright C_i]\!]^{\mathrm{TY}\ i\in I},\ v_i^{upd} : [\![\Gamma \triangleright C_i]\!]^{\mathrm{TY}} \to X^{\ i\in I}, \\ & m_j^{sel} : [\![\Gamma \triangleright B_j\{X := A\}]\!]^{\mathrm{TY}\ j\in J}, \\ & m_j^{gen} : \forall(U \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}})U^* \to [\![\Gamma, X \triangleright B_j\{X\}]\!]^{\mathrm{TY}}\{X{:=}U^*\}^{\ j\in J}, \\ & ext : \forall(U \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}})U^* \to U^*\ ] \end{array} \tag{4}$$

From the the last judgement,

$$[\![\Gamma]\!] \vdash \mathtt{unfold}([\![\Gamma \triangleright a]\!]).ext : \forall(U \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}})U^* \to U^* \tag{5}$$

Since $[\![\Gamma \triangleright A]\!]^{\mathrm{OP}}$ *is defined, by Lemma 5.13,* $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A]\!]^{\mathrm{OP}} \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}}$. *From* $\Gamma \vdash A^+ \lll\# A$, *derivable in* $\mathtt{Ob}^+$ *by (Match* $\mathtt{pro}$*), by Lemma 5.11,* $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A^+]\!]^{\mathrm{IN}} \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}}$. *From the last to judgements, by transitivity,* $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A^+]\!]^{\mathrm{OP}} \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}}$. *From (5), and the last judgement, by (Val Appl 2), we have that* $[\![\Gamma]\!] \vdash \mathtt{unfold}([\![\Gamma \triangleright a]\!]).ext([\![\Gamma \triangleright A^+]\!]^{\mathrm{OP}}) : ([\![\Gamma \triangleright A^+]\!]^{\mathrm{OP}})^* \to ([\![\Gamma \triangleright A^+]\!]^{\mathrm{OP}})^*$ *is derivable. By Lemma 5.12, the last judgement is*

$$[\![\Gamma]\!] \vdash \mathtt{unfold}([\![\Gamma \triangleright a]\!]).ext([\![\Gamma \triangleright A^+]\!]^{\mathrm{OP}}) : [\![\Gamma \triangleright A^+]\!]^{\mathrm{TY}} \to [\![\Gamma \triangleright A^+]\!]^{\mathrm{TY}}.$$

To conclude, it only remains to show that the arguments of MKOBJ_{A^+} *are well* *typed, that is, that the following judgements are derivable in $F_{\omega<:\mu}$:*

$$[\![\Gamma]\!] \vdash \texttt{unfold}([\![\Gamma \triangleright a]\!]).v_i^{sel} : [\![\Gamma \triangleright C_i]\!]^{\mathrm{TY}} \tag{6}$$

$$[\![\Gamma]\!] \vdash \texttt{unfold}([\![\Gamma \triangleright a]\!]).m_i^{gen} \tag{7}$$
$$: \ \forall(U \leq [\![\Gamma \triangleright A^+]\!]^{\mathrm{IN}})U^* \to [\![\Gamma, U \lessdot\!\# A^+, x : U \triangleright B_j\{U\}]\!]^{\mathrm{TY}}$$

$$[\![\Gamma]\!] \vdash \Lambda(U \leq [\![\Gamma \triangleright A^+]\!]^{\mathrm{IN}})\lambda(x : U^*)\,[\![\Gamma, U \lessdot\!\# A^+, x : U \triangleright b\{U, x\}]\!] \tag{8}$$
$$: \ \forall(U \leq [\![\Gamma \triangleright A^+]\!]^{\mathrm{IN}})U^* \to [\![\Gamma, U \lessdot\!\# A^+, s : U \triangleright B\{U\}]\!]^{\mathrm{TY}}$$

The judgements in (6) derive directly from (4) by (Val Select). That (8) is derivable follows as in the case (Val Object) discussed above. Finally, for the judgements in (7), we reason as follows. From $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A^+]\!]^{\mathrm{IN}} \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}}$ the following judgements derive routinely in $F_{\omega<:\mu}$, for all $j \in J$:

$$[\![\Gamma]\!] \vdash \forall(U \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}})U^* \to [\![\Gamma, U \lessdot\!\# A, x : U \triangleright B_j\{U\}]\!]^{\mathrm{TY}} \tag{9}$$
$$\leq \ \forall(U \leq [\![\Gamma \triangleright A^+]\!]^{\mathrm{IN}})U^* \to [\![\Gamma, U \lessdot\!\# A^+, x : U \triangleright B_j\{U\}]\!]^{\mathrm{TY}}$$

Then, each of the judgements is (7) derives from (4) by (Val Select) and from the corresponding judgement in (9).

(Val Select) *The judgement in question is $\Gamma \vdash a \cdot \ell : B\{A\}$, derived from the two judgements*

$$\Gamma \vdash a : A \tag{1}$$

$$\Gamma \vdash A \lessdot\!\# \texttt{pro}(X)\langle\!\langle \ell : B\{X\}\rangle\!\rangle \tag{2}$$

From (1), by induction hypothesis, $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright a]\!] : [\![\Gamma \triangleright A]\!]^{\mathrm{TY}}$ in $F_{\omega<:\mu}$. From this,

$$[\![\Gamma]\!] \vdash \texttt{unfold}([\![\Gamma \triangleright a]\!]) : [\![\Gamma \triangleright A]\!]^{\mathrm{OP}}([\![\Gamma \triangleright A]\!]^{\mathrm{TY}}) \tag{3}$$

derives by (Val Unfold). From (2), an inspection of the typing rules of $\texttt{Ob}^+$, shows that A is either a pro*-type or a type variable, match-bound in Γ. In both cases, $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A]\!]^{\mathrm{OP}}$ is defined. Then by Lemma 5.13, $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A]\!]^{\mathrm{OP}} \leq [\![\Gamma \triangleright A]\!]^{\mathrm{IN}}$. Again from (2), by Lemma 5.11, we have $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A]\!]^{\mathrm{IN}} \leq [\![\Gamma \triangleright \texttt{pro}(X)\langle\!\langle \ell : B\{X\}\rangle\!\rangle]\!]^{\mathrm{IN}}$. Using the definition of $[\![\Gamma \triangleright \texttt{pro}(X)\langle\!\langle \ell : B\{X\}\rangle\!\rangle]\!]^{\mathrm{IN}}$, it follows $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright \texttt{pro}(X)\langle\!\langle \ell : B\{X\}\rangle\!\rangle]\!]^{\mathrm{IN}} \leq \lambda(X)[\ell^{sel} : [\![\Gamma, X \triangleright B\{X\}]\!]^{\mathrm{TY}}]$. Now, by transitivity, we have $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A]\!]^{\mathrm{OP}} \leq \lambda(X)[\ell^{sel} : [\![\Gamma, X \triangleright B\{X\}]\!]^{\mathrm{TY}}]$. From this, by (Con Sub Appl), (Con Eval Beta), and Lemma 5.8.1:*

$$[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A]\!]^{\mathrm{OP}}([\![\Gamma \triangleright A]\!]^{\mathrm{TY}}) \leq [\ell^{sel} : [\![\Gamma \triangleright B\{A\}]\!]^{\mathrm{TY}}] \tag{4}$$

From (3) and (4), by (Val Subsumption), $[\![\Gamma]\!] \vdash \texttt{unfold}([\![\Gamma \triangleright a]\!]) : [\ell^{sel} : [\![\Gamma \triangleright B\{A\}]\!]^{\mathrm{TY}}]$. Then $[\![\Gamma]\!] \vdash \texttt{unfold}([\![\Gamma \triangleright a]\!]).\ell^{sel} : [\![\Gamma \triangleright B\{A\}]\!]^{\mathrm{TY}}$ derives by (Val Select).

(Val Field Update) *The judgement in question is $\Gamma \vdash a.v \leftarrow_A b$, derived from the judgements $\Gamma \vdash a : A$, $\Gamma \vdash A \lessdot\!\# \texttt{pro}(X)\langle\!\langle v : B\rangle\!\rangle$ and $\Gamma \vdash b : B$. By induction hypothesis, $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright b]\!] : [\![\Gamma \triangleright B]\!]^{\mathrm{TY}}$ and $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright a]\!] : [\![\Gamma \triangleright A]\!]^{\mathrm{TY}}$ are derivable in $F_{\omega<:\mu}$. From the last judgement, by (Val Unfold) $[\![\Gamma]\!] \vdash \texttt{unfold}([\![\Gamma \triangleright a]\!]) : [\![\Gamma \triangleright A]\!]^{\mathrm{OP}}([\![\Gamma \triangleright A]\!]^{\mathrm{TY}})$. Reasoning as in the case (Val Select) above, one shows that $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A]\!]^{\mathrm{OP}} \leq \lambda(X)[v^{upd} : [\![\Gamma \triangleright B]\!]^{\mathrm{TY}} \to X]$*

is derivable. From this, $[\![\Gamma]\!] \vdash [\![\Gamma \triangleright A]\!]^{\text{OP}}([\![\Gamma \triangleright A]\!]^{\text{TY}}) \leq [v^{upd} : [\![\Gamma \triangleright B]\!]^{\text{TY}} \to [\![\Gamma \triangleright A]\!]^{\text{TY}}]$ *and the proof follows as in the previous case by (Val Subsumption) and (Val Select).*

(Val Method Override) *The proof is similar to the case (Val Extend), discussed above.*

□

6. COMPUTATIONAL ADEQUACY

In this section we give a proof that the translation is computationally adequate, that is, closed terms converge in $\mathtt{Ob}^+$ if and only if their encodings converge in $F_{\omega<:\mu}$. At a first look, it is tempting to try and give a direct proof by induction, showing that given any closed term a of $\mathtt{Ob}^+$, and its encoding $[\![\varnothing \triangleright a]\!]$, one has $a \Downarrow_o r$ for some value r if and only if $[\![\varnothing \triangleright a]\!] \Downarrow_f [\![\varnothing \triangleright r]\!]$. Unfortunately, this fails even in the simplest case, when a is in object form. In fact, for any such a, $a \Downarrow_o a$, while $[\![\varnothing \triangleright a]\!]$ does not converge to itself, simply because $[\![\varnothing \triangleright a]\!]$ is not a $F_{\omega<:\mu}$ value: it is a `letrec` definition.

To circumvent the problem, we introduce two new evaluation relations for the source and target calculi. Specifically, we define $\overset{*}{\leadsto}_f$ and $\overset{*}{\leadsto}_o$ to be the transitive and reflexive congruences generated by the reduction relations in $F_{\omega<:\mu}$ and in $\mathtt{Ob}^+$, defined in Figures 6 and 7.

$$a \equiv \varsigma(X{=}A)\langle v_i = c_i{}^{i\in I}, m_j = \varsigma(x:X)b_j\{X,x\}^{j\in J}\rangle$$

$$\begin{array}{llll}
(\mathit{Select}_v) & a.v_i \succ_o c_i & & i \in I \\
(\mathit{Select}_m) & a \circ m_j \succ_o b_j\{A,a\} & & j \in J
\end{array}$$

$$a \equiv \varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J}\rangle$$

$$\begin{array}{ll}
(\mathit{Extend}) & \ell \notin \{\ell_i\}^{i\in I} \\
& a \cdot \ell \longleftarrow\!\!\!+_{(X,A^+)} \mathbf{b}\{X\} \succ_o \varsigma(X{=}A^+)\langle \ell =_{(X)} \mathbf{b}\{X\}, \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J}\rangle \\
(\mathit{Override}) & k \in I \\
& a \cdot \ell_k \leftarrow_{(X,A)} \mathbf{b}\{X\} \succ_o \langle \ell_i =_{(A)} \mathbf{b}_i\{X\}^{i\in I-\{k\}}, \ell_k =_{(A)} \mathbf{b}\{X\}\rangle
\end{array}$$

Fig. 6. Reduction for $\mathtt{Ob}^+$

Then we define an equivalence relation over results in $F_{\omega<:\mu}$, stating that two results r_1 and r_2 are *equivalent*, written $r_1 \approx r_2$, if and only if there exists a result r such that $r_1 \overset{*}{\leadsto}_f r$ and $r_2 \overset{*}{\leadsto}_f r$ (i.e. r_1 and r_2 differ for reductions under lambdas or other constructors). Given that, we are able to prove that $a \Downarrow_o r_o$ if and only if there exists r_f such that $[\![\varnothing \triangleright a]\!] \Downarrow_f r_f$ and $[\![\varnothing \triangleright r_o]\!] \approx r_f$.

`***********************Qui metterei due parole sul perché non usare direttamente la small step semantics: il teorema 6.6 non si semplifica e la prova di 6.5 rimane essenzialmente la stessa`

(β_1)	$(\lambda(x : \mathbb{A})M)\ N$	$\succ_f$	$M\{x := N\}$
(β_2)	$(\Lambda(X \leq \mathbb{A})M)\ \mathbb{B}$	$\succ_f$	$M\{X := \mathbb{B}\}$
(select)	$[\ldots, m_j = M_j, \ldots].m_j$	$\succ_f$	M_j
(unfold)	$\mathtt{unfold}(\mathtt{fold}(\mathbb{A}, M))$	$\succ_f$	M
(letrec)	$\mathtt{letrec}\ f = M\ \mathtt{in}\ N$	$\succ_f$	$N\{f := M\{f := \mathtt{letrec}\ f = M\ \mathtt{in}\ f\}\}$

Fig. 7. Reduction for $F_{\omega<:\mu}$

Lemma 6.1 Evaluation vs Contextual Reduction.

(1) *For every closed $F_{\omega<:\mu}$ term M and result r, $M \Downarrow_f r \Longrightarrow M \rightsquigarrow^*_f r$*
(2) *for every closed $\mathtt{Ob}^+$ term a and result r, $a \Downarrow_o r \Longrightarrow a \rightsquigarrow^*_o r$*

Proof. *Standard.* □

Lemma 6.2 Substitution.

(1) *If $\Gamma, x : A, \Gamma' \vdash b\{x\} : B$ and $\Gamma \vdash a : A$ in $\mathtt{Ob}^+$, then $[\![\Gamma, x : A, \Gamma' \triangleright b]\!]\{x := [\![\Gamma \triangleright a]\!]\} = [\![\Gamma, \Gamma' \triangleright b\{a\}]\!]$*
(2) *If $\Gamma, U \lessdot\!\!\# A, \Gamma' \vdash b\{U\} : B$ and $\Gamma \vdash C \lessdot\!\!\# A$ in $\mathtt{Ob}^+$, then $[\![\Gamma, U \lessdot\!\!\# A, \Gamma' \triangleright b\{U\}]\!]\{U := [\![\Gamma \triangleright C]\!]^{\mathrm{OP}}\} = [\![\Gamma, \Gamma'\{U := C\} \triangleright b\{C\}]\!]$*

Proof. *(1) is by induction on the derivation of $\Gamma, x : A, \Gamma' \vdash b : B$. (2) is by induction on the derivation of $\Gamma, U \lessdot\!\!\# A, \Gamma' \vdash b\{U\} : B$, using Lemma 5.8.2.* □

Lemma 6.3 Properties of Contextual Reduction in $F_{\omega<:\mu}$.

(1) *Let M and N be two terms, and r be a result, all of $F_{\omega<:\mu}$. If $M \rightsquigarrow^*_f N$ and $N \Downarrow_f r$, then there exists a result r' such that $M \Downarrow_f r'$ and $r' \rightsquigarrow^*_f r$.*
(2) *Let M be a term, and r a result, both of $F_{\omega<:\mu}$. If $M \rightsquigarrow^*_f r$ then there exists a result r' s.t. $M \Downarrow_f r'$ and $r' \rightsquigarrow^*_f r$.*
(3) *Let M and N be two terms of $F_{\omega<:\mu}$ such that $M \rightsquigarrow^*_f N$. If $M \Downarrow_f$ then $N \Downarrow_f$.*
(4) *Given a term M, we write $h(M \Downarrow_f r)$ for the height of the derivation of $M \Downarrow_f r$. Let M and M' be two terms, such that $M \rightsquigarrow^*_f M'$. If $M \Downarrow_f r$, and $M' \Downarrow_f r'$, then $h(M' \Downarrow_f r') \leq h(M \Downarrow_f r)$.*

Proof. *(1) is by induction on the number of steps in $M \rightsquigarrow^*_f N$: for the base case, by induction on the derivation of $N \Downarrow_f r$. (2) is an immediate corollary of (1). (3) is proved by induction on the number of steps in $M \rightsquigarrow^*_f N$, and (4) by induction on the height of $M \Downarrow_f r$.* □

Lemma 6.4 Validation of Contextual Reduction.
Let a be an $\mathtt{Ob}^+$ term such that $\Gamma \vdash a : A$ is derivable in $\mathtt{Ob}^+$, then for every b such that $a \rightsquigarrow^*_o b$, one has $[\![\Gamma \triangleright a]\!] \rightsquigarrow^*_f [\![\Gamma \triangleright b]\!]$.

Proof. *We prove the lemma for the reduction $\succ_o$, showing that $a \succ_o b$ implies $[\![\Gamma \triangleright a]\!] \rightsquigarrow^*_f [\![\Gamma \triangleright b]\!]$. The proof for contextual reduction follows by induction on* *the context of reduction and by a subsequent induction on the number of reduction steps. For the reduction $\succ_o$, the only nontrivial case is when the redex is a method selection $a \circ m_k$. In that case, $a \equiv \varsigma(X{=}A)\langle v_i = c_i{}^{i\in I}, m_j = \varsigma(x:X)b_j\{X,x\}^{j\in J}\rangle$, $k \in J$, and the reductum is $b_k\{A,a\}$. By definition,*

$$[\![\,\Gamma \triangleright a\,]\!] \equiv \textsc{mkobj}_A(\,[\![\,\Gamma \triangleright c_i\,]\!]^{\,i\in I},\ \Lambda(U \le \mathbb{A}^{\rm IN})\lambda(x:U^*)\,[\![\,\Gamma, U \lessdot\!\!\# A, x:U \triangleright b_j\{U,x\}\,]\!]^{\,j\in J})$$

Then we have:

$$\begin{array}{ll}
[\![\,\Gamma \triangleright a \circ m_k\,]\!] & \triangleq\ \texttt{unfold}(\,[\![\,\Gamma \triangleright a\,]\!]\,).m_k^{sel} \\
\leadsto^*_f & \big(\Lambda(U \le [\![\,\Gamma \triangleright A\,]\!]^{\rm IN})\lambda(x:U^*)\,[\![\,\Gamma, U \lessdot\!\!\# A, x:U \triangleright b_k\{U,x\}\,]\!]\,\big)\,(\,[\![\,\Gamma \triangleright A\,]\!]^{\rm OP})(\,[\![\,\Gamma \triangleright a\,]\!]\,) \\
\succ_f & \big(\lambda(x:(\,[\![\,\Gamma \triangleright A\,]\!]^{\rm OP})^*)(\,[\![\,\Gamma, U \lessdot\!\!\# A, x:U \triangleright b_k\{U,x\}\,]\!]\{U := [\![\,\Gamma \triangleright A\,]\!]^{\rm OP}\})\big)\ \ (\,[\![\,\Gamma \triangleright a\,]\!]\,) \\
\equiv & \textit{by Lemma 5.12} \\
& \big(\lambda(x:[\![\,\Gamma \triangleright A\,]\!]^{\rm TY})\,[\![\,\Gamma, U \lessdot\!\!\# A, x:U \triangleright b_k\{U,x\}\,]\!]\{U := [\![\,\Gamma \triangleright A\,]\!]^{\rm OP}\}\big)\ \ (\,[\![\,\Gamma \triangleright a\,]\!]\,) \\
\equiv & \textit{by Lemma 6.2.2} \\
& \big(\lambda(x:[\![\,\Gamma \triangleright A\,]\!]^{\rm TY})\,[\![\,\Gamma, x:A \triangleright b_k\{A,x\}\,]\!]\,\big)\ \ (\,[\![\,\Gamma \triangleright a\,]\!]\,) \\
\succ_f & [\![\,\Gamma, x:A \triangleright b_k\{A,x\}\,]\!]\{x := [\![\,\Gamma \triangleright a\,]\!]\} \\
\equiv & \textit{by Lemma 6.2.1} \\
& [\![\,\Gamma \triangleright b_k\{A,a\}\,]\!]
\end{array}$$

□

THEOREM 6.5 COMPUTATIONAL ADEQUACY.
Let a be an $\texttt{Ob}^+$ term such that $\varnothing \vdash a : A$ is derivable in $\texttt{Ob}^+$. Then $a \Downarrow_o$ if and only if $[\![\,\varnothing \triangleright a\,]\!] \Downarrow_f$.

PROOF. *We first prove the "only if" direction: if $a \Downarrow_o$, then $[\![\,\varnothing \triangleright a\,]\!] \Downarrow_f$. Since $a \Downarrow_o$, there exists an $\texttt{Ob}^+$ result r such that $a \Downarrow_o r$. By Lemma 6.1, $a \leadsto^*_o r$, and by Lemma 6.4, $[\![\,\varnothing \triangleright a\,]\!] \leadsto^*_f [\![\,\varnothing \triangleright r\,]\!]$. Since r is a result, it has the form $\varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J}\rangle$; by definition, $[\![\,\varnothing \triangleright r\,]\!] \equiv \textsc{mkobj}_A(...)$, and then $[\![\,\varnothing \triangleright r\,]\!] \leadsto^*_f \texttt{fold}(\mathbb{A}, ...)$, which is a $F_{\omega<:\mu}$ result, say $\bar{r}$. By transitivity, $[\![\,\varnothing \triangleright a\,]\!] \leadsto^*_f \bar{r}$, and, by Lemma 6.3.2, $[\![\,\varnothing \triangleright a\,]\!] \Downarrow_f$ as desired.*

The proof of the opposite direction is by induction on the height of the derivation of $[\![\,\varnothing \triangleright a\,]\!] \Downarrow_f$. The base case is vacuous, as there exists no expression a such that $[\![\,\varnothing \triangleright a\,]\!] \Downarrow_f$ has a derivation of height ≤ 1. For the inductive case, we distinguish several cases, depending on the possible shapes of a.

************`sicuro che non esistono derivazioni di lunghezza 1?? In effetti non abbiamo scritto un assioma` $r \Downarrow r$`, manca???? Allora il caso base diventerebbe il caso (Obj) sotto`

(OBJ) *This case is trivial since every expression a in object form is a result, hence $a \Downarrow_o$.*

(FIELD UPDATE) *By hypothesis, there exists an $F_{\omega<:\mu}$ result, say r_F, such that $[\![\,\varnothing \triangleright a.v_k \leftarrow_A b\,]\!] \Downarrow_f r_F$. By definition, we have that $[\![\,\varnothing \triangleright a.v_k \leftarrow_A b\,]\!] = \texttt{unfold}(\,[\![\,\varnothing \triangleright a\,]\!]\,).v_k^{upd}(\,[\![\,\varnothing \triangleright b\,]\!]\,)$. By an inspection of the evaluation rules, we have that $\texttt{unfold}(\,[\![\,\varnothing \triangleright a\,]\!]\,).v_k^{upd}(\,[\![\,\varnothing \triangleright b\,]\!]\,) \Downarrow_f r_F$ must have been derived from $\texttt{unfold}(\,[\![\,\varnothing \triangleright a\,]\!]\,) \Downarrow_f [...\ v_k^{upd} = M\ ...]$, for some M. $\texttt{unfold}(\,[\![\,\varnothing \triangleright a\,]\!]\,) \Downarrow_f [...\ v_k^{upd} = M\ ...]$, in turn, must have come from $[\![\,\varnothing \triangleright a\,]\!] \Downarrow_f \texttt{fold}(\mathbb{A}, N)$ for* *some N such that $N \Downarrow_f [\ldots\ v_k^{upd} = M\ \ldots]$. From $[\![\varnothing \rhd a]\!] \Downarrow_f \texttt{fold}(\mathbb{A}, N)$, by induction hypothesis there exists a result r_o such that $a \Downarrow_o r_o$. Since $\varnothing \vdash a.v_k \leftarrow_A b$ is derivable, by Theorem 2.2, r_o is in object form and contains a field labeled v_k. Hence, $a.v_k \leftarrow_A b \Downarrow_o$ by (Override).*

(METHOD OVERRIDE) *By hypothesis, $[\![\varnothing \rhd a \circ m \leftarrow \varsigma(X{=}A)\varsigma(x:X)b]\!] \Downarrow_f r_F$ for some result r_F. By definition, we have that $[\![\varnothing \rhd a \circ m \leftarrow \varsigma(X{=}A)\varsigma(x:X)b]\!] = \texttt{unfold}([\![\varnothing \rhd a]\!]).ext([\![\varnothing \rhd A]\!]^{\text{OP}})(\text{MKOBJ}_A(\ldots))$. $\texttt{unfold}([\![\varnothing \rhd a]\!]).ext([\![\varnothing \rhd A]\!]^{\text{OP}})(\text{MKOBJ}_A(\ldots)) \Downarrow_f r_F$ must have come from $[\![\varnothing \rhd a]\!] \Downarrow_f r$, for some r. By induction hypothesis, $a \Downarrow_o r_o$ for some result r_o. Then, by (Override), one derives $a \circ m \leftarrow \varsigma(X{=}A)\varsigma(x:X)b \Downarrow_o$. As in the previous case, Theorem 2.2 ensures that (Override) is applicable.*

(EXTEND) *By hypothesis one has $[\![\varnothing \rhd a \cdot \ell \longleftarrow\!\!+_{(X,A)} \mathbf{b}\{X\}]\!] \Downarrow_f r_F$ for some result r_F. By definition, $[\![\varnothing \rhd a \cdot \ell \longleftarrow\!\!+_{(X,A)} \mathbf{b}\{X\}]\!] = \texttt{unfold}([\![\varnothing \rhd a]\!]).ext(\ldots)$. Now, $\texttt{unfold}([\![\varnothing \rhd a]\!]).ext(\ldots) \Downarrow_f r_F$ must have come from $[\![\varnothing \rhd a]\!] \Downarrow_f r$, for some result r. By induction hypothesis, $a \Downarrow_o r_o$ for some result r_o. Then $a \cdot \ell \longleftarrow\!\!+_{(X,A)} \mathbf{b}\{X\} \Downarrow_o$ by (Extend): again, Theorem 2.2 ensures that (Extend) is applicable.*

(SELECT) *We first handle the two cases of field and method selection uniformly: we will distinguish them later in the proof. By hypothesis, there exists a result r_F such that $[\![\varnothing \rhd a \cdot \ell_k]\!] \Downarrow_f r_F$. By definition, $[\![\varnothing \rhd a \cdot \ell_k]\!] = \texttt{unfold}([\![\varnothing \rhd a]\!]).\ell_k^{sel}$. Now, $\texttt{unfold}([\![\varnothing \rhd a]\!]).\ell_k^{sel} \Downarrow_f r_F$ must have come from $\texttt{unfold}([\![\varnothing \rhd a]\!]) \Downarrow_f [\ldots\ \ell_k^{sel} = M_k \ldots]$ for some M_k such that $M_k \Downarrow_f r_F$. Similarly, $\texttt{unfold}([\![\varnothing \rhd a]\!]) \Downarrow_f [\ldots\ \ell_k^{sel} = M_k \ldots]$ must have come from $[\![\varnothing \rhd a]\!] \Downarrow_f \texttt{fold}(\mathbb{A}, M)$, for some M such that $M \Downarrow_f [\ldots\ \ell_k^{sel} = M_k \ldots]$. From $[\![\varnothing \rhd a]\!] \Downarrow_f \texttt{fold}(\mathbb{A}, M)$, by induction hypothesis $a \Downarrow_o r_o$, for some result r_o. By Theorem 2.2, we know that r_o is in object form: more precisely, $r_o = \varsigma(X{=}A)\langle v_i = c_i{}^{i \in I}, m_j = \varsigma(x:X)b_j\{X,x\}^{j \in J}\rangle$. Since $a \Downarrow_o r_o$, by Lemma 6.1, $a \overset{*}{\rightsquigarrow}_o r_o$ and, by Lemma 6.4 $[\![\varnothing \rhd a]\!] \overset{*}{\rightsquigarrow}_f [\![\varnothing \rhd r_o]\!]$. Given the shape of r_o, by definition,*

$$[\![\varnothing \rhd r_o]\!] = \\ \text{MKOBJ}_A([\![\varnothing \rhd c_i]\!]^{i \in I}, \Lambda(U \le [\![\varnothing \rhd A]\!]^{\text{IN}})\lambda(x:U^*)[\![U \lll\!\!\!\# A, x:U \rhd b_j\{U,x\}]\!]^{j \in J})$$

Clearly, we have that $[\![\varnothing \rhd r_o]\!] \overset{*}{\rightsquigarrow}_f \texttt{fold}([\![\varnothing \rhd A]\!]^{\text{TY}}, [\ldots,\ \ell_k^{sel} = M'_k, \ldots])$. This, together with $[\![\varnothing \rhd a]\!] \overset{*}{\rightsquigarrow}_f [\![\varnothing \rhd r_o]\!]$ implies, by transitivity, $[\![\varnothing \rhd a]\!] \overset{*}{\rightsquigarrow}_f \texttt{fold}([\![\varnothing \rhd A]\!]^{\text{TY}}, [\ldots,\ \ell_k^{sel} = M'_k\ \ldots])$, By Lemma 6.3.2, there exists a result r such that $[\![\varnothing \rhd a]\!] \Downarrow_f r$ and $r \overset{*}{\rightsquigarrow}_f \texttt{fold}(\mathbb{A}, [\ldots,\ \ell_k^{sel} = M'_k, \ldots])$. But we derived $[\![\varnothing \rhd a]\!] \Downarrow_f \texttt{fold}(\mathbb{A}, M)$ from the hypothesis: then, since $\Downarrow_f$ is deterministic, $r \equiv \texttt{fold}(\mathbb{A}, M)$, and therefore $\texttt{fold}(\mathbb{A}, M) \overset{*}{\rightsquigarrow}_f \texttt{fold}(\mathbb{A}, [\ldots,\ \ell_k^{sel} = M'_k\ \ldots])$. This implies $M \overset{*}{\rightsquigarrow}_f [\ldots,\ \ell_k^{sel} = M'_k\ \ldots]$. From $M \overset{*}{\rightsquigarrow}_f [\ldots,\ \ell_k^{sel} = M'_k\ \ldots]$, and from $M \Downarrow_f [\ldots\ \ell_k^{sel} = M_k \ldots]$ (which follows again from the hypothesis), by Lemma 6.3.2 and the reasoning we have just described, $[\ldots\ \ell_k^{sel} = M_k \ldots] \overset{*}{\rightsquigarrow}_f [\ldots,\ \ell_k^{sel} = M'_k\ \ldots]$. From this, $M_k \overset{*}{\rightsquigarrow}_f M'_k$. Furthermore, since $M_k \Downarrow_f r_F$ by hypothesis, by Lemma 6.3.3, we know that $M'_k \Downarrow_f r'$ for some result r'.

Summarizing: $M_k \overset{*}{\rightsquigarrow}_f M'_k$, $M_k \Downarrow_f r_F$ and $M'_k \Downarrow_f r'$, with r_F and r' two results in $F_{\omega<:\mu}$. Now we distinguish the two cases of field and method selection.

(Field selection) *In this case* $M'_k \equiv [\![\varnothing \triangleright c_k]\!]$, *for some* c_k, *so* $[\![\varnothing \triangleright c_k]\!] \Downarrow_f r'$. *By Lemma 6.3.4,* $h(M_k \Downarrow_f r') \leq h(M_k \Downarrow_f r_F)$. *Furthermore, since* $h(M_k \Downarrow_f r_F) \leq h([\![\varnothing \triangleright a.\ell_k]\!] \Downarrow_f r_F)$ *(by Lemma 6.3.4, as we showed that* $[\![\varnothing \triangleright a.\ell_k]\!] \Downarrow_f r_F$ *depends on* $M_k \Downarrow_f r_F$*), by transitivity* $h(M_k \Downarrow_f r') \leq h([\![\varnothing \triangleright a.\ell_k]\!] \Downarrow_f r_F)$. *Then, by induction hypothesis, it follows that there exists* $\widehat{r}$ *such that* $c_k \Downarrow_o \widehat{r}$. *Now, from* $a \Downarrow_o r_o$, *and from* $c_k \Downarrow_o \widehat{r}$, *by (Select$_v$) one has* $a.\ell_k \Downarrow_o \widehat{r}$ *as desired.*

(Method selection) *In this case*
$M'_k \equiv (\Lambda(U \leq [\![\varnothing \triangleright A]\!]^{\text{IN}})\lambda(x : U^*)[\![U \Lleftarrow\!\!\!\!\# A, x : U \triangleright b_k\{U,x\}]\!])([\![\varnothing \triangleright A]\!]^{\text{OP}})([\![\varnothing \triangleright r_o]\!])$.
and $M'_k \Downarrow_f r'$ *depends on the following derivations:*

- $(\Lambda(U \leq [\![\varnothing \triangleright A]\!]^{\text{IN}})\lambda(x : U^*)[\![U \Lleftarrow\!\!\!\!\# A, x : U \triangleright b_k\{U,x\}]\!])([\![\varnothing \triangleright A]\!]^{\text{OP}}) \Downarrow_f \lambda(s : \mathbb{C})N$ *for some type* $\mathbb{C}$ *and expression* N *such that* $N\{x := [\![\varnothing \triangleright r_o]\!]\} \Downarrow_f r'$;
- $\Lambda(U \leq [\![\varnothing \triangleright A]\!]^{\text{IN}})\lambda(x : U^*)[\![U \Lleftarrow\!\!\!\!\# A, x : U \triangleright b_k\{U,x\}]\!] \Downarrow_f \Lambda(U \leq [\![\varnothing \triangleright A]\!]^{\text{IN}})N'$ *for some expression* N' *such that* $N'\{U := [\![\varnothing \triangleright A]\!]^{\text{OP}}\} \Downarrow_f \lambda(x : \mathbb{C})N$;

Looking at the shapes of those derivations, we can give the exact description of the expressions N *and* N', *namely:* $N' \equiv \lambda(x : U^*)[\![U \Lleftarrow\!\!\!\!\# A, x : U \triangleright b_k\{U,x\}]\!]$, *from which* $N \equiv [\![x : A \triangleright b_k\{A,x\}]\!]$ *by Lemma 6.2.2. From the hypothesis* $N\{x := [\![\varnothing \triangleright r_o]\!]\} \Downarrow_f r'$, *by Lemma 6.2.1,* $[\![\varnothing \triangleright b_k\{A,r_o\}]\!] \Downarrow_f r'$.

Summarizing, $M_k \overset{*}{\leadsto}_f M'_k \overset{*}{\leadsto}_f [\![\varnothing \triangleright b_k\{A,r_o\}]\!]$. *Then, by Lemma 6.3.4,* $h(b_k\{A,r_o\} \Downarrow_f r') \leq h(M_k \Downarrow_f v_F)$. *Furthermore, since* $h(M_k \Downarrow_f r_F) \leq h([\![\varnothing \triangleright a \circ \ell_k]\!] \Downarrow_f r_F)$, *by the inductive hypothesis, it follows that there exists* $\widehat{r}$ *such that* $b_k\{A,r_o\} \Downarrow_o \widehat{r}$. *Now, from* $a \Downarrow_o r_o$, *and from* $b_k\{A,r_o\} \Downarrow_o \widehat{r}$, *by (Select$_m$) one has* $a.\ell_k \Downarrow_o \widehat{r}$ *as desired.*

□

Theorem 6.6. *Let* a *be an* $\texttt{Ob}^+$ *term such that* $\varnothing \vdash a : A$ *is derivable in* $\texttt{Ob}^+$. *Then,* $a \Downarrow_o r_o$ *for a given* $\texttt{Ob}^+$ *result* r_o *if and only if there exists an* $F_{\omega<:\mu}$ *result* r_F *such that* $[\![\varnothing \triangleright a]\!] \Downarrow_f r_F$ *and* $[\![\varnothing \triangleright r_o]\!] \approx r_F$.

Proof. *By theorem 6.5 we know that* $a \Downarrow_o r_o$ *if and only if* $[\![\varnothing \triangleright a]\!] \Downarrow_f r_F$ *for two given results* r_o *and* r_F. *We only need to show that* $[\![\varnothing \triangleright r_o]\!] \approx r_F$ *, i.e. that there exists* r *such that* $[\![\varnothing \triangleright r_o]\!] \overset{*}{\leadsto}_f r$ *and* $r_F \overset{*}{\leadsto}_f r$.

If $a \Downarrow_o r_o$ *then we have that* r_o *is an expression in object form, i.e.* $r_o \equiv \varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i \in I \cup J}\rangle$. *By definition,* $[\![\varnothing \triangleright r_o]\!] = \text{MKOBJ}_A(\ldots)$ *and hence* $[\![\varnothing \triangleright r_o]\!] \overset{*}{\leadsto}_f r = \texttt{fold}(\mathbb{A}, \ldots)$. *From the hypothesis* $a \Downarrow_o r_o$ *it also follows, by Lemma 6.1, that* $a \overset{*}{\leadsto}_o r_o$*: then, by Lemma 6.4,* $[\![\varnothing \triangleright a]\!] \overset{*}{\leadsto}_f [\![\varnothing \triangleright r_o]\!]$, *and hence* $[\![\varnothing \triangleright a]\!] \overset{*}{\leadsto}_f r$ *by transitivity. Now, by Lemma 6.3.2, there exists a result* r' *such that* $[\![\varnothing \triangleright a]\!] \Downarrow_f r'$ *and* $r' \overset{*}{\leadsto}_f r$. *Since* $\Downarrow_f$ *is deterministic, this implies* $r' \equiv r_F$ *and hence* $r_F \approx [\![\varnothing \triangleright r_o]\!]$ *as desired.*

7. SUBTYPING AND NONEXTENSIBLE OBJECTS

Combining object extension with subtyping has long been known to be a difficult problem. The essence of the problem is easily explained: in an object with two methods m_1 and m_2 of types B_1 and B_2, the typing of m_1 may require m_2 to have type B_2. Forgetting m_2 by *width* subtyping would result in the possible re-addition of m_2 with a type B'_2 incompatible with B_2. A related problem arises from the combination of *depth* subtyping with overriding. Suppose, in the example above, that we promote the type of m_2 to a new type B'_2, super-type of B_2: then, it is possible to override m_2 with a new method of proper type B'_2, thus violating the assumption used in the typing of m_1.

Solutions to these problems may be found in the current literature that follow two orthogonal approaches: either allow a limited form of subtyping in the presence of object extension, or distinguish extensible from nonextensible objects and disallow subtyping on the former while allowing it on the latter. Both approaches have their own merits, but the type systems following the former do not lend themselves to direct interpretations, as they rely on somewhat ad-hoc notions of object types (see, for example [Bono et al. 1999; Rémy 1998]). For this reason, in the rest of this section we focus our attention on the second approach. Besides having a number of interesting conceptual consequences [Fisher and Mitchell 1998], this approach is interesting for its generality: as we discuss next, it allows different subtype relations to be formalized uniformly within the same framework.

7.1 From extensible to nonextensible objects, via subtyping

The idea of distinguishing extensible from nonextensible objects originated with the work of Fisher and Mitchell in [Fisher and Mitchell 1995]. This distinction may be accounted for in our source calculus by extending the type system of $\mathtt{Ob}^{+}$ with new types, contexts, and judgements: the typing rules for the extended system, named $\mathtt{Ob}^{+}_{<:}$, are collected in Appendix C.

The new types are of the form $\mathtt{obj}(X)\langle\!\langle \ell_i{:}B_i\{X\}^{i\in 1..n}\rangle\!\rangle$, and their reading is similar to that of the pro-types of Section 2, with two important differences: (i) like pro-typed objects, elements of obj-types may be sent messages, or modify their own structure from the inside, via self-inflicted updates; (ii) unlike pro-typed objects, methods and fields of obj-typed object may *not* be modified or added from the outside.

The typing of field and method selection for obj-typed objects is governed by (Val Select $\mathtt{Obj}$) rule, whose reading is essentially the same as the corresponding rule for pro-types.

The subtype relation, denoted by $<:$, obeys the standard laws of reflexivity and transitivity, and it is used in conjunction with a rule of subsumption, also standard. Subtyping judgements have the form $\Gamma \vdash A <: B$ where A and B are pro-types, obj-types or type variables. Contexts of the extended type system may include subtype constraints over type variables. The translation we defined in the previous sections extends uniformly to the new contexts and judgements: the new clauses are given in Figure 8.

The subtype relation orders pro and obj- types, so that pro-types can be may be promoted, by subtyping, to their corresponding obj-types. While only reflexive subtyping is defined over pro-types, different rules may be introduced to define subtyping over obj-types. The translation of obj-types depends on the chosen notion of subtyping: we illustrate two cases – the obj-types of [Fisher and Mitchell 1995], and the covariant *Self Types* of [Abadi and Cardelli 1996b] – and we show that in both cases the interpretation validates the expected subtyping relationships.

Type Variables
$$[\![\Gamma,' X <: A, \Gamma'' \triangleright X]\!]^{\mathrm{TY}} \triangleq X$$

Judgements
$$[\![\Gamma \vdash A <: B]\!] \triangleq [\![\Gamma]\!] \vdash [\![\Gamma \triangleright A]\!]^{\mathrm{TY}} <: [\![\Gamma \triangleright B]\!]^{\mathrm{TY}}$$

Contexts
$$[\![\Gamma, X <: A]\!] \triangleq [\![\Gamma]\!], X <: [\![\Gamma \triangleright A]\!]^{\mathrm{TY}}$$
$$[\![\Gamma \triangleright \Gamma', X <: A]\!] \triangleq [\![\Gamma \triangleright \Gamma']\!], X <: [\![\Gamma, \Gamma' \triangleright A]\!]^{\mathrm{TY}}$$

Fig. 8. Translation of Contexts and Judgements in $\mathtt{Ob}^{+}_{<:}$.

7.2 Covariant Subtyping à la *Fisher-Mitchell'95*

The subtype relation is defined by the following rule from [Fisher and Mitchell 1995], where `probj` stands for `pro` or `obj`:

$$\frac{\Gamma, Y, X <: Y \vdash B_i\{X\} <: B'_i\{Y\} \quad \forall i \in 1..n}{\Gamma \vdash \mathtt{probj}(X)\langle\!\langle \ell_i{:}B_i\{X\}^{i\in 1..n+k} \rangle\!\rangle <: \mathtt{obj}(Y)\langle\!\langle \ell_i{:}B'_i\{Y\}^{i\in 1..n} \rangle\!\rangle} \;.$$

According to this rule, `pro`-types may only be promoted to `obj`-types, not to other `pro`-types: as a consequence, only reflexive subtyping is available for `pro`-type, as it is required for the soundness of object extension and override. For `obj`-types, instead, the rule allows subtyping both in *width* and *depth*: since objects with `obj`-types may not be extended or updated from the outside, this form of subtyping is sound.

A translation for `obj`-types is given in Figure 9.

$$[\![\Gamma \triangleright \mathtt{obj}(X)\langle\!\langle \ell_i : B_i\{X\}^{i\in 1..n} \rangle\!\rangle]\!]^{\mathrm{TY}} \triangleq \mu(X)[\ell_i{}^{sel} : [\![\Gamma, X \triangleright B_i\{X\}]\!]^{\mathrm{TY}\, i\in 1..n}]$$

Fig. 9. Translation for `obj` types

`obj`-types are interpreted directly as recursive record-types: for each of the field and method labels occurring in the `obj`-type, the interpretation lists the corresponding selectors, hiding method generators and field updaters. Consequently, as in the source calculus, any attempt to modify the structure of an `obj`-typed object from the outside is ill-typed in the interpretation. Also, given the encoding of `pro` and `obj`-types, it is easy to verify that the subtype relation

$$[\![\Gamma \triangleright \mathtt{pro}(X)\langle\!\langle \ell_i : B_i\{X\}^{i\in 1..n+k} \rangle\!\rangle]\!]^{\mathrm{TY}} <: [\![\Gamma \triangleright \mathtt{obj}(X)\langle\!\langle \ell_i : B_i\{X\}^{i\in 1..n} \rangle\!\rangle]\!]^{\mathrm{TY}}$$

is validated by the subtying rules for recursive and record types of $F_{\omega<:\mu}$. The properties of the translation are stated in the following theorem:

Theorem 7.1 Validation of *Fisher-Mitchell* Subtyping.

(*1*) *If* $\Gamma \vdash A <: B$ *in* $\texttt{Ob}^{+}_{<:}$, *then* $[\![\,\Gamma\,]\!] \vdash [\![\,\Gamma \triangleright A\,]\!]^{\text{TY}} <: [\![\,\Gamma \triangleright B\,]\!]^{\text{TY}}$ *in* $F_{\omega<:\mu}$.
(*2*) *If* $\Gamma \vdash a : A$ *in* $\texttt{Ob}^{+}_{<:}$, *then* $[\![\,\Gamma \triangleright a : A\,]\!] \vdash [\![\,\Gamma \triangleright A\,]\!]$ *in* $F_{\omega<:\mu}$.

PROOF. *(1) follows by induction on the derivation of* $\Gamma \vdash A <: B$. *(2) follows also by induction, using (1) to handle the subtyping judgements in the derivation of* $\Gamma \vdash a : A$.

7.3 Invariant Subtyping for Covariant Self Types à la *Abadi-Cardelli'96*

Covariant *Self Types*, denoted here by the type expression $\texttt{obj}_{\texttt{AC}}(X)\langle\!\langle \ell_i : B_i\{X\}^{i\in 1..n}\rangle\!\rangle$ are described by Abadi and Cardelli in their book [Abadi and Cardelli 1996b] (cf. Chaps. 15, 16). These types share several features with the `obj` types of [Fisher and Mitchell 1995], notably the fact that both describe collections of nonextensible objects. However, they have important specificities, that we summarize as follows: (i) updates are legal on elements of $\texttt{obj}_{\texttt{AC}}$ types, and (ii) subtyping over $\texttt{obj}_{\texttt{AC}}$ types is only allowed in *width*, and defined by the following rule:

$$\frac{\Gamma, X \vdash B_i\{X^+\} \quad \forall i \in 1..n+k}{\Gamma \vdash \texttt{probj}_{\texttt{AC}}(X)\langle\!\langle \ell_i : B_i\{X\}^{i\in 1..n+k}\rangle\!\rangle <: \texttt{obj}_{\texttt{AC}}(X)\langle\!\langle \ell_i : B_i\{X\}^{i\in 1..n}\rangle\!\rangle}$$

The notation $B\{X^+\}$ and the covariance condition expressed by the judgements $\Gamma, X \vdash B_i\{X^+\}$, both defined in [Abadi and Cardelli 1996b], imply that X must not occur within an $\texttt{obj}_{\texttt{AC}}$ type within any of the B_i's. A translation that validates the subtyping relation defined by the rule above is given in Figure 10. Note that

$$A \equiv \texttt{obj}_{\texttt{AC}}(X)\langle\!\langle v_i : C_i{}^{i\in 1..n}, m_j : B_j\{X\}^{j\in 1..m}\rangle\!\rangle$$
$$[\![\,\Gamma \triangleright A\,]\!]^{\text{TY}} \triangleq \mu(X)[v_i{}^{upd} : [\![\,\Gamma \triangleright C_i\,]\!]^{\text{TY}} \to X,\ v_i{}^{sel} : [\![\,\Gamma \triangleright C_i\,]\!]^{\text{TY}\ i\in 1..n},$$
$$m_j{}^{sel} : [\![\,\Gamma, X \triangleright B_j\{X\}\,]\!]^{\text{TY}\ j\in 1..m}]$$

Fig. 10. Translation of $\texttt{obj}_{\texttt{AC}}$ types

the field updaters are exposed by the translation, thus making the translation of field updates well typed, as these operations are allowed on Abadi-Cardelli objects. Each of the components C_i's is invariant in the translated type, as a result of a contravariant occurrence, in the updater's type, and of a covariant occurrence in the selector's type. From this observation, the proof that the translation validates the expected subtypings follows routinely using the covariance condition on the occurrences of X.

THEOREM 7.2 VALIDATION OF *Abadi-Cardelli* SUBTYPING.

(*1*) *If* $\Gamma \vdash A <: B$ *in* $\texttt{Ob}^{+}_{<:}$, *then* $[\![\,\Gamma\,]\!] \vdash [\![\,\Gamma \triangleright A\,]\!]^{\text{TY}} <: [\![\,\Gamma \triangleright B\,]\!]^{\text{TY}}$ *in* $F_{\omega<:\mu}$.
(*2*) *If* $\Gamma \vdash a : A$ *in* $\texttt{Ob}^{+}_{<:}$, *then* $[\![\,\Gamma \triangleright a : A\,]\!] \vdash [\![\,\Gamma \triangleright A\,]\!]$ *in* $F_{\omega<:\mu}$.

Remarks. A final remark is in order, to complete the comparisons with the type systems of [Fisher and Mitchell 1995] and [Abadi and Cardelli 1996b]. There is a difference between nonextensible objects in our $\texttt{Ob}^{+}_{<:}$ and the corresponding notions in those systems, as we disallow self-inflicted method overrides that are instead legal in [Fisher and Mitchell 1995] and [Abadi and Cardelli 1996b]. The results of Theorems 7.1 and 7.2 do hold even though we lift our restriction, and extend the translation to include method updaters. Again, however, the soundness of these theorems is conditioned to the soundness of the extended type theory needed to interpret self-inflicted method overrides via method updaters.

8. RELATED WORK

Several papers on object encodings have recently been published in literature.

Encodings of nonextensible objects. In [Abadi et al. 1996], Abadi, Cardelli and Viswanathan describe and encoding based on the "split-method" technique that inspired our encoding. While the two interpretations share this initial idea, they are different in several respects. In [Abadi et al. 1996] (nonextensible) object types are interpreted by means of a clever combination of recursive and existential types that validates the invariant subtyping relationship of their source calculus. The use of existential types has a price however, as it complicates the interpretation of message sends: a *self* field, filled with a recursive pointer to the object representation, must be maintained in the object, to validate the typed interpretation of method invocation by self-application. As discussed in Section 7, our translation specializes to nonextensible objects with essentially equivalent results. The absence of abstraction primitives in our setting, due to the absence of existential quantifiers, simplifies our treatment of method and field selection, and makes our translation applicable to other notions of objects, notably, to the nonextensible objects of [Fisher and Mitchell 1995], whose semantics is rendered with exact precision.

In [Abadi and Cardelli 1996b], Abadi and Cardelli study several variants of the split-method encoding of [Abadi et al. 1996]. The closest to ours is the first variant, that uses recursive types and no existentials (see Chapter 18.3.5). We extend that interpretation to capture object extension and *MyType* specialization.

In [Viswanathan 1998], Viswanathan improves the encoding of [Abadi et al. 1996] with a fully abstract interpretation that uses a first-order target theory.

In [Bruce et al. 1999], Bruce, Cardelli and Pierce give an insightful analysis of object encodings, where several models of objects are compared within the same target theory, an equivalent of $F_{\omega<:\mu}$. The recursive record model of [Cardelli 1988; Cook 1989; Reddy 1988], the encodings of [Bruce 1994] based on recursive and existential types, of [Pierce and Turner 1994] (existential encoding), and the bounded existentials of [Abadi et al. 1996] are all evaluated and compared on the basis of several parameters: responsibility for packaging results, internal access to "self methods", target type theory, quantifiers and subtyping, support for covariant interface, and method update.

In [Crary 1999], K. Crary shows that a simple encoding of the object types studied in [Abadi et al. 1996] may be given using a combination of existential and intersection types. In his paper, Crary studies the cases of nonextensible objects, showing that his translation avoids the need to split methods in several components. It would be interesting to investigate whether his proposal can be adapted to the translation of extensible objects we have presented.

***Boudol and Dal Zilio's* encoding of extensible objects.** In [Boudol and Dal Zilio 1999], Boudol and Dal Zilio study an encoding for extensible objects that relies on essentially the same idea used in our interpretation. They also represent an extensible object as a pair of a generator and a non extensible object. Technically, however, the two interpretations are fundamentally different. Firstly, the source calculus of [Boudol and Dal Zilio 1999] is the original *Lambda Calculus of Objects* of [Fisher et al. 1994], with untyped terms and λ-abstractions. Secondly, the target theory used in their translation is a variant of $F_{\omega<:\mu}$ where (i) recursive types are assumed to be equivalent (rather than isomorphic) to their unfoldings (ii), records are *extensible*, and consequently, (iii) subtyping over record types is non-standard. Based on these assumptions, their translation models object generators in ways similar to [Cook 1989]. Interestingly, the target theory requires fixed points at type operators to handle self-inflicted method updates, just as in our translation.

Other type systems for object extension and subtyping. In [Riecke and Stone 1998], Riecke and Stone take a different approach to combining object extension with subtyping. Their type system allows object extension even in the presence of subtyping: the system is sound as, operationally, when a method m happens to be re-added to an object, it is treated as a complete new component. This is accomplished by generating a fresh name for the new method, and storing it in a *dictionary* which is held as part of the object representation. The name m then refers to the newly added method body, whereas the newly generated name refers to the body of the old m method, and it is not available for external references.

A different approach to the problem is to allow object extension to coexist with limited forms of subtyping. This idea has been pursued in several papers: the type systems proposed by Abadi [Abadi 1994], Bono *et al.* [Bono et al. 1999], Rémy [Rémy 1998], and [Liquori 1997] may all be ascribed to this approach. To account for the combination of object extension and subtyping, these proposals devise different mechanisms that allow subtyping to be traced: the intention is to either isolate safe uses of subtyping in the presence of primitives of object extension, or reconstruct the "true type" of an object and use it to prevent unsound extensions. Unfortunately, these tracing mechanisms are rather difficult to interpret and explain within translations like the one we have described.

9. CONCLUSIONS

We have presented (sub)type preserving translation of an object calculus that supports method invocation, functional method override, object extension, and subtyping over nonextensible objects. The complexity of the target theory varies with the form of method override assumed in the source calculus. An equivalent of system $F^{\omega}_{<:}$ with recursive types suffices to model field updates and a restricted form of method override. Self-inflicted method overrides, instead, require more generality.

Beside using standard techniques, the translation we presented is formally intricate. For instance, the mismatch between the call-by-value semantics of $\mathtt{Ob}^{+}$ and call-by-name semantics of $F_{\omega<:\mu}$ requires the presence of an additional filed (ext) in the record that encodes an object. Choosing a call-by-value semantics also for

$F_{\omega<:\mu}$ wolud have allowed a simpler, still computationally adequate, encoding with no extra field. Moreover, using a call-by-value semantics for the target language wuold make the encoding practical: it could be used as such when programming a language that is equipped with a type system at least as powerful as $F_{\omega<:\mu}$.

Besides providing a fully formal interpretation of extensible objects, the interpretation we presented also clarifies the relationship between calculi of extensible and nonextensible objects appearing in the recent literature. We showed in fact, that the encoding specializes smoothly to the case of nonextensible objects and object types, validating the expected subtyping relationships. Although we focused on one particular calculus – specifically, on one approach to combining object extension with subtyping – the translation is shown to be sufficiently general to capture other notions of subtyping over object types (a notable example are the rules for covariant subtyping of [Abadi and Cardelli 1996b]).

REFERENCES

ABADI, M. 1994. Baby Modula–3 and a Theory of Objects. *Journal of Functional Programming 4,* 2, 249–283.

ABADI, M. AND CARDELLI, L. 1996a. On Subtyping and Matching. *ACM Transactions on Programming Languages and Systems 11,* 4, 401–423.

ABADI, M. AND CARDELLI, L. 1996b. *A Theory of Objects.* Monographs in Computer Science. Springer.

ABADI, M., CARDELLI, L., AND VISWANATHAN, R. 1996. An Interpretation of Objects and Object Types. In *Proceedings of POPL'96.* 396–409.

BONO, V. AND BUGLIESI, M. 1999a. Interpretations of extensible objects and types. In *Proceedings of FCT'99.* LNCS, vol. 1684. Springer–Verlag, 112–124.

BONO, V. AND BUGLIESI, M. 1999b. Matching for the Lambda Calculus of Objects. *Theoretical Computer Science 212,* 1/2 (Feb.), 101–140.

BONO, V., BUGLIESI, M., DEZANI, M., AND LIQUORI, L. 1999. Subtyping Constraints for Incomplete Objects. *Fundamenta Informaticae 38,* 4 (June), 325–364.

BONO, V. AND LIQUORI, L. 1995. A Subtyping for the Fisher-Honsell-Mitchell Lambda Calculus of Objects. In *Proceedings of CSL'95.* Lecture Notes in Computer Science, vol. 933. Springer-Verlag, 16–30.

BOUDOL, G. AND DAL ZILIO, S. 1999. An interpretation of extensible objects. In *Proceedings of FCT'99.* LNCS, vol. 1684. Springer–Verlag, 148–160.

BRUCE, K. 1994. A Paradigmatic Object-Oriented Programming Language: Design, Static Typing and Semantcs. *Journal of Functional Programming 1,* 4, 127–206.

BRUCE, K., CARDELLI, L., AND PIERCE, B. 1999. Comparing Object Encodings. *Information and Computation 155,* 1/2, 108–133.

CARDELLI, L. 1988. A Semantics of Multiple Inheritance. *Information and Computation 76,* 138–164.

COOK, W. 1987. A Self-ish Model of Inheritance. Manuscript.

COOK, W. 1989. A Denotational Semantics of Inheritance. Ph.D. thesis, Brown University.

CRARY, K. 1999. Simple, efficient object encoding using intersection types. Tech. rep., CMU-CS-99-100, Cornell University. January.

FISHER, K., HONSELL, F., AND MITCHELL, J. C. 1994. A Lambda Calculus of Objects and Method Specialization. *Nordic Journal of Computing 1,* 1, 3–37.

FISHER, K. AND MITCHELL, J. C. 1995. A Delegation-based Object Calculus with Subtyping. In *Proceedings of FCT'95.* Lecture Notes in Computer Science, vol. 965. Springer-Verlag, 42–61.

FISHER, K. AND MITCHELL, J. C. 1998. On the Relationship between Classes, Objects, and Data Abstraction. *Theory and Practice of Object Systems 4,* 1, 3–25.

KAMIN, S. 1988. Inheritance in Smalltalk-80: a denotational definition. In *Proceedings of POPL'88*. ACM Press, 80–87.

LIQUORI, L. 1997. An Extended Theory of Primitive Objects: First Oder System. In *Proc. of ECOOP'97*. Lecture Notes in Computer Science, vol. 1241. Springer-Verlag, 146–169.

PIERCE, B. AND TURNER, D. 1994. Simple type-theoretic foundations for object-oriented programming. *Journal of Functional Programming 4,* 2, 207–248.

REDDY, U. 1988. Objects as Closures: Abstarct Semantics of Object Oriented Languages. In *Proceedings of the ACM Int. Conf. on Lisp and Functional Programming*. 289–297.

RÉMY, D. 1998. From classes to objects via subtyping. In *Proceedings of ESOP'98*. Lecture Notes in Computer Science, vol. 1381. Springer-Verlag.

RIECKE, J. C. AND STONE, C. A. 1998. Privacy via Subsumption. In *Proceedings of FOOL'98*. Extented version to appear in Theory and Practice of Object Systems.

VISWANATHAN, R. 1998. Full abstraction and first-order objects with recursive types and subtyping. In *Proceedings of LICS'98*. 380–391.

A. GLOSSARY

A.1 THE SOURCE CALCULUS

Syntax

A, B, C	$::=$	*Types*	
		X, U	*variable*
		$\mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$	*object type*
			(v_i's and m_i's distinct)
a, b, c	$::=$	*Terms*	
		x	*variable*
		$\varsigma(X{=}A)\langle v_i = c_i{}^{i\in I}, m_j = \varsigma(x:X)b_j\{X,x\}^{j\in J}\rangle$	*object*
			(v_i's and m_i's distinct)
		$a.v$	*field selection*
		$a.v \leftarrow_A b$	*field update*
		$a.v \leftarrow\!\!+_A b$	*field addition*
		$a \circ m$	*method invocation*
		$a \circ m \leftarrow \varsigma(X{=}A)\varsigma(x:X)b\{X,x\}$	*method override*
		$a \circ m \leftarrow\!\!+ \varsigma(X{=}A)\varsigma(x:X)b\{X,x\}$	*method addition*

Notation Shorthands

$$
\begin{aligned}
\varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J}\rangle &\triangleq \varsigma(X{=}A)\langle v_i = c_i{}^{i\in I}, m_j = \varsigma(x:X)b_j\{X,x\}^{j\in J}\rangle \\
a\cdot\ell &\triangleq a\circ\ell \text{ or } a.\ell \\
a\cdot\ell \leftarrow_{(X,A)} \mathbf{b}\{X\} &\triangleq a\circ\ell \leftarrow \varsigma(X{=}A)\varsigma(x:X)b\{X,x\} \text{ or } a.\ell \leftarrow {}_A b \\
a\cdot\ell \leftarrow\!\!+_{(X,A)} \mathbf{b}\{X\} &\triangleq a\circ\ell \leftarrow\!\!+ \varsigma(X{=}A)\varsigma(x:X)b\{X,x\} \text{ or } a.\ell \leftarrow\!\!+ {}_A b
\end{aligned}
$$

Operational Semantics over closed terms of $\mathtt{Ob}^+$

Results$r = \varsigma(X{=}A)\langle v_i = c_i{}^{i\in I}, m_j = \varsigma(x:X)b_j\{X,x\}^{j\in J}\rangle$

$$
(\text{Select}_{\text{v}})\quad \frac{a \Downarrow_o \varsigma(X{=}A)\langle \ldots, v_j{=}c_j, \ldots\rangle \quad c_j \Downarrow_o r}{a.v_j \Downarrow_o r}
$$

$$
(\text{Select}_{\text{m}})\quad \frac{a \Downarrow_o \widehat{a} \quad b_j\{A,\widehat{a}\} \Downarrow_o r \quad (\widehat{a} \equiv \varsigma(X{=}A)\langle \ldots, m_j = \varsigma(x:X)b_j\{X,x\}, \ldots\rangle)}{a\circ m_j \Downarrow_o r}
$$

$$
(\text{Override})\quad \frac{a \Downarrow_o \varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J}\rangle \quad k\in I\cup J}{a\cdot\ell_k \leftarrow_{(X,A)} \mathbf{b}\{X\} \Downarrow_o \varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J-\{k\}}, \ell_k =_{(X)} \mathbf{b}\{X\}\rangle}
$$

$$
(\text{Extend})\quad \frac{a \Downarrow_o \varsigma(X{=}A)\langle \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J}\rangle \quad \ell\notin\{\ell_i\}^{i\in I\cup J}}{a\cdot\ell \leftarrow\!\!+_{(X,A^+)} \mathbf{b}\{X\} \Downarrow_o \varsigma(X{=}A^+)\langle \ell =_{(X)} \mathbf{b}\{X\}, \ell_i =_{(X)} \mathbf{b}_i\{X\}^{i\in I\cup J}\rangle}
$$

A.2 THE TARGET CALCULUS

Syntax

$\mathtt{K}$ ::=	*Kinds*	
	$\mathtt{T}$	type
	$\mathtt{K} \Rightarrow \mathtt{K}$	type operator
$\mathbb{A}, \mathbb{B}$ ::=	*Constructors*	
	X	constructor variable
	$\mathtt{Top}$	greatest constructor of kind $\mathtt{T}$
	$\mathbb{A} \rightarrow \mathbb{B}$	function type
	$[m_1 : \mathbb{B}, \ldots, m_k : \mathbb{B}]$	record type
	$\forall(X <: \mathbb{A} :: \mathtt{K})\mathbb{A}$	bounded universal type
	$\mu(X)\mathbb{A}$	recursive type
	$\lambda(X :: \mathtt{K})\mathbb{B}$	operator
	$\mathbb{B}(\mathbb{A})$	operator application
M, N ::=	*Expressions*	
	x	variable
	$\lambda(x : \mathbb{A})M$	abstraction
	$M\,N$	application
	$\Lambda(X <: \mathbb{A} :: \mathtt{K})e$	type-abstraction
	$M\,\mathbb{A}$	type-application
	$[m_1 = M_1, \ldots, m_k = M_k]$	record
	$M.m$	record selection
	$\mathtt{fold}(\mathbb{A}, M)$	recursive fold
	$\mathtt{unfold}(M)$	recursive unfold
	$\mathtt{let}\ x = M\ \mathtt{in}\ N$	local definition
	$\mathtt{letrec}\ f(x : \mathbb{A}) : \mathbb{B} = M\ \mathtt{in}\ N$	recursive local definition

Operational Semantics over closed terms of $F_{\omega<:\mu}$

Results r ::= $\lambda(x : \mathbb{A})\,M \mid [m_1 = M_1, \ldots, m_k = M_k] \mid \mathtt{fold}(\mathbb{A}, M) \mid \Lambda(X <: \mathbb{A} :: \mathtt{K})\,M$

$$(\beta_1)\qquad \frac{M \Downarrow_f (\lambda(x : \mathbb{A})M') \quad M'\{x := N\} \Downarrow_f r}{M\,N \Downarrow_f r}$$

$$(\beta_2)\qquad \frac{M \Downarrow_f (\Lambda(X <: \mathbb{A} :: \mathtt{K})\,N) \quad N\{X := \mathbb{B}\} \Downarrow_f r}{M\,\mathbb{B} \Downarrow_f r}$$

$$(\text{select})\qquad \frac{M \Downarrow_f [\ldots, m_j = M_j, \ldots] \quad M_j \Downarrow_f r}{M.m_j \Downarrow_f r}$$

$$(\text{unfold})\qquad \frac{M \Downarrow_f \mathtt{fold}(\mathbb{A}, N) \quad N \Downarrow_f r}{\mathtt{unfold}(M) \Downarrow_f r}$$

$$(\text{letrec})\qquad \frac{N\{f := M\{f := \mathtt{letrec}\ f = M\ \mathtt{in}\ f\}\} \Downarrow_f r}{\mathtt{letrec}\ f = M\ \mathtt{in}\ N \Downarrow_f r}$$

A.3 ENCODING OF EXTENSIBLE OBJECTS AND TYPES

Translation of Contexts and Judgements

$$
\begin{array}{rcl}
[\![\, \varnothing \,]\!] & \triangleq & \varnothing \\
[\![\, \Gamma, x{:}A \,]\!] & \triangleq & [\![\, \Gamma \,]\!], x{:}[\![\, \Gamma \triangleright A \,]\!]^{\mathrm{TY}} \\
[\![\, \Gamma, X \,]\!] & \triangleq & [\![\, \Gamma \,]\!], X \\
[\![\, \Gamma, X \lessdot\!\!\# A \,]\!] & \triangleq & [\![\, \Gamma \,]\!], X \leq [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}} \\
\\
[\![\, \Gamma \triangleright \varnothing \,]\!] & \triangleq & \varnothing \\
[\![\, \Gamma \triangleright \Gamma', x{:}A \,]\!] & \triangleq & [\![\, \Gamma \triangleright \Gamma' \,]\!], x{:}[\![\, \Gamma, \Gamma' \triangleright A \,]\!]^{\mathrm{TY}} \\
[\![\, \Gamma \triangleright \Gamma', X \,]\!] & \triangleq & [\![\, \Gamma \triangleright \Gamma' \,]\!], X \\
[\![\, \Gamma \triangleright \Gamma', X \lessdot\!\!\# A \,]\!] & \triangleq & [\![\, \Gamma \triangleright \Gamma' \,]\!], X \leq [\![\, \Gamma, \Gamma' \triangleright A \,]\!]^{\mathrm{IN}} \\
\\
[\![\, \Gamma \vdash * \,]\!] & \triangleq & [\![\, \Gamma \,]\!] \vdash \diamond \\
[\![\, \Gamma \vdash A \,]\!] & \triangleq & [\![\, \Gamma \,]\!] \vdash [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{TY}} \\
[\![\, \Gamma \vdash A \lessdot\!\!\# B \,]\!] & \triangleq & [\![\, \Gamma \,]\!] \vdash [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}} \leq [\![\, \Gamma \triangleright B \,]\!]^{\mathrm{IN}} \\
[\![\, \Gamma \vdash a : A \,]\!] & \triangleq & [\![\, \Gamma \,]\!] \vdash [\![\, \Gamma \triangleright a \,]\!] : [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{TY}}
\end{array}
$$

Translation of Types

$$A \equiv \mathtt{pro}(X)\langle v_i : C_i{}^{i \in I}, m_j : B_j\{X\}^{j \in J} \rangle$$

INTERFACES

$$
\begin{array}{l}
[\![\, \Gamma', X \lessdot\!\!\# A, \Gamma'' \triangleright X \,]\!]^{\mathrm{IN}} \triangleq X \\
\qquad [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}} \triangleq \lambda(X)[\, v_i^{sel} : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}\ i \in I},\ v_i^{upd} : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}} \to X^{\ i \in I}, \\
\qquad\qquad m_j^{sel} : [\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}\ j \in J}\ ]
\end{array}
$$

OPERATORS

$$
\begin{array}{l}
[\![\, \Gamma', X \lessdot\!\!\# A, \Gamma'' \triangleright X \,]\!]^{\mathrm{OP}} \triangleq X \\
[\![\, \Gamma \triangleright A \,]\!]^{\mathrm{OP}} \triangleq \lambda(X)[\, v_i^{sel} : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}\ i \in I},\ v_i^{upd} : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}} \to X^{\ i \in I}, \\
\qquad m_j^{sel} : [\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}\ j \in J}, \\
\qquad m_j^{gen} : \forall (U \leq [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}}) U^* \to [\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}} \{X{:=}U^*\}^{j \in J}, \\
\qquad ext : \forall (U \leq [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}}) U^* \to U^*]
\end{array}
$$

TYPES

$$
\begin{array}{l}
[\![\, \Gamma', X \lessdot\!\!\# A, \Gamma'' \triangleright X \,]\!]^{\mathrm{TY}} \triangleq X^* \\
[\![\, \Gamma', X, \Gamma'' \triangleright X \,]\!]^{\mathrm{TY}} \triangleq X \\
[\![\, \Gamma \triangleright A \,]\!]^{\mathrm{TY}} \triangleq \mu(X)[\, v_i^{sel} : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}\ i \in I},\ v_i^{upd} : [\![\, \Gamma \triangleright C_i \,]\!]^{\mathrm{TY}} \to X^{\ i \in I}, \\
\qquad m_j^{sel} : [\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}\ j \in J}, \\
\qquad m_j^{gen} : \forall (U \leq [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}}) U^* \to [\![\, \Gamma, X \triangleright B_j\{X\} \,]\!]^{\mathrm{TY}} \{X{:=}U^*\}^{j \in J}, \\
\qquad ext : \forall (U \leq [\![\, \Gamma \triangleright A \,]\!]^{\mathrm{IN}}) U^* \to U^* \ ]
\end{array}
$$

Translation of Terms

Notation Let be $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in 1..n}, m_j : B_j\{X\}^{j\in 1..m}\rangle\!\rangle$ in:

$$
\begin{array}{l}
\textsc{mkobj}_A(M_1,\ldots,M_n,N_1,\ldots,N_m) \equiv \\
\quad \mathtt{letrec}\ \mathit{mkobj}(w_i : [\![\,\Gamma \triangleright C_i\,]\!]^{\mathrm{TY}\, i\in 1..n}, \\
\qquad\qquad\qquad f_j : \forall(U \leq [\![\,\Gamma \triangleright A\,]\!]^{\mathrm{IN}})U^* \to [\![\,\Gamma, X \triangleright B_j\{X\}\,]\!]^{\mathrm{TY}}\{X{:=}U^*\}^{j\in 1..m} \\
\qquad\qquad\quad) : [\![\,\Gamma \triangleright A\,]\!]^{\mathrm{TY}} = \\
\qquad \mathtt{fold}(\,[\![\,\Gamma \triangleright A\,]\!]^{\mathrm{TY}}, \\
\qquad\quad [v_i^{sel} = w_i{}^{\;i\in 1..n} \\
\qquad\quad v_i^{upd} = \lambda(z : [\![\,\Gamma \triangleright C_i\,]\!]^{\mathrm{TY}})\mathit{mkobj}(w_l{}^{\;l\in 1..i-1},\ z,\ w_l{}^{\;l\in i+1..n}, f_1,\ldots,f_m)^{i\in 1..n}, \\
\qquad\quad m_j^{sel} = f_j(\,[\![\,\Gamma \triangleright A\,]\!]^{\mathrm{OP}})(\mathit{mkobj}(w_1,\ldots,w_n,f_1,\ldots,f_m))^{j\in 1..m}, \\
\qquad\quad m_j^{gen} = f_j{}^{j\in 1..m}, \\
\qquad\quad \mathit{ext} = \Lambda(U \leq [\![\,\Gamma \triangleright A\,]\!]^{\mathrm{IN}})\lambda(x : U^*)x] \\
\quad \mathtt{in}\ \mathit{mkobj}(M_1,\ldots,M_n,N_1,\ldots,N_m)
\end{array}
$$

Translation of Variables

$[\![\,\Gamma \triangleright x\,]\!] \triangleq x$

Translation of Objects

$[\![\,\Gamma \triangleright \varsigma(X{=}A)\langle v_i = c_i{}^{i\in I}, m_j = \varsigma(x:X)b_j\{X,x\}^{j\in J}\rangle\,]\!] \triangleq$
$\quad \textsc{mkobj}_A(\,[\![\,\Gamma \triangleright c_i\,]\!]^{i\in I}, \Lambda(U \leq [\![\,\Gamma \triangleright A\,]\!]^{\mathrm{IN}})\lambda(x:U^*)\,[\![\,\Gamma, U {<\!\!\#}\, A, x : U \triangleright b_j\{U,x\}\,]\!]^{j\in J})$
where $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$

Translation of Field and method selection

$[\![\,\Gamma \triangleright a \cdot \ell\,]\!] \triangleq \mathtt{unfold}(\,[\![\,\Gamma \triangleright a\,]\!]\,).\ell^{sel}$

Translation of Field update

$[\![\,\Gamma \triangleright a.v \leftarrow_A b\,]\!] \triangleq \mathtt{unfold}(\,[\![\,\Gamma \triangleright a\,]\!]\,).v^{upd}(\,[\![\,\Gamma \triangleright b\,]\!]\,)$

Translation of Field addition

$[\![\,\Gamma \triangleright a.v_{n+1} \longleftarrow\!\!\!+_{A^+}\ b\,]\!] \triangleq$
$\quad \widehat{a}.\mathit{ext}\ (\,[\![\,\Gamma \triangleright A^+\,]\!]^{\mathrm{OP}})\ (\textsc{mkobj}_{A^+}(\widehat{a}.v_i^{sel\ \ i\in 1..n},\ [\![\,\Gamma \triangleright b\,]\!], \widehat{a}.m_j{}^{gen\ \ j\in J}))$
where $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in 1..n},\ m_j : B_j\{X\}^{j\in J}\rangle\!\rangle,$
$\quad A^+ \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in 1..n+1},\ m_i : B_i\{X\}^{i\in J}\rangle\!\rangle\ .$
$\quad \widehat{a} \equiv \mathtt{unfold}(\,[\![\,\Gamma \triangleright a\,]\!]\,)$

Translation of Method addition

$[\![\,\Gamma \triangleright a \circ m_{n+1} \longleftarrow\!\!\!+\ \varsigma(X{=}A^+)\varsigma(x:X)b\{X,x\}\,]\!] \triangleq$
$\quad \widehat{a}.\mathit{ext}\ (\,[\![\,\Gamma \triangleright A^+\,]\!]^{\mathrm{OP}})\ (\textsc{mkobj}_{A^+}(\widehat{a}.v_i^{sel\ \ i\in I}, \widehat{a}.m_j^{gen\ \ j\in 1..n},\ \Lambda(U \leq [\![\,\Gamma \triangleright A^+\,]\!]^{\mathrm{IN}})\widehat{b}))$
where $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I},\ m_j : B_j\{X\}^{j\in 1..n}\rangle\!\rangle,$
$\quad A^+ \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I},\ m_i : B_i\{X\}^{i\in 1..n+1}\rangle\!\rangle,$.
$\quad \widehat{a} \equiv \mathtt{unfold}(\,[\![\,\Gamma \triangleright a\,]\!]\,)$, and $\widehat{b} \equiv \lambda(x:U^*)\,[\![\,\Gamma, U {<\!\!\#}\, A^+, x : U \triangleright b\{U,x\}\,]\!]$

Translation of Method override

$[\![\,\Gamma \triangleright a \circ m_k \leftarrow \varsigma(X{=}A)\varsigma(x:X)b\{X,x\}\,]\!] \triangleq$
$\widehat{a}.\mathit{ext}\ (\,[\![\,\Gamma \triangleright A\,]\!]^{\mathrm{OP}})\ (\textsc{mkobj}_A(\widehat{a}.v_i^{sel\ \ i\in I}, \widehat{a}.m_1{}^{gen},\ldots\Lambda(U \leq [\![\,\Gamma \triangleright A\,]\!]^{\mathrm{IN}})\widehat{b},\ldots,\widehat{a}.m_n{}^{gen}))$
where $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I},\ m_j : B_j\{X\}^{j\in 1..n}\rangle\!\rangle,\ \widehat{a} \equiv \mathtt{unfold}(\,[\![\,\Gamma \triangleright a\,]\!]\,)$
$\quad \widehat{b} \equiv \lambda(x:U^*)\,[\![\,\Gamma, U {<\!\!\#}\, A^+, x : U \triangleright b\{U,x\}\,]\!]$.

B. THE OBJECT CALCULUS $\mathtt{OB}^+$

Context Formation

(Ctx $\oslash$)
$$\frac{}{\varnothing \vdash *}$$

(Ctx Match)
$$\frac{\Gamma \vdash A \quad U \notin Dom(\Gamma) \quad (A \equiv \mathtt{pro}(X)\langle\!\langle \ell_i : B_i\{X\}^{i\in I}\rangle\!\rangle)}{\Gamma, U \lessdot\!\!\# A \vdash *}$$

(Ctx X)
$$\frac{\Gamma \vdash * \quad X \notin Dom(\Gamma)}{\Gamma, X \vdash *}$$

Type formation

(Type X)
$$\frac{\Gamma', X, \Gamma'' \vdash *}{\Gamma', X, \Gamma'' \vdash X}$$

(Type Match U)
$$\frac{\Gamma', U \lessdot\!\!\# A, \Gamma'' \vdash *}{\Gamma', U \lessdot\!\!\# A, \Gamma'' \vdash U}$$

(Type $\mathtt{pro}$)
$$\frac{\Gamma, X \vdash B_i\{X\}}{\Gamma \vdash \mathtt{pro}(X)\langle\!\langle \ell_i : B_i\{X\}^{i\in I}\rangle\!\rangle}$$

Term Formation

(Val x)
$$\frac{\Gamma', x : A, \Gamma'' \vdash *}{\Gamma', x : A, \Gamma'' \vdash x : A}$$

(Val Select)
$$\frac{\Gamma \vdash e : A \quad \Gamma \vdash A \lessdot\!\!\# \mathtt{pro}(X)\langle\!\langle \ell : B\{X\}\rangle\!\rangle}{\Gamma \vdash e \cdot \ell : B\{A\}}$$

(Val Object: $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$)
$$\frac{\Gamma \vdash c_i : C_i \qquad \Gamma, U \lessdot\!\!\# A, x : U \vdash b_j\{U, x\} : B_j\{U\} \qquad \forall\, i \in I, i \in J}{\Gamma \vdash \varsigma(X{=}A)\langle v_i = c_i{}^{i\in I}, m_j = \varsigma(x : X) b_j\{X, x\}^{j\in J}\rangle : A}$$

(Val Extend Field: $A \equiv \mathtt{pro}(X)\langle\!\langle \ell_i : B_i\{X\}^{i\in I}\rangle\!\rangle$, $A^+ \equiv \mathtt{pro}(X)\langle\!\langle \ell : B\{X\}, \ell_i : B_i\{X\}^{i\in I}\rangle\!\rangle$)
$$\frac{\Gamma \vdash a : A, \quad \Gamma, \vdash c : B, \quad (\ell \neq \ell_i \ \forall i \in I)}{\Gamma \vdash a . \ell \longleftarrow\!\!+_{A^+}\, c : A^+}$$

(Val Extend Method: $A \equiv \mathtt{pro}(X)\langle\!\langle \ell_i : B_i\{X\}^{i\in I}\rangle\!\rangle \quad A^+ \equiv \mathtt{pro}(X)\langle\!\langle \ell : B\{X\}, \ell_i : B_i\{X\}^{i\in I}\rangle\!\rangle$)
$$\frac{\Gamma \vdash a : A, \quad \Gamma, U \lessdot\!\!\# A^+, x : U \vdash b\{U, x\} : B\{U\}, \quad (\ell \neq \ell_i \ \forall i \in I)}{\Gamma \vdash a \circ m \longleftarrow\!\!+\ \varsigma(X{=}A)\varsigma(x : X) b\{X, x\} : A^+}$$

(Val Method Override: $A \equiv \mathtt{pro}(X)\langle\!\langle v_i : C_i{}^{i\in I}, m_j : B_j\{X\}^{j\in J}\rangle\!\rangle$)
$$\frac{\Gamma \vdash a : A, \quad \Gamma, U \lessdot\!\!\# A, x : U \vdash b : B\{U\} \quad k \in J}{\Gamma \vdash a \circ m_k \leftarrow \varsigma(x : A) b : A}$$

(Val Field Update)
$$\frac{\Gamma \vdash a : A, \quad \Gamma \vdash A \lessdot\!\!\# \mathtt{pro}(X)\langle\!\langle v : C\rangle\!\rangle \quad \Gamma \vdash c : C}{\Gamma \vdash a . v \leftarrow_A c : A}$$

Matching

(Match U)
$$\frac{\Gamma', U \lessdot\!\!\# A, \Gamma'' \vdash *}{\Gamma', U \lessdot\!\!\# A, \Gamma'' \vdash U \lessdot\!\!\# A}$$

(Match Refl)
$$\frac{\Gamma', U \lessdot\!\!\# A, \Gamma'' \vdash U}{\Gamma', U \lessdot\!\!\# A, \Gamma'' \vdash U \lessdot\!\!\# U}$$

(Match Trans)
$$\frac{\Gamma \vdash U \lessdot\!\!\# B \quad \Gamma \vdash B \lessdot\!\!\# A}{\Gamma \vdash U \lessdot\!\!\# A}$$

(Match $\mathtt{pro}$)

$$\frac{\Gamma \vdash \mathtt{pro}(X)\langle \ell_i : B_i\{X\}^{i\in 1..n+k}\rangle}{\Gamma \vdash \mathtt{pro}(X)\langle \ell_i : B_i\{X\}^{i\in 1..n+k}\rangle \mathrel{<\!\!\#} \mathtt{pro}(X)\langle \ell_i : B_i\{X\}^{i\in 1..n}\rangle}$$

C. ADDITIONAL RULES FOR $\mathtt{OB}^{+}_{<:}$

Context and Type Formation

(Ctx Sub)

$$\frac{\Gamma \vdash A \quad U \notin Dom(\Gamma)}{\Gamma, U <: A \vdash *}$$

(Type Sub X)

$$\frac{\Gamma', X <: A, \Gamma'' \vdash *}{\Gamma', X <: A, \Gamma'' \vdash X}$$

(Type $\mathtt{obj}$)

$$\frac{\Gamma, X \vdash B_i \quad \forall i \in I}{\Gamma \vdash \mathtt{obj}(X)\langle \ell_i : B_i{}^{i\in I}\rangle}$$

Term Formation

(Val Select $\mathtt{Obj}$)

$$\frac{\Gamma \vdash a : A \quad (A \equiv \mathtt{obj}(X)\langle \ell_i : B_i\{X\}^{i\in I}\rangle,\ \ j \in I)}{\Gamma \vdash a \cdot \ell_j : B_j\{A\}}$$

(Val Subsumption)

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash A <: B}{\Gamma \vdash a : B}$$

Subtyping

(Sub U)

$$\frac{\Gamma', U <: A, \Gamma'' \vdash *}{\Gamma', U <: A, \Gamma'' \vdash U <: A}$$

(Sub Refl)

$$\frac{\Gamma \vdash A}{\Gamma \vdash A <: A}$$

(Sub Trans)

$$\frac{\Gamma \vdash A_1 <: A_2 \quad \Gamma \vdash A_2 <: A_3}{\Gamma \vdash A_1 <: A_3}$$

(Sub $\mathtt{probjFM95}$)

$$\frac{\Gamma, Y, X <: Y \vdash B'_i\{X\} <: B_i\{Y\} \qquad (i = 1..n)}{\Gamma \vdash \mathtt{probj}(X)\langle \ell_i : B'_i\{X\}^{i\in 1..n+k}\rangle <: \mathtt{obj}(Y)\langle \ell_i : B_i\{Y\}^{i\in 1..n}\rangle}$$

(Sub $\mathtt{probjAC96}$)

$$\frac{\Gamma, X \vdash B_i\{X\} \quad B_i \textit{ covariant in } X \quad \forall i = 1..n+k}{\Gamma \vdash \mathtt{probj}_{\mathtt{AC}}(X)\langle \ell_i : B_i\{X\}^{i\in 1..n+k}\rangle <: \mathtt{obj}_{\mathtt{AC}}(X)\langle \ell_i : B_i\{X\}^{i\in 1..n}\rangle}$$

D. THE TARGET CALCULUS $F_{\omega<:\mu}$

Judgements

$\Gamma \vdash \diamond$	Γ is a valid context
$\Gamma \vdash \mathtt{K}\ \mathit{kind}$	$\mathtt{K}$ is a kind
$\Gamma \vdash \mathbb{A} :: \mathtt{K}$	Constructor $\mathbb{A}$ has kind $\mathtt{K}$
$\Gamma \vdash \mathbb{A} \leftrightarrow \mathbb{B} :: \mathtt{K}$	$\mathbb{A}$ and $\mathbb{B}$ are equivalent constructors of kind $\mathtt{K}$
$\Gamma \vdash \mathbb{A} <: \mathbb{B} :: \mathtt{K}$	$\mathbb{A}$ is a subconstructor of $\mathbb{B}$, both of kind $\mathtt{K}$
$\Gamma \vdash a : \mathbb{A}$	a is a value of type $\mathbb{A}$

Notation

$\lceil \mathtt{T} \rceil$	$\triangleq \mathtt{Top}$	Op	$\triangleq \mathtt{T} \Rightarrow \mathtt{T}$
$\lceil \mathtt{K}_1 \Rightarrow \mathtt{K}_2 \rceil$	$\triangleq \lambda(X :: \mathtt{K}_1)\lceil \mathtt{K}_2 \rceil$	$\mathbb{I} \leq \mathbb{J}$	$\triangleq \mathbb{I} <: \mathbb{J} :: Op$
$X :: \mathtt{K}$	$\triangleq X <: \lceil \mathtt{K} \rceil :: \mathtt{K}$	$\Gamma \vdash \mathbb{A}$	$\triangleq \Gamma \vdash \mathbb{A} :: \mathtt{T}$
$X <: \mathbb{A}$	$\triangleq X <: \mathbb{A} :: \mathtt{T}$	$\Gamma \vdash \mathbb{A} \leftrightarrow \mathbb{B}$	$\triangleq \Gamma \vdash \mathbb{A} \leftrightarrow \mathbb{B} :: \mathtt{T}$
X	$\triangleq X <: \mathtt{Top} :: \mathtt{T}$	$\Gamma \vdash \mathbb{A} <: \mathbb{B}$	$\triangleq \Gamma \vdash \mathbb{A} <: \mathbb{B} :: \mathtt{T}$

Context Formation

$$\text{(Env }\varnothing\text{)}\ \frac{}{\varnothing \vdash \diamond} \qquad \text{(Env } X <:\text{)}\ \frac{\Gamma \vdash \mathbb{A} :: \mathtt{K} \quad X \notin Dom(\Gamma)}{\Gamma, X <: \mathbb{A} :: \mathtt{K} \vdash \diamond} \qquad \text{(Env } x\text{)}\ \frac{\Gamma \vdash \mathbb{A} \quad x \notin Dom(\Gamma)}{\Gamma, x : \mathbb{A} \vdash \diamond}$$

Kind Formation

$$\text{(Kind }\mathtt{T}\text{)}\ \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathtt{T}\ kind} \qquad \text{(Kind }\Rightarrow\text{)}\ \frac{\Gamma \vdash \mathtt{K}\ kind \quad \Gamma \vdash \mathtt{H}\ kind}{\Gamma \vdash \mathtt{K} \Rightarrow \mathtt{H}\ kind}$$

Constructor Formation

$$\text{(Con } X\text{)}\ \frac{\Gamma', X <: \mathbb{A} :: \mathtt{K}, \Gamma'' \vdash \diamond}{\Gamma', X <: \mathbb{A} :: \mathtt{K}, \Gamma'' \vdash X :: \mathtt{K}} \qquad \text{(Con }\mathtt{Top}\text{)}\ \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathtt{Top}} \qquad \text{(Con }\to\text{)}\ \frac{\Gamma \vdash \mathbb{A} \quad \Gamma \vdash \mathbb{B}}{\Gamma \vdash \mathbb{A} \to \mathbb{B}} \qquad \text{(Con Record)}\ \frac{\Gamma \vdash \mathbb{B}_i \quad (\forall i \in I)}{\Gamma \vdash [m_i : \mathbb{B}_i]^{i \in I}}$$

$$\text{(Con }\forall\text{)}\ \frac{\Gamma, X <: \mathbb{A} :: \mathtt{K} \vdash \mathbb{B}}{\Gamma \vdash \forall(X <: \mathbb{A} :: \mathtt{K})\mathbb{B}} \qquad \text{(Con }\mu\text{)}\ \frac{\Gamma, X \vdash \mathbb{A}}{\Gamma \vdash \mu(X)\mathbb{A}} \qquad \text{(Con Abs)}\ \frac{\Gamma, X :: \mathtt{K} \vdash \mathbb{B} :: \mathtt{H}}{\Gamma \vdash \lambda(X :: \mathtt{K})\mathbb{B} :: \mathtt{K} \Rightarrow \mathtt{H}} \qquad \text{(Con Appl)}\ \frac{\Gamma \vdash \mathbb{B} :: \mathtt{K} \Rightarrow \mathtt{H} \quad \Gamma \vdash \mathbb{A} :: \mathtt{K}}{\Gamma \vdash \mathbb{B}(\mathbb{A}) :: \mathtt{H}}$$

Constructor Equivalence

$$\text{(Con Eq } X\text{)}\ \frac{\Gamma \vdash X :: \mathtt{K}}{\Gamma \vdash X \leftrightarrow X :: \mathtt{K}} \qquad \text{(Con Eq Symm)}\ \frac{\Gamma \vdash \mathbb{A} \leftrightarrow \mathbb{B} :: \mathtt{K}}{\Gamma \vdash \mathbb{B} \leftrightarrow \mathbb{A} :: \mathtt{K}} \qquad \text{(Con Eq Trans)}\ \frac{\Gamma \vdash \mathbb{A} \leftrightarrow \mathbb{B} :: \mathtt{K} \quad \Gamma \vdash \mathbb{B} \leftrightarrow \mathbb{C} :: \mathtt{K}}{\Gamma \vdash \mathbb{A} \leftrightarrow \mathbb{C} :: \mathtt{K}} \qquad \text{(Con Eq }\mathtt{Top}\text{)}\ \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathtt{Top} \leftrightarrow \mathtt{Top}}$$

$$\text{(Con Eq }\mu\text{)}\ \frac{\Gamma, X \vdash \mathbb{B} \leftrightarrow \mathbb{B}'}{\Gamma \vdash \mu(X)\mathbb{B} \leftrightarrow \mu(X)\mathbb{B}'} \qquad \text{(Con Eval Beta)}\ \frac{\Gamma, X :: \mathtt{K} \vdash \mathbb{B}\{X\} :: \mathtt{H} \quad \Gamma \vdash \mathbb{A} :: \mathtt{K}}{\Gamma \vdash (\lambda(X :: \mathtt{K})\mathbb{B}\{X\})(\mathbb{A}) \leftrightarrow \mathbb{B}\{\mathbb{A}\} :: \mathtt{H}}$$

$$\text{(Con Eq Record)}\ \frac{\Gamma \vdash \mathbb{B}_i \leftrightarrow \mathbb{B}'_i \quad (\forall i \in I)}{\Gamma \vdash [m_i : \mathbb{B}_i]^{i \in I} \leftrightarrow [m_i : \mathbb{B}'_i]^{i \in I}} \qquad \text{(Con Eq }\forall\text{)}\ \frac{\Gamma \vdash \mathbb{A} \leftrightarrow \mathbb{A}' :: \mathtt{K} \quad \Gamma, X <: \mathbb{A} :: \mathtt{K} \vdash \mathbb{B} \leftrightarrow \mathbb{B}'}{\Gamma \vdash \forall(X <: \mathbb{A} :: \mathtt{K})\mathbb{B} \leftrightarrow \forall(X <: \mathbb{A}' :: \mathtt{K})\mathbb{B}'}$$

$$\text{(Con Eq Abs)}\ \frac{\Gamma, X :: \mathtt{K} \vdash \mathbb{B} \leftrightarrow \mathbb{B}'_i :: \mathtt{H}}{\Gamma \vdash \lambda(X :: \mathtt{K})\mathbb{B} \leftrightarrow \lambda(X :: \mathtt{K})\mathbb{B}' :: \mathtt{K} \Rightarrow \mathtt{H}} \qquad \text{(Con Eq Appl)}\ \frac{\Gamma \vdash \mathbb{B} \leftrightarrow \mathbb{B}' :: \mathtt{K} \Rightarrow \mathtt{H} \quad \Gamma \vdash \mathbb{A} \leftrightarrow \mathbb{A}' :: \mathtt{K}}{\Gamma \vdash \mathbb{B}(\mathbb{A}) \leftrightarrow \mathbb{B}'(\mathbb{A}') :: \mathtt{H}}$$

Constructor Inclusion

$$\text{(Con Sub Relf)}\ \frac{\Gamma \vdash \mathbb{A} \leftrightarrow \mathbb{B} :: \mathtt{K}}{\Gamma \vdash \mathbb{A} <: \mathbb{B} :: \mathtt{K}} \qquad \text{(Con Sub Trans)}\ \frac{\Gamma \vdash \mathbb{A} <: \mathbb{B} :: \mathtt{K} \quad \Gamma \vdash \mathbb{B} <: \mathbb{C} :: \mathtt{K}}{\Gamma \vdash \mathbb{A} <: \mathbb{C} :: \mathtt{K}} \qquad \text{(Con Sub } X\text{)}\ \frac{\Gamma', X <: \mathbb{A} :: \mathtt{K}, \Gamma'' \vdash \diamond}{\Gamma', X <: \mathbb{A} :: \mathtt{K}, \Gamma'' \vdash X <: \mathbb{A} :: \mathtt{K}}$$

(Con Sub $\mathtt{Top}$)
$$\frac{\Gamma \vdash \mathbb{A}}{\Gamma \vdash \mathbb{A} <: \mathtt{Top}}$$

(Con Sub Arrow)
$$\frac{\Gamma \vdash \mathbb{A}' <: \mathbb{A} \quad \Gamma \vdash \mathbb{B} <: \mathbb{B}'}{\Gamma \vdash \mathbb{A} \to \mathbb{B} <: \mathbb{A}' \to \mathbb{B}'}$$

(Con Sub μ)
$$\frac{\Gamma \vdash \mu(X)\mathbb{A} \quad \Gamma \vdash \mu(Y)\mathbb{B} \quad \Gamma, Y, X <: Y \vdash \mathbb{A} <: \mathbb{B}}{\Gamma \vdash \mu(X)\mathbb{A} <: \mu(Y)\mathbb{B}}$$

(Con Sub Record)
$$\frac{\Gamma \vdash \mathbb{B}_i <: \mathbb{B}'_i \ (\forall i \in 1..n) \quad \Gamma \vdash \mathbb{B}_i \ (\forall i \in n+1..n+k)}{\Gamma \vdash [m_i : \mathbb{B}_i]^{i \in 1..n+k} <: [m_i : \mathbb{B}'_i]^{i \in 1..n}}$$

(Con Sub $\forall$)
$$\frac{\Gamma \vdash \mathbb{A}' <: \mathbb{A} :: \mathtt{K} \quad \Gamma, X <: \mathbb{A}' :: \mathtt{K} \vdash \mathbb{B} <: \mathbb{B}'}{\Gamma \vdash \forall(X <: \mathbb{A} :: \mathtt{K})\mathbb{B} <: \forall(X <: \mathbb{A}' :: \mathtt{K})\mathbb{B}'}$$

(Con Sub Abs)
$$\frac{\Gamma, X :: \mathtt{K} \vdash \mathbb{B} <: \mathbb{B}' :: \mathtt{H}}{\Gamma \vdash \lambda(X :: \mathtt{K})\mathbb{B} <: \lambda(X :: \mathtt{K})\mathbb{B}' :: \mathtt{K} \Rightarrow \mathtt{H}}$$

(Con Sub Appl)
$$\frac{\Gamma \vdash \mathbb{B} <: \mathbb{B}' :: \mathtt{K} \Rightarrow \mathtt{H} \quad \Gamma \vdash \mathbb{A} :: \mathtt{K}}{\Gamma \vdash \mathbb{B}(\mathbb{A}) <: \mathbb{B}'(\mathbb{A}) :: \mathtt{H}}$$

Term Typing

(Val Subsumption)
$$\frac{\Gamma \vdash M : \mathbb{A} \quad \Gamma \vdash \mathbb{A} <: \mathbb{B}}{\Gamma \vdash M : \mathbb{B}}$$

(Val x)
$$\frac{\Gamma', x : \mathbb{A}, \Gamma'' \vdash \diamond}{\Gamma', x : \mathbb{A}, \Gamma'' \vdash x : \mathbb{A}}$$

(Val Abs)
$$\frac{\Gamma, x : \mathbb{A} \vdash M : \mathbb{B}}{\Gamma \vdash \lambda(x : \mathbb{A})M : \mathbb{A} \to \mathbb{B}}$$

(Val Appl)
$$\frac{\Gamma \vdash M : \mathbb{A} \to \mathbb{B} \quad \Gamma \vdash N : \mathbb{A}}{\Gamma \vdash M\,N : \mathbb{B}}$$

(Val Abs2)
$$\frac{\Gamma, X <: \mathbb{A} :: \mathtt{K} \vdash M : \mathbb{B}}{\Gamma \vdash \Lambda(X <: \mathbb{A} :: \mathtt{K})M : \forall(X <: \mathbb{A} :: \mathtt{K})\mathbb{B}}$$

(Val Appl2)
$$\frac{\Gamma \vdash M : \forall(X <: \mathbb{A} :: \mathtt{K})\mathbb{B} \quad \Gamma \vdash \mathbb{A}' <: \mathbb{A} :: \mathtt{K}}{\Gamma \vdash M\mathbb{A}' : \mathbb{B}\{\mathbb{A}'\}}$$

(Val Record)
$$\frac{\Gamma \vdash M_i : \mathbb{B}_i \ (\forall i \in I) \quad \Gamma \vdash \mathbb{A} \leftrightarrow [m_i : \mathbb{B}_i]^{i \in I}}{\Gamma \vdash [m_i = M_i]^{i \in I} : \mathbb{A}}$$

(Val Select)
$$\frac{\Gamma \vdash M : [m_i : \mathbb{B}_i]^{i \in I} \quad (j \in I)}{\Gamma \vdash M.m_j : \mathbb{B}_j}$$

(Val Fold)
$$\frac{\Gamma \vdash M : \mathbb{B}\{\mathbb{A}\} \quad \Gamma \vdash A \equiv \mu(X)\mathbb{B}\{X\}}{\Gamma \vdash \mathtt{fold}(\mathbb{A}, M) : \mathbb{A}}$$

(Val Unfold)
$$\frac{\Gamma \vdash M : \mathbb{A} \quad A \equiv \mu(X)\mathbb{B}\{X\}}{\Gamma \vdash \mathtt{unfold}(M) : \mathbb{B}\{\mathbb{A}\}}$$

(Val Let)
$$\frac{\Gamma \vdash M : \mathbb{A} \quad \Gamma, x : \mathbb{A} \vdash N : \mathbb{B}}{\Gamma \vdash \mathtt{let}\ x : \mathbb{A} = M\ \mathtt{in}\ N : \mathbb{B}}$$

(Val LetRec)
$$\frac{\Gamma, f : \mathbb{A} \to \mathbb{B} \vdash \lambda(x : \mathbb{A})M : \mathbb{A} \to \mathbb{B} \quad \Gamma, f : \mathbb{A} \to \mathbb{B}, x : \mathbb{A} \vdash N : \mathbb{C}}{\Gamma \vdash \mathtt{letrec}\ f(x : \mathbb{A}) : \mathbb{B} = M\ \mathtt{in}\ N : \mathbb{C}}$$

Received September 2000; accepted April 2001