Subtyping object and recursive types logically[⋆]
(Extended Abstract)

Steffen van Bakel[1] Ugo de'Liguoro[2]

[1] Department of Computing, Imperial College,
180 Queen's Gate, London SW7 2BZ, UK,
svb@doc.ic.ac.uk

[2] Dipartimento di Informatica, Università di Torino,
Corso Svizzera 185, 10149 Torino, Italy
deliguoro@di.unito.it

Abstract. We study subtyping in first-order object calculi with respect to the logical semantics obtained by identifying terms that satisfy the same set of predicates, as formalized through an assignment system. It is shown that equality in the full first-order ς-calculus is modeled by this notion, which is included in a Morris-style contextual equivalence.

1 Introduction

Subtyping is a prominent feature of type-theoretic foundation of object oriented programming languages. The basic idea is expressed by subsumption: any piece of code of type A can masquerade as a code of type B whenever A is a subtype of B, written $A <: B$.

In typed calculi, equations are stated amongst terms of the same type; when terms may have several types because of subsumption, it is commonly postulated that if $a = b : A$ and $A <: B$ then $a = b : B$ (but not vice-versa): let's call this *equational subsumption*. In the realm of object calculi, object types are essentially *interfaces*, and subtyping *interface extension*; therefore, subsumption is justified by the intuition that any object which is able to react to messages mentioned in A a fortiori will answer correctly to messages in the smaller interfaces represented by its supertypes. Similarly, equational subsumption is understood on the ground of context separability: a and b are contextually equivalent at type A if both are typeable by A and no context with a hole of type A can distinguish them. This provides an interpretation of subtyping: $A <: B$ should hold if any pair of terms that are contextually equivalent at type A cannot be separated at type B.

Semantically this is understood in two ways, according to the existing literature (see [12] Ch. 10 for a gentle introduction to these approaches): either by means of coercions [6], or by inclusion of partial equivalence relations ([7, 8] and [1] Ch. 14). But coercion semantics does not reflect the actual implementation practice of object-oriented languages; also, PER semantics is quite complex to use for reasoning about programs, and suffers of technical problems which are still open.

⋆ Partially supported by the MIKADO project of the IST-FET Global Computing Initiative, no IST-2001-32222

We propose a third approach which, in our view, can lead to a simpler logical framework for reasoning about object oriented programs. It is based on the ideas of logical semantics and domain logic. In the latter perspective, the meaning of a term is determined by the set of the predicates it satisfies, so that two terms are equivalent if they are indiscernible. To account for equivalence "at" a certain type A we relativize this form of absolute indiscernibility to sets of *predicates* indexed over types, calling them *languages*. Hence a and b are logically equivalent at type A if they satisfy the same set of predicates form the language $\mathcal{L}_A$ associated to A.

For equational subsumption to be sound in our framework, it is needed that some relation between $\mathcal{L}_A$ and $\mathcal{L}_B$ exists whenever $A <: B$. Were we dealing with a calculus of pure objects, such a relation would be simply $\mathcal{L}_A \supseteq \mathcal{L}_B$, and this is clearly enough. However, since here we consider a richer calculus with functions and recursive types, called $\mathsf{FOb}_{1<:\mu}$ in [1], this is no longer true in general, and is replaced by a more complex inclusion relation.

The logical equivalence is indeed the theory of a model. Such a model can be obtained by the filter model construction as in [5], with a more complex structure due to the presence of types (see [10] and [4]). Here we leave the investigation of the model aside and concentrate on the theory itself, establishing two results:

1. if $\vdash a \leftrightarrow b : A$ is derivable in the equational theory of system $\mathsf{FOb}_{1<:\mu}$, then a and b are logically equivalent at type A;
2. two terms logically equivalent at type A are contextually equivalent at the same type.

The latter result is a consequence of the characterisation of convergence in terms of derivability of non-trivial predicates in $\mathcal{L}_A$ much as in the case of λ-calculus and intersection types (see e.g. [3]). A similar result was proved for the type-free ς-calculus in [9].

Because of the limited space available, proofs are presented in the appendix.

1.1 Related work

The present paper follows some previous works by the authors in [9, 10, 4]. The added feature of this paper is the treatment of sub-typing of object and recursive types, while sub-typing polymorphism was considered in [10] for a λ-calculus with function and record types only. The idea of using languages to model types in a filter model originates from [2]: however, in Abramsky's work the modelling of polymorphism was left out. In this case the predicate languages cannot be disjoint; moreover, they need to have a structure reflecting the sub-typing relation, as stressed above, a topic which has not been addressed in the literature.

The theory of objects in [1] is a natural environment for the investigation of the themes we address here; Morris-style contextual equivalence for first-order object calculi is introduced and studied in [11], where system $\mathsf{FOb}_{1<:\mu}$ is considered: this is the reason for the choice of the same calculus in the present paper.

Fig. 1 Fragments $\Delta_K \cup \Delta_x \cup \Delta_{\mathsf{Ob}} \cup \Delta_{\rightarrow} \cup \Delta_X \cup \Delta_\mu$

$$(\mathsf{Env}\ \emptyset):\ \frac{}{\emptyset \vdash \diamond} \qquad (\mathsf{Type\ Const}):\ \frac{E \vdash \diamond}{E \vdash K} \qquad (\mathsf{Env}\ x):\ \frac{E \vdash A}{E, x{:}A \vdash \diamond}\ (x \notin E) \qquad (\mathsf{Val}\ x):\ \frac{E', x{:}A, E'' \vdash \diamond}{E', x{:}A, E'' \vdash x{:}A} \qquad (\mathsf{Type\ Rec}):\ \frac{E, X \vdash A}{E \vdash \mu X.A}$$

$$(\mathsf{Type\ Object}):\ \frac{E \vdash B_i \quad (\forall i \in I)}{E \vdash [\ell_i{:}B_i\ ^{(i\in I)}]} \qquad (\mathsf{Type\ Arrow}):\ \frac{E \vdash A \quad E \vdash B}{E \vdash A \rightarrow B} \qquad (\mathsf{Env}\ X):\ \frac{E \vdash \diamond}{E, X \vdash \diamond}\ (X \notin E) \qquad (\mathsf{Type}\ X):\ \frac{E', X, E'' \vdash \diamond}{E', X, E'' \vdash X}$$

We omit rules (Val Object),(Val Select),(Val Fun), (Val Appl), (Val Fold), and (Val Unfold), since these can be easily constructed from the rules in Figure 3.

2 The system $\mathsf{FOb}_{1<:\mu}$

To keep the present exposition self-contained, we recall the definition of the system $\mathsf{FOb}_{1<:\mu}$ of [1]. As usual for polymorphic calculi, we will introduce type and term syntax in two steps: first by defining type expressions (pre-types) and pre-terms, namely terms decorated by pre-types; types and terms are then defined together with the the type derivation system as well-formed pre-types and well-typed pre-terms respectively.

Definition 1 (Pre-types and Pre-terms). Let $\mathcal{K}$ be a set of type constants, ranged over by K, and $\mathcal{V}$ a set of type-variables, ranged over by X, $\{\ell_i \mid i \in N\}$ a denumerable set of *labels*, I, J finite subsets of $\mathbb{N}$. The set of *types* $\mathcal{T}$, ranged over by $A, B, C, \ldots$ is defined by the following grammar:

$$A, B ::= K \mid X \mid [\ell_i{:}B_i\ ^{(i\in I)}] \mid A \rightarrow B \mid \mu X.A$$

The pre-terms of $\mathsf{FOb}_{1<:\mu}$ are defined through the following grammar, where c ranges over constants:

$$\begin{aligned} a, b ::= {} & x \mid c \mid \lambda x^A.a \mid a(b) \mid [\ell_i = \varsigma(x_i^A)b_i\ ^{(i\in I)}] \mid a.\ell \mid \\ & a.\ell \Leftarrow \varsigma(x)b \mid \mathsf{fold}(A, a) \mid \mathsf{unfold}(a) \end{aligned}$$

A type expression of the shape $[\ell_i{:}B_i\ ^{(i\in I)}]$ is used for an object type; $A \rightarrow B$ is the usual functional type and $\mu X.A$ is a recursive type: in the latter the type-variable X is bound in A. In the expressions $\varsigma(x^A)b$ and $\lambda x^A.b$, x is bound in b; free and bound variables are defined as usual. Types and pre-terms are considered equal modulo α-conversion, i.e. up to renaming of bound variables.

In [1] the system is defined as the union of several fragments, which we subdivide into two parts; the first one concerns contexts, types and terms formation[3]:

[3] As in [1], we will use a short-hand for rules, and write for example (where $I = \{1, \ldots, n\}$)

$$\frac{E, x_i{:}A \vdash_\Sigma b_i{:}B_i \quad (\forall i \in I)}{E \vdash_\Sigma [\ell_i = \varsigma(x_i^A)b_i\ ^{(i\in I)}]{:}A} \quad \text{for} \quad \frac{E, x_1{:}A \vdash_\Sigma b_1{:}B_1 \quad \ldots \quad E, x_n{:}A \vdash_\Sigma b_n{:}B_n}{E \vdash_\Sigma [\ell_i = \varsigma(x_i^A)b_i\ ^{(i\in I)}]{:}A}$$

Fig. 2 Fragments $\varDelta_{<:} \cup \varDelta_{<:\mathsf{Ob}} \cup \varDelta_{<:\rightarrow} \cup \varDelta_{<:X} \cup \varDelta_{\mu}$.

$$(\mathsf{Sub\ Refl}):\ \frac{E \vdash A}{E \vdash A <: A} \qquad (\mathsf{Sub\ Trans}):\ \frac{E \vdash A <: B \quad E \vdash B <: C}{E \vdash A <: C} \qquad (\mathsf{Val\ Subsumption}):\ \frac{E \vdash a{:}A \quad E \vdash A <: B}{E \vdash a{:}B}$$

$$(\mathsf{Type\ Top}):\ \frac{E \vdash \diamond}{E \vdash \mathsf{Top}} \qquad (\mathsf{Sub\ Top}):\ \frac{E \vdash A}{E \vdash A <: \mathsf{Top}} \qquad (\mathsf{Sub\ Object}):\ \frac{E \vdash B_i \quad (\forall i \in I)}{E \vdash [\ell_i{:}B_i\ ^{i \in I}] <: [\ell_i{:}B_i\ ^{i \in J}]}\ (J \subseteq I)$$

$$(\mathsf{Sub\ Arrow}):\ \frac{E \vdash A' <: A \quad E \vdash B <: B'}{E \vdash A \rightarrow B <: A' \rightarrow B'} \qquad (\mathsf{Env}\ X{<:}):\ \frac{E \vdash A}{E, X <: A \vdash \diamond}\ (X \notin \mathit{dom}(E)) \qquad (\mathsf{Type}\ X{<:}):\ \frac{E', X <: A, E'' \vdash \diamond}{E', X <: A, E'' \vdash X}$$

$$(\mathsf{Sub}\ X):\ \frac{E', X <: A, E'' \vdash \diamond}{E', X <: A, E'' \vdash X <: A} \qquad (\mathsf{Type\ Rec}{<:}):\ \frac{E, X <: \mathsf{Top} \vdash A}{E \vdash \mu X.A}$$

$$(\mathsf{Sub\ Rec}):\ \frac{E \vdash \mu X.A \quad E \vdash \mu Y.B \quad E, Y <: \mathsf{Top}, X <: Y \vdash A <: B}{E \vdash \mu X.A <: \mu Y.B}$$

Definition 2. 1. A *context* for a type judgement is just a finite set E of type-decorated variables, of the shape $x{:}A$, and we write $x \in E$ if there exists A such that $x{:}A \in E$.
2. The system $\varDelta_K \cup \varDelta_x \cup \varDelta_{\mathsf{Ob}} \cup \varDelta_{\rightarrow} \cup \varDelta_X \cup \varDelta_{\mu}$ is given in Figure 1.
3. E is a *well-formed context* if $E \vdash \diamond$ is derivable, and A *is a type for* a if there exists E with $E \vdash a{:}A$.

The second one is about sub-typing:

Definition 3. The system $\varDelta_{<:} \cup \varDelta_{<:\mathsf{Ob}} \cup \varDelta_{<:\rightarrow} \cup \varDelta_{<:X} \cup \varDelta_{<:\mu}$ can be found in Figure 2.

It is understood that such unions produce a set of inductive clauses generating a unique system where contexts and types in the rules from the first part can be formed according to the rules of the second part and vice-versa. There is also a certain redundancy: the context E, X is the same as $E, X <: \mathsf{Top}$. In what follows we will use the generic notation $\cdot\ \{\cdot \hookleftarrow \cdot\}$ for substitution both of type-variables by type expressions and of term-variables by terms, implicitly replacing all occurrences of the first parameter of $\{\cdot \hookleftarrow \cdot\}$ by the second in the preceding expression; as usual the replacements occur up to α-congruence to avoid variable clashes.

Definition 4 (Reduction). *Evaluating contexts* are term expressions with a hole $[\_]$, and are generated by the grammar:

$$\mathcal{E}[\_] ::= \_ \mid \mathcal{E}[\_].\ell \mid \mathcal{E}[\_].\ell \Leftarrow \varsigma(x^A)b \mid \mathcal{E}[\_](a) \mid \mathsf{unfold}(\mathcal{E}[\_]) \mid \mathsf{fold}(A, \mathcal{E}[\_]).$$

We will write $\mathcal{E}[a]$ for the replacement of $\_$ by a in $\mathcal{E}$.

The *one-step reduction relation* on terms is the binary relation defined by the following rules:

$$\begin{array}{rcl}
[\ell_i = \varsigma(x_i^{A_i})b_i \; ^{(i\in I)}].\ell_j & \rightarrow & b_j\{x_j \hookleftarrow [\ell_i = \varsigma(x_i^{A_i})b_i \; ^{(i\in I)}]\} \\
{[\ell_i = \varsigma(x_i^{A_i})b_i \; ^{(i\in I)}]}.\ell_j \Leftarrow \varsigma(x^A)b & \rightarrow & [\ell_i = \varsigma(x_i^{A_i})b_i{}^{i\in I\setminus j}, \ell_j = \varsigma(x^{A_j})b] \\
(\lambda x^A.a)(b) & \rightarrow & a\{x \hookleftarrow b\} \\
\mathsf{unfold}(\mathsf{fold}(X,a)) & \rightarrow & a \\
a \;\rightarrow\; b & \Rightarrow & \mathcal{E}[a] \;\rightarrow\; \mathcal{E}[b]
\end{array}$$

The relation $\xrightarrow{*}$ is the reflexive and transitive closure of $\rightarrow$.

The one-step reduction is from [11]. In [1], Ch. 6 the operational semantics of the object calculi is defined by means of a big-step predicate $a \rightsquigarrow v$, where a is a closed term, v is a *value* as it is defined by the grammar:

$$v ::= c \mid \lambda x^A.a \mid [\ell_i{:}\varsigma(x_i^A)b_i \; ^{i\in I}] \mid \mathsf{fold}(A,v).$$

It is easy to see that $a \rightsquigarrow v$ if and only if $a \xrightarrow{*} v$. The reduction relation is more general since it is defined for any term (possibly with free variable occurrences); it is even true that normal forms are not necessarily values. However it is easy to adapt the arguments in [1] to establish the following theorem:

Theorem 5 (Subject reduction property of $\mathsf{FOb}_{1<:\mu}$). If $E \vdash a{:}A$ is derivable in the system $\mathsf{FOb}_{1<:\mu}$ and $a \;\rightarrow\; b$, then $E \vdash b{:}A$ is derivable as well.

We just stress that, consistently with the definition of $\rightsquigarrow$ in [1], in the clause:

$$[\ell_i = \varsigma(x_i^{A_i})b_i \; ^{(i\in I)}].\ell_j \Leftarrow \varsigma(x^A)b \;\rightarrow\; [\ell_i = \varsigma(x_i^{A_i})b_i{}^{i\in I\setminus j}, \ell_j = \varsigma(x^{A_j})b]$$

a renaming of the self type of the bound variable x^A into x^{A_j} occurs. This is immaterial in the fragments of the ς-calculus without sub-typing, but it is needed in the presence of rule $(\mathsf{Val\ Subsumption})$ since if $A = [\ell_i{:}B_i{}^{i\in I}]$, and $A <: C$, then we can give type C to any term of type A and therefore update a method in an object of type A with $\varsigma(x^C)b$; but the result of (naively) performing the update saving the self type C is no longer typeable, as the *selves* of the methods now have different types, so that rule $(\mathsf{Val\ Object})$ will not apply.

The reduction relation is trivially confluent. Even relaxing Definition 4 and taking the closure of $\rightarrow$ under arbitrary contexts would not destroy confluence, as can be shown e.g. by adapting the Martin-Löf technique for proving the Church-Rosser theorem for the λ-calculus. As for typed λ-calculi with recursion (e.g. PCF), typed terms do not necessarily have a normal form: $\Omega_B \equiv [\ell = \varsigma(x^A)x.\ell].\ell$ is typeable by B if A is any object type $[\ell{:}B, \ldots]$, and it is such that $\Omega_B \;\rightarrow\; \Omega_B$.

3 Predicates and assignment

In this section we will introduce the syntax of the predicates and an assignment system to syntactically derive judgements associating predicates to typed terms under the assumption of similar judgements about a finite set of typed variables.

Predicates are transparently intersection types for a λ-calculus with records, and come from [9]. The essential difference is that the set of predicates is stratified into languages (see [10, 4]), in such a way that whenever a predicate can be deduced for a term a, it belongs to the language $\mathcal{L}_A$ associated with A.

Much in the style of [3], in this section we will present a notion of *strict intersection types*, called *strict predicates* here; this is a technical choice and a departure from [4], making the proof theory of the system more manageable, without loss of expressivity. Using these, we will define a notion of *predicate assignment*, which will consists basically of associating a predicate to a typed term.

Definition 6 (Predicates). $\mathcal{P}_{\text{S}}$, the set of *strict predicates*, and the set $\mathcal{P}$ of *intersection predicates*, both ranged over by $\sigma, \tau, \ldots$, are defined through:

$$\begin{array}{rll} \mathcal{P}_{\text{S}} & ::= \kappa \mid (\mathcal{P} \rightarrow \mathcal{P}_{\text{S}}) \mid \langle \ell{:}\mathcal{P}_{\text{S}} \rangle \mid \mu(\mathcal{P}_{\text{S}}) & \\ \mathcal{P} & ::= (\mathcal{P}_{\text{S}1} \wedge \ldots \wedge \mathcal{P}_{\text{S}n}) & (n \geq 0) \end{array}$$

where κ ranges over a countable set of atoms. We will write ω for an intersection of zero strict types, and write $\wedge_{\underline{n}}\sigma_i$ for $\sigma_1 \wedge \ldots \wedge \sigma_n$, where we assume that each $\sigma_i \in \mathcal{P}_{\text{S}}$. Also, rather than $\langle \ell{:}\sigma_1 \rangle \wedge \cdots \wedge \langle \ell{:}\sigma_n \rangle$ we will write $\langle \ell{:}\sigma_1 \wedge \cdots \wedge \sigma_n \rangle$ or $\langle \ell{:}\wedge_{\underline{n}}\sigma_i \rangle$, where $\underline{n} = \{1, \ldots, n\}$; also, rather than $\langle \ell_1{:}\sigma_1 \rangle \wedge \cdots \wedge \langle \ell_n{:}\sigma_n \rangle$ where the ℓ_i are distinct, we will write $\langle \ell_i{:}\sigma_i{}^{i \in \underline{n}} \rangle$ or $\langle \ell_i{:}\sigma_i{}^{\,(i \in I)} \rangle$.

Atomic predicates κ are intended to describe elements of atomic type in the domain of interpretation; $\sigma \rightarrow \tau$ is the property of functions sending element satisfying σ into elements satisfying τ; $\langle \ell{:}\sigma \rangle$ is the property of records having values that satisfy σ associated with the field ℓ. Predicates ω and $\sigma \wedge \tau$ mean '*truth*' and '*conjunction*' respectively. It should be noted that arbitrary intersection predicates like $(\sigma \rightarrow \tau) \wedge \langle \ell{:}\rho \rangle$ are allowed by the above definition.

To build a logic of predicates we need a notion of implication, written $\sigma \leq \tau$, which is a reflexive and transitive relation on predicates, defined below.

Definition 7 (Predicate pre-order). On predicates a pre-order $\leq$ is inductively defined by:

$$\frac{\sigma \leq \sigma_i}{\sigma \leq \wedge_{\underline{n}}\sigma_i}\ (\forall i \leq n \geq 0) \qquad \frac{}{\wedge_{\underline{n}}\sigma_i \leq \sigma_i}\ (\forall i \leq n \geq 1)$$

$$\frac{\rho \leq \sigma \quad \tau \leq \mu}{\sigma \rightarrow \tau \leq \rho \rightarrow \mu} \qquad \frac{\sigma \leq \tau \leq \rho}{\sigma \leq \rho} \qquad \frac{\sigma \leq \tau}{\langle \ell{:}\sigma \rangle \leq \langle \ell{:}\tau \rangle} \qquad \frac{\sigma \leq \tau}{\mu(\sigma) \leq \mu(\tau)}$$

Finally $\sigma = \tau \iff \sigma \leq \tau \leq \sigma$. A predicate is called *trivial* if it is equivalent to ω.

Lemma 8. *The following rules are admissible*

$$\frac{}{\langle \ell_i{:}\sigma_i{}^{i \in I} \rangle \wedge \langle \ell_j{:}\tau_j{}^{j \in J} \rangle \leq \langle \ell_k{:}\rho_k{}^{\,(k \in I \cup J)} \rangle}, \text{ where } \begin{cases} \rho_k = \sigma_k \wedge \tau_k, & \text{if } k \in I \cap J, \\ \rho_k = \sigma_k, & \text{if } k \in I \backslash J, \\ \rho_k = \tau_k, & \text{if } k \in J \backslash I \end{cases}$$

$$\frac{}{\langle \ell_i{:}\sigma_i{}^{\,(i \in I)} \rangle \leq \langle \ell_j{:}\sigma_j{}^{j \in J} \rangle}\ (J \subseteq I)$$

Lemma 9. $\langle \ell_i{:}\sigma_i{}^{i\in I}\rangle \wedge \langle \ell_j{:}\tau_j{}^{j\in J}\rangle = \langle \ell_k{:}\rho_k{}^{(k\in I\cup J)}\rangle$, *provided* $\sigma_i = \tau_i$ *for* $i \in I \cap J$.

Although predicates are basically properties of untyped terms (resulting from typed terms essentially by erasing type decorations), types are quite relevant in the equational theory of the $\mathsf{FOb}_{1<:\mu}$ calculus; this was accounted for in [10, 4] by means of the notion of *predicate languages*, whose definition easily extends to the present richer syntax.

Definition 10 (Languages). The set of all predicates $\mathcal{L}$ is stratified into a family $\{\mathcal{L}_A\}_A$ of sets of predicates called *languages*, indexed over closed types such that:

1. for every κ, there exists *exactly one* $K \in \mathcal{K}$ such that $\kappa \in \mathcal{L}_K$;
2. $\mathcal{L}_A$ is the least set (including atoms if $A \equiv K$) such that

$$\frac{\sigma_i \in \mathcal{L}_A \quad (\forall i \in \underline{n})}{\wedge_{\underline{n}}\sigma_i \in \mathcal{L}_A}\ (n \geq 0) \qquad \frac{\sigma \in \mathcal{L}_A \quad \tau \in \mathcal{L}_B}{\sigma \to \tau \in \mathcal{L}_{A\to B}} \qquad \frac{\sigma \in \mathcal{L}_{A\{X \hookleftarrow \mu X.A\}}}{\mu(\sigma) \in \mathcal{L}_{\mu X.A}}\ (\sigma \in \mathcal{P}_{\mathrm{s}})$$

$$\frac{\sigma \in \mathcal{L}_{A\to B_j}}{\langle \ell_j{:}\sigma\rangle \in \mathcal{L}_A}\ (A = [\ell_i{:}B_i{}^{(i\in I)}], j \in I, \sigma \in \mathcal{P}_{\mathrm{s}})$$

The intuition behind languages is the following. Properties in $\mathcal{L}_A$ give some information about values of type A; to be a value of type A should then imply to enjoy at least a non-trivial property in $\mathcal{L}_A$. That two values are logically equivalent at type A means that they satisfy the same set of properties in that language; consistently $\mathcal{L}_{\mathsf{Top}}$ is the set of trivial types. A natural question is whether there exists a relation between languages and the sub-typing relation, which is partly answered in the following proposition, for which we need to introduce the following definition.

Definition 11 (Language extension). We say that $\mathcal{L}_B$ is an *extension of* $\mathcal{L}_A$ (written $\mathcal{L}_A \sqsubseteq^\natural \mathcal{L}_B$), if and only if $\forall \sigma \in \mathcal{L}_A \exists \tau \in \mathcal{L}_B.\ \sigma \leq \tau$, and $\forall \tau \in \mathcal{L}_B \exists \sigma \in \mathcal{L}_A.\ \sigma \leq \tau$.

Proposition 12. Let A and B be closed type expressions not including recursion and such that $\vdash A <: B$ then:

1. if A and B are object types then $\mathcal{L}_B \subseteq \mathcal{L}_A$;
2. if A and B are arrow types then $\mathcal{L}_A \sqsubseteq^\natural \mathcal{L}_B$.

The relation $\sqsubseteq^\natural$ is Egli-Milner pre-order of (arbitrary) sets of predicates generated by $\leq$. Note that, since $\omega \in \mathcal{L}_B$ for any B (take $n = 0$ in the rule about intersection in Definition 10) and $\sigma \leq \omega$ for all σ, we have that $\mathcal{L}_B \subseteq \mathcal{L}_A$ implies $\mathcal{L}_A \sqsubseteq^\natural \mathcal{L}_B$.

The proof of Proposition 12 is by induction on the derivation of $\vdash A <: B$ and does not need to take the context into account at any step because of the assumptions (this is no longer true when recursive types are considered).

Definition 13. A map η from type-variables to closed types is called a *type-environment*. For E a well-formed context, we say that η *respects the context* E if for any $X <: A \in E$ (if $X \in E$ then it is read as $X <: \mathsf{Top} \in E$) it is the case that $\mathcal{L}_{\eta(X)} \sqsubseteq^\natural \mathcal{L}_{\eta(A)}$, where $\eta(A)$ is the value of application to A of the obvious extension of η to the set of types.

Fig. 3 Predicate Assignment

(Val x) :
$$\frac{}{\mathit{\Gamma} \vdash x{:}B : \sigma}\ (x{:}B{:}\tau \in \mathit{\Gamma}, \tau \le \sigma)$$

($<:$) :
$$\frac{\mathit{\Gamma} \vdash a{:}B : \sigma \quad \overline{\mathit{\Gamma}} \vdash B <: C}{\mathit{\Gamma} \vdash a{:}C : \sigma}\ (\sigma \in \mathcal{L}_C)$$

(Val Fun) :
$$\frac{\mathit{\Gamma}, x{:}A{:}\tau \vdash a{:}B : \sigma}{\mathit{\Gamma} \vdash \lambda x^A.a{:}A\to B : \tau\to\sigma}$$

(Val Appl) :
$$\frac{\mathit{\Gamma} \vdash a{:}A\to B : \tau\to\sigma \quad \mathit{\Gamma} \vdash b{:}A : \tau}{\mathit{\Gamma} \vdash a(b){:}B : \sigma}$$

(Val Fold) :
$$\frac{\mathit{\Gamma} \vdash a{:}A\{X \mapsfrom \mu X.A\} : \sigma}{\mathit{\Gamma} \vdash \mathsf{fold}(\mu X.A, a){:}\mu X.A : \mu(\sigma)}$$

(Val Unfold) :
$$\frac{\mathit{\Gamma} \vdash a{:}\mu X.A : \mu(\sigma)}{\mathit{\Gamma} \vdash \mathsf{unfold}(a){:}A\{X \mapsfrom \mu X.A\} : \sigma}$$

(Val Select) :
$$\frac{\mathit{\Gamma} \vdash a{:}A : \langle \ell_j{:}\tau\to\sigma\rangle \quad \mathit{\Gamma} \vdash a{:}A : \tau}{\mathit{\Gamma} \vdash a.\ell_j{:}B_j : \sigma}$$

(Val Object) :
$$\frac{\mathit{\Gamma}, x_i{:}A{:}\tau_i \vdash b_i{:}B_i : \sigma_i \quad (\forall i \in I)}{\mathit{\Gamma} \vdash [\ell_i = \varsigma(x_i^A)b_i\ ^{(i\in I)}]{:}A : \langle \ell_j{:}\tau_j\to\sigma_j\rangle}\ (j \in I)$$

(Val Update$_1$) :
$$\frac{\mathit{\Gamma} \vdash a{:}A : \sigma \quad \mathit{\Gamma}, y{:}A{:}\rho \vdash b{:}B_j : \tau}{\mathit{\Gamma} \vdash (a.\ell_j \Leftarrow \varsigma(y^A)b){:}A : \langle \ell_j{:}\rho\to\tau\rangle}$$

(Val Update$_2$) :
$$\frac{\mathit{\Gamma} \vdash a{:}A : \langle \ell_j{:}\sigma\rangle \quad \mathit{\Gamma}, y{:}A{:}\rho \vdash b{:}B_j : \tau}{\mathit{\Gamma} \vdash (a.\ell_j \Leftarrow \varsigma(y^A)b){:}A : \langle \ell_j{:}\sigma\rangle}\ (i \neq j)$$

(ω) :
$$\frac{E \vdash a{:}B}{\mathit{\Gamma} \vdash a{:}B : \omega}\ (E \triangleleft \mathit{\Gamma})$$

($\wedge I$) :
$$\frac{\mathit{\Gamma} \vdash a{:}B : \sigma_i \quad (\forall i \in \underline{n})}{\mathit{\Gamma} \vdash a{:}B : \wedge_{\underline{n}}\sigma_i}\ (n \ge 1)$$

$A \equiv [\ell_i{:}B_i\ ^{(i\in I)}]$ in rules (Val Select), (Val Object), (Val Update$_1$), and (Val Update$_2$).

Theorem 14. *If $E \vdash A <: B$, then for any type-environment η that respects E we have $\mathcal{L}_{\eta(A)} \sqsubseteq^{\natural} \mathcal{L}_{\eta(B)}$.*

We are now in place to introduce the main tool of the present work, namely the predicate assignment system. It is a formal system to derive judgements of the form $a{:}A : \sigma$, whose intended meaning is: the denotation of a satisfies the property σ when seen as a value of type A (here a "value" could be the undefined object in the domain of interpretation: we shall see that in such a case σ has to be trivial).

Definition 15 (Statements, bases, compatibility). 1. A *statement* is an expression of the shape $a{:}A : \sigma$, where a is a term, A is a type for a, and σ is a predicate; a is called the *subject* of this statement.
2. A *basis* $\mathit{\Gamma}$ is a finite set of statements with only (distinct) term-variables as subject.
3. For a basis $\mathit{\Gamma}$, we say that E *fits into* $\mathit{\Gamma}$, written $E \triangleleft \mathit{\Gamma}$, if $x{:}A{:}\sigma \in \mathit{\Gamma}$ implies $x{:}A \in E$. We write $\overline{\mathit{\Gamma}}$ for the largest context that fits into $\mathit{\Gamma}$.
4. We say that two bases $\mathit{\Gamma}_0, \mathit{\Gamma}_1$ are *compatible* if there exists a context E including all variables occurring in both $\mathit{\Gamma}_0$ and $\mathit{\Gamma}_1$, fitting into both of them.
5. We say that $\mathit{\Gamma}$ *preserves languages* if $\sigma \in \mathcal{L}_{\eta(A)}$ whenever $x{:}A{:}\sigma \in \mathit{\Gamma}$ and η is a type-environment respecting $\overline{\mathit{\Gamma}}$.
6. We extend $\le$ to bases by: $\mathit{\Gamma}' \le \mathit{\Gamma}$ if and only if for every $x{:}A{:}\sigma \in \mathit{\Gamma}$ there exists $x{:}A{:}\sigma' \in \mathit{\Gamma}'$ such that $\sigma' \le \sigma$.

Definition 16 (Predicate Assignment). The *predicate assignment system* to derive judgements of the form $\Gamma \vdash a{:}B{:}\sigma$ where Γ is a basis preserving languages, a a term, A a type and σ a predicate is defined in Figure 3.

Lemma 17. *1. The rules*

$$\dfrac{\Gamma \vdash a{:}A{:}\sigma \quad \sigma \leq \tau}{\Gamma \vdash a{:}A{:}\tau} \quad \textit{and} \quad \dfrac{\Gamma \vdash a{:}A{:}\sigma \quad \sigma \leq \tau \quad \overline{\Gamma} \vdash A <: B}{\Gamma \vdash a{:}B{:}\tau}\,(\tau \in \mathcal{L}_B)$$

are admissible.

2. *If $\overline{\Gamma} \vdash a{:}A$, $\overline{\Gamma} \vdash A <: B$ and $\Gamma \vdash a{:}B{:}\tau$, then there exists $\sigma \in \mathcal{L}(A)$ such that $\sigma \leq \tau$ and $\Gamma \vdash a{:}A{:}\sigma$.*

4 Subject Reduction and Expansion

A minimal requirement for soundness of the assignment system is that predicates are invariant under reduction. This is established through the following result.

Theorem 18 (Subject Reduction). If $\Gamma \vdash a{:}A{:}\rho$, and $a \to a'$, then $\Gamma \vdash a'{:}A{:}\rho$.

Example 19. To better appreciate the importance of this standard result in the present setting, we review an example given in [4].

Suppose that $A \equiv [\ell_0{:}\mathit{Int}, \ell_1{:}\mathit{Int}]$ and $a \equiv [\ell_0 = \varsigma(x^A)1, \ell_1 = \varsigma(x^A)x.\ell_0]$ (using a constant 1 of type *Int*), so that in $\mathsf{FOb}_{1<:\mu}$ we have $\vdash a{:}A$. Then

$$\dfrac{\dfrac{}{x{:}A{:}\omega \vdash 1{:}\mathit{Int}{:}\mathsf{O}} \qquad \dfrac{\dfrac{}{x{:}A{:}\langle\ell_0{:}\omega\to\mathsf{O}\rangle \vdash x{:}A{:}\langle\ell_0{:}\omega\to\mathsf{O}\rangle}\,(\mathsf{Val}\ x) \quad \dfrac{}{x{:}A{:}\langle\ell_0{:}\omega\to\mathsf{O}\rangle \vdash x{:}A{:}\omega}\,(\omega)}{x{:}A{:}\langle\ell_0{:}\omega\to\mathsf{O}\rangle \vdash x.\ell_0{:}\mathit{Int}{:}\mathsf{O}}\,(\mathsf{Val\ Select})}{\vdash a{:}A{:}\langle\ell_0{:}\omega\to\mathsf{O}, \ell_1{:}\langle\ell_0{:}\omega\to\mathsf{O}\rangle\to\mathsf{O}\rangle}\,(\mathsf{Val\ Object}, \wedge I)$$

where ℓ_0 is a field, ℓ_1 is the method $\texttt{get}\ell_0$, and $\mathsf{O} \in \mathcal{L}_{\mathit{Int}}$ is the predicate of being an *odd* integer. Using rules $(\mathsf{Val\ Update}_1)$, $(\mathsf{Val\ Update}_2)$ and $(\wedge I)$ one can derive (the seemingly incorrect):

$$\dfrac{\vdash a{:}A{:}\langle\ell_0{:}\omega\to\mathsf{O}, \ell_1{:}\langle\ell_0{:}\omega\to\mathsf{O}\rangle\to\mathsf{O}\rangle \qquad \dfrac{}{y{:}A{:}\omega \vdash 2{:}\mathit{Int}{:}\mathsf{E}}}{\vdash (a.\ell_0 \Leftarrow \varsigma(y^A)2){:}A{:}\langle\ell_0{:}\omega\to\mathsf{E}, \ell_1{:}\langle\ell_0{:}\omega\to\mathsf{O}\rangle\to\mathsf{O}\rangle}$$

where $\mathsf{E} \in \mathcal{L}_{\mathit{Int}}$ is the predicate of being an *even* integer. This makes sense, however, since it simply states that if the value at ℓ_0 is an odd integer, then the method ℓ_1 will return an odd integer; it also states that this is vacuously true of the actual object $a.\ell_0 \Leftarrow \varsigma(y^A)2$, since it has an even integer at ℓ_0. As a consequence of Theorem 18 we also know that this is harmless: indeed $(a.\ell_0 \Leftarrow \varsigma(y^A)2).\ell_1 \xrightarrow{*} 2$ and we clearly assume that $\nvdash 2{:}\mathit{Int} : \mathsf{O}$, so by contraposition $\nvdash (a.\ell_0 \Leftarrow \varsigma(y^A)2).\ell_1{:}\mathit{Int} : \mathsf{O}$. As a matter of fact, rule (Val Select) is not applicable, since $\nvdash (a.\ell_0 \Leftarrow \varsigma(y^A)2){:}A : \langle\ell_0{:}\omega\to\mathsf{O}\rangle$.

On the other hand, the following odd-looking assignment is legal as well, this time by rule (Val Object) and $(\wedge I)$:

$$\dfrac{\dfrac{}{x{:}A{:}\omega \vdash 1{:}\mathit{Int}{:}\mathsf{O}} \qquad \dfrac{\dfrac{}{x{:}A{:}\langle\ell_0{:}\omega\to\mathsf{E}\rangle \vdash x{:}A{:}\langle\ell_0{:}\omega\to\mathsf{E}\rangle} \qquad \dfrac{}{x{:}A{:}\langle\ell_0{:}\omega\to\mathsf{E}\rangle \vdash x{:}A{:}\omega}}{x{:}A{:}\langle\ell_0{:}\omega\to\mathsf{E}\rangle \vdash (x.\ell_0){:}\mathit{Int}{:}\mathsf{E}}}{a \vdash A{:}\langle \ell_0{:}\omega\to\mathsf{O}, \ell_1{:}\langle\ell_0{:}\omega\to\mathsf{E}\rangle\to\mathsf{E}\rangle}$$

In the last case, however, the apparently odd predicate we deduce is of use to conclude as before:

$$\dfrac{\vdash a{:}A{:}\langle\ell_0{:}\omega\to\mathsf{O}, \ell_1{:}\langle\ell_0{:}\omega\to\mathsf{E}\rangle\to\mathsf{E}\rangle \qquad \dfrac{}{y{:}A{:}\omega \vdash 2{:}\mathit{Int}{:}\mathsf{E}}}{(a.\ell_0 \Leftarrow\varsigma(y^A)2) \vdash A{:}\langle\ell_0{:}\omega\to\mathsf{E}, \ell_1{:}\langle\ell_0{:}\omega\to\mathsf{E}\rangle\to\mathsf{E}\rangle}$$

which is what we expected.

The invariant property of predicates w.r.t. reduction is stronger as they are preserved even by expansion, as is the case for standard intersection type assignment systems (see e.g. [5, 3]). However, we have to be careful, since the simply typed λ-calculus is a subcalculus of $\mathsf{FOb}_{1<:\mu}$, for which it is known that subject expansion does not hold. In fact, we can prove $\vdash (\lambda x^{A\to A}.x)(\lambda x^A.x){:}A\to A{:}\sigma\to\sigma$, but $\Gamma \not\vdash yy\{y \hookleftarrow (\lambda x^C.x)\}{:}A\to A : \sigma\to\sigma$, since there is no way to derive a type for yy for any choice of Γ and C.

But subject expansion does hold for predicates whenever it is the case for types, and this suffices for giving semantics to typed terms consistently with the restriction of convertibility relation to terms of the same type.

Theorem 20 (Subject Expansion). If $\Gamma \vdash a{:}A{:}\tau$, and a' is such that $\overline{\Gamma} \vdash a'{:}A$ and $a' \to a$, then $\Gamma \vdash a'{:}A{:}\tau$.

5 The logical equivalence

The predicate assignment system of Definition 16 induces a logical notion of equivalence, according to which a and b are equal at A if they can be assigned the same set of predicates from $\mathcal{L}_A$. By extending this notion to open terms, we get the following definition.

Definition 21 (Logical Equivalence).

1. Let a and b be terms such that $E \vdash a{:}A$ and $E \vdash b{:}A$; we define
$$[\![a{:}A]\!]_E = \{\sigma \in \mathcal{L}_A \mid \exists\Gamma.\overline{\Gamma} = E \ \&\ \Gamma \vdash a{:}A{:}\sigma\}.$$
2. a and b are *logically equivalent at A and environment E* ($a \simeq^{\mathcal{L}}_E b : A$) if
$$E \vdash a{:}A, E \vdash b{:}A \text{ and } [\![a{:}A]\!]_E = [\![b{:}A]\!]_E.$$

Fig. 4 The equation system $\Delta_{=} \cup \Delta_{=x} \cup \Delta_{=<:} \cup \Delta_{=\to} \cup \Delta_{=\mathsf{Ob}} \cup \Delta_{=\mu}$

(Eval Beta) :
$$\frac{E \vdash \lambda x^{A} b{:}A \to B \quad E \vdash a{:}A}{E \vdash (\lambda x^{A} b)(a) \leftrightarrow b\{x \leftarrow\!\!\dashv a\} : B}$$

(Eq Subsumption) :
$$\frac{E \vdash a \leftrightarrow a' : A \quad E \vdash A <: B}{E \vdash a \leftrightarrow a' : B}$$

(Eq Top) :
$$\frac{E \vdash a{:}A \quad E \vdash b{:}B}{E \vdash a \leftrightarrow b : \mathsf{Top}}$$

(Eq Select) :
$$\frac{E \vdash a \leftrightarrow a' : [\ell_i{:}B_i \;^{i \in I}]}{E \vdash a.\ell_j \leftrightarrow a'.\ell_j : B_j} \; (j \in I)$$

(Eq Update) where $A \equiv [\ell_i{:}B_i \;^{(i \in I)}]$:
$$\frac{E \vdash a \leftrightarrow a' : A \quad E, x{:}A \vdash b \leftrightarrow b' : B_j}{E \vdash a.\ell_j \Leftarrow \varsigma(x^{A})b \leftrightarrow a'.\ell_j \Leftarrow \varsigma(x^{A})b' : A} \; (j \in I)$$

(Eq Sub Object) where $I \cap J = \emptyset,\; A \equiv [\ell_i{:}B_i \;^{i \in I}],\; A' \equiv [\ell_i{:}B_i \;^{i \in I \cup J}]$:
$$\frac{E, x_i{:}A \vdash b_i{:}B_i \; (\forall i \in I) \quad E, x_j{:}A' \vdash b_j{:}B_j \; (\forall j \in J)}{E \vdash [\ell_i = \varsigma(x_i^{A})b_i \;^{(i \in I)}] \leftrightarrow [\ell_i = \varsigma(x_i^{A'})b_i \;^{i \in I \cup J}] : A}$$

(Eval Select) where $I \cap J = \emptyset,\; A \equiv [\ell_i{:}B_i \;^{i \in I}],\; A' \equiv [\ell_i{:}B_i \;^{i \in I \cup J}],\; a \equiv [\ell_i = \varsigma(x_i^{A'})b_i \;^{i \in I}]$:
$$\frac{E \vdash a{:}A}{E \vdash a.\ell_j \leftrightarrow b_j\{x_j \leftarrow\!\!\dashv a\} : B_j} \; (j \in I)$$

(Eval Update) where $I \cap J = \emptyset,\; A \equiv [\ell_i{:}B_i \;^{i \in I}],\; A' \equiv [\ell_i{:}B_i \;^{i \in I \cup J}],\; a \equiv [\ell_i = \varsigma(x_i^{A'})b_i \;^{i \in I}]$:
$$\frac{E \vdash a{:}A \qquad E, x{:}A \vdash b{:}B_j}{E \vdash a.\ell_j \Leftarrow \varsigma(x^{A})b \leftrightarrow [\ell_j = \varsigma(x^{A'})b, \ell_i = \varsigma(x^{A'})b_i \;^{(i \in I \cup J \setminus \{j\})}] : A} \; (j \in J)$$

With respect to the original system [1], we have omitted the obvious rules, like (Eq Appl), as well as the extensionality rules (called (Eval Eta) and (Eval Fold), respectively)

Notice that, if the basis Γ respects languages, the requirement $\sigma \in \mathcal{L}_A$ in the above definition is clearly redundant.

Logical equivalence is the theory of a model built out of predicates, where the denotation of a term is exactly the set of its properties: i.e. the *filter model*. It can be constructed along the lines of [4], even if the type interpretation cannot be the same, because retractions do not model sub-typing. We leave this investigation to further study, and concentrate here on the properties of logical equivalence.

Definition 22. Equivalence among terms of $\mathsf{FOb}_{1<:\mu}$ is defined via a system deriving statements of the shape $a \leftrightarrow b : A$, meaning that terms a and b are equal at type A; the system $\Delta_{=} \cup \Delta_{=x} \cup \Delta_{=<:} \cup \Delta_{=\to} \cup \Delta_{=\mathsf{Ob}} \cup \Delta_{=\mu}$ is shown in Figure 4.

This notion includes (typed) convertibility but it does not coincide with it: in fact, '$\leftrightarrow$' is a congruence whereas ' $\to$ ' is not closed under arbitrary contexts; more importantly, this is a consequence of sub-typing and precisely of rule (Eq Sub Object) (see the next example). Therefore, from the subject reduction and expansion theorems it does not follow that equality implies logical equivalence.

Example 23. Consider the terms (where $A \equiv [\ell_0{:}\mathit{Int}, \ell_1{:}\mathit{Int}]$)

$$a \equiv [\ell_0 = \varsigma(x_1^{A})1, \ell_1 = \varsigma(x_1^{A})1], b \equiv [\ell_0 = \varsigma(x_0^{A})1, \ell_1 = \varsigma(x_1^{A})x.\ell_0].$$

In [1], Section 7.6.2 it is argued that a and b cannot be equated at A. Indeed, they are not logically equivalent at A since, if we assume that 1 is the predicate expressing the property of “being the number 1” (so $1 \in \mathcal{L}_{\mathit{Int}}$, and $\vdash 1{:}\mathit{Int}{:}1$), then $\vdash a{:}A{:}\langle \ell_1{:}\omega \to 1\rangle$ but $\not\vdash b{:}A{:}\langle \ell_1{:}\omega \to 1\rangle$. Indeed (omitting some parts of the derivation for readability):

$$\dfrac{\dfrac{}{x_1{:}A{:}\omega \vdash 1{:}\mathit{Int}{:}1}}{\vdash a{:}A{:}\langle \ell_1{:}\omega{\to}1\rangle}\ (\mathsf{Val\ Object})$$

Replacing a by b would not yield a valid derivation. The best we can do for b is instead:

$$\dfrac{\dfrac{\dfrac{}{x_1{:}A{:}\langle \ell_0{:}\omega{\to}1\rangle \vdash x_1{:}A{:}\langle \ell_0{:}\omega{\to}1\rangle} \qquad \dfrac{}{x_1{:}A{:}\langle \ell_0{:}\omega{\to}1\rangle \vdash x_1{:}A{:}\omega}}{x_1{:}A{:}\langle \ell_0{:}\omega{\to}1\rangle \vdash x_1.\ell_0{:}\mathit{Int}{:}1}\ (\mathsf{Val\ Select})}{\vdash b{:}A{:}\langle \ell_1{:}\langle \ell_0{:}\omega{\to}1\rangle{\to}1\rangle}\ (\mathsf{Val\ Object})$$

To express this in natural language, what we have proven is that the value of a on calling method ℓ_1 is 1, and that this is a “field”, in that it does not depend on other parts of a; on the other hand, for b the value returned by ℓ_1 depends on the actual value of ℓ_0 in b: the predicate $\langle \ell_1{:}\langle \ell_0{:}\omega{\to}1\rangle{\to}1\rangle$ expresses this.

However, in [1] paragraph 8.4.2 is observed that the equality $\vdash a \leftrightarrow b : [\ell_0{:}\mathit{Int}]$ is derivable since both

$$\vdash [\ell_0 = \varsigma(x_0^B)1] \leftrightarrow a : [\ell_0{:}\mathit{Int}] \quad \text{and} \quad \vdash [\ell_0 = \varsigma(x_0^B)1] \leftrightarrow b : [\ell_0{:}\mathit{Int}]$$

can be obtained by rule ($\mathsf{Eq\ Sub\ Object}$); this clearly shows that ‘$\leftrightarrow$’ is not convertibility, since a, b and $[\ell_0 = \varsigma(x_0^B)1]$ are distinct normal forms and the reduction is confluent.

In our setting, we can show that $a \simeq^{\mathcal{L}}_{\emptyset} b : [\ell_0{:}\mathit{Int}]$ as well, and this is the effect of restricting to the language $\mathcal{L}_{[\ell_0:\mathit{Int}]}$; in fact, the only non-trivial predicates in $\mathcal{L}_{[\ell_0:\mathit{Int}]}$ that we can derive for either a or b are $\langle \ell_0{:}\omega{\to}1\rangle$ (or greater than this w.r.t. $\leq$).

Theorem 14 is first evidence of the consistency of the predicate assignment system with respect to the sub-typing relation. It is however not enough, and we need to establish the following.

Corollary 24. If $a \simeq^{\mathcal{L}}_{E} b : A$ and $E \vdash A <: B$ then $a \simeq^{\mathcal{L}}_{E} b : B$.

We conclude this section by showing that equality in $\mathsf{FOb}_{1<:\mu}$ system implies logical equivalence, proving that what we have seen in the Example 23 actually holds in general.

Theorem 25. *If $E \vdash a \leftrightarrow b : A$ then $a \simeq^{\mathcal{L}}_{E} b : A$.*

6 Observational semantics and adequacy

Observational semantics for $\mathsf{FOb}_{1<:\mu}$ has been defined in [11] in Morris-style, called there “contextual equivalence”. In the same paper it has been shown that this coincides with a notion of bisimulation which is stronger than '$\leftrightarrow$'. We will adopt a slightly more general definition (we will write a^A for a closed term a such that $\vdash a{:}A$).

Definition 26 (Convergence). Given any (well formed) closed term a^A, it *converges* to value v ($a \Downarrow v$), if $a \xrightarrow{*} v$. Moreover, a^A is *convergent* ($a \Downarrow$) if there exists a value v such that $a \Downarrow v$, and is *divergent* ($a \Uparrow$) if not $a \Downarrow$)

We will write $\_{:}A \vdash C[\_]{:}B$ to express that the closed context $C[\_]$ is well typed with type B, under the assumption that the "hole $\_$" has type A; $C[a]$ is the result of replacing '$\_$' by a in $C[\_]$.

Definition 27 (Observational Equivalence). Two closed terms a and b are called *observationally equivalent at type A*, written $a \simeq_A^{\mathcal{O}} b$, if both a^A and b^A, and for any ground type K and value v_K it is the case:

$$\forall C[\_].(\ \_{:}A \vdash C[\_]{:}K \ \Rightarrow\ C[a] \Downarrow v \Longleftrightarrow C[b] \Downarrow v.$$

This differs from the definition of contextual equivalence in [11] in some respect. First, we consider contexts of any ground type as an "experiment"; moreover, we do not consider reduction rules for constants as "if then else"; as a consequence we cannot discriminate between different constants like **true** and **false**. It is for that reason that we use in Definition 21 the predicate $a \Downarrow v$ instead of $a \Downarrow$.

We claim that, when restricted to closed terms, logical equivalence is included in observational equivalence. To this aim we establish an adequacy result of the logical semantics w.r.t. convergence, by means of a realizability interpretation of predicates, proving that the characterisation results of [9] are preserved in the typed context of the calculus $\mathsf{FOb}_{1<:\mu}$.

Definition 28. The set of labels of A is defined as *Label*$(A) = \{\ell_i \mid i \in I\}$ only for $A \equiv [\ell_i{:}A_i\ ^{(i \in I)}]$; it is empty in all other cases.

If a^A for some object type A, $\ell_j \in$ *Label*(A) and $a \Downarrow [\ell_i = \varsigma(x_i^A)b_i\ ^{(i \in I)}]$, then, for any c^A, $a.\ell(c)$ abbreviates $b_j\{x_j \leftarrow c\}$.

Definition 29 (Realizability Interpretation). The *realizability interpretation* of the predicate σ is a set $[\![\sigma]\!]$ of closed terms defined by induction over the structure of predicates as follows:

1. $[\![\kappa]\!] = \{a^K \mid \kappa \in \mathcal{L}_K\}$,
2. $[\![\sigma{\to}\tau]\!] = \{a^{A\to B} \mid \exists x, b.\ a \Downarrow (\lambda x^A.b)\ \&\ \forall c^A \in [\![\sigma]\!].\ b\{x \leftarrow c\} \in [\![\tau]\!]\}$,
3. $[\![\langle \ell{:}\sigma{\to}\tau\rangle]\!] = \{a^A \mid a \Downarrow\ \&\ \ell \in \textit{Label}(A)\ \&\ \forall c^A \in [\![\sigma]\!].\ a.\ell(c) \in [\![\tau]\!]\}$,
4. $[\![\mu(\sigma)]\!] = \{a^{\mu X.A} \mid a \xrightarrow{*} \mathsf{fold}(\mu X.A, b)\ \&\ b_{A\{X \leftarrow \mu X.A\}} \in [\![\sigma]\!]\}$,
5. $[\![\omega]\!] = \{a^A \mid A \text{ is a type}\}$,
6. $[\![\sigma\wedge\tau]\!] = [\![\sigma]\!] \cap [\![\tau]\!]$.

The next Lemma states that, for any σ, $[\![\sigma]\!]$ is closed under reduction and expansion.

Lemma 30. *If $a^A \in [\![\sigma]\!]$ then for any b^A, if $a \xrightarrow{*} b$ or $b \xrightarrow{*} a$ then $b^A \in [\![\sigma]\!]$.*

Lemma 31. *If $\sigma \leq \tau$ then $[\![\sigma]\!] \subseteq [\![\tau]\!]$.*

Theorem 32 (Realizability theorem). Let ϑ be any , and $a\vartheta$ be the effect of applying ϑ to a (with usual conventions to avoid free and bound variable clashes) . If $\Gamma \vdash a{:}A : \sigma$ and for all $x{:}B{:}\tau \in \Gamma$ it is the case that $\vartheta(x) \in [\![\tau]\!]$, then $a\vartheta \in [\![\sigma]\!]$.

It is easily seen that values v can be assigned non-trivial predicates, so that $a \Downarrow v$ implies that the same predicates can be derived for a because of Theorem 20; on the other hand a straightforward induction shows that if σ is non-trivial, then any $a^A \in [\![\sigma]\!]$ converges: by this and Theorem 32 we obtain a proof of the following corollary.

Corollary 33 (Characterization of convergence). Let a^A be any closed term: then $a \Downarrow$ if and only if $\vdash a{:}A : \sigma$ for some non-trivial σ.

Theorem 34 (Logical Equivalence and Observational Equivalence). Suppose that for any value v of ground type K we have exactly one non-trivial predicate $\kappa_v \in \mathcal{L}_K$, that these predicates are distinct for different values and that $\vdash v{:}K : \kappa_v$ is assumed for each v. Then for any a^A and b_B, if $a \simeq^{\mathcal{L}} b : A$ then $a \simeq^{\mathcal{O}}_A b$.

7 Concluding remarks

By using bisimulation and its coincidence with observational equivalence, in [11] was shown that, taking a and b as in Example 23, $a \simeq^{\mathcal{O}}_{[\ell_1:\mathit{Int}]} b$. This is intuitively clear: the only way to separate a from b is to change the value of ℓ_0, since then the fact that $b.\ell_1$ depends on such a value while $a.\ell_1$ does not, becomes apparent; but the overriding of ℓ_0 is inhibited in contexts with the hole of type $[\ell_1{:}\mathit{Int}]$, where ℓ_0 is hidden.

It is not true, however, that $a \simeq^{\mathcal{L}} b : [\ell_1{:}\mathit{Int}]$, because the predicate $\langle \ell_1{:}\omega \to 1\rangle$ is in $\mathcal{L}_{[\ell_1:\mathit{Int}]}$, it is derivable for a even at type $[\ell_1{:}\mathit{Int}]$ but cannot be derived for b at any type.

That language inclusion is not sufficient to account for sub-typing of object types, while it is for record types (see [10]) is the essential reason for the presence of rule $(\textsf{Val Select})$ in our system. It is reasonable to think that the failure of equivalencies like $a \simeq^{\mathcal{L}} b : [\ell_1{:}\mathit{Int}]$ from Example 23 depends on the fact that no rule accounts for the hiding effect of sub-typing in the case of object types. One possibility for coping with such a limitation is the following rule:

$$\frac{\begin{array}{c} I \cap J = \emptyset,\ A \equiv [\ell_i{:}B_i\ ^{i \in I \cup J}],\ A' \equiv [\ell_i{:}B_i\ ^{i \in J}],\ \langle \ell{:}\tau \to \rho\rangle \in \mathcal{L}_{A'} : \\ \Gamma \vdash a{:}A : \langle \ell{:}\langle \ell_i{:}\sigma_i\ ^{i \in I}\rangle\rangle \wedge \tau \to \rho \quad \Gamma \vdash a{:}A : \langle \ell_i{:}\sigma_i\ ^{i \in I}\rangle \end{array}}{\Gamma \vdash a{:}A' : \langle \ell{:}\tau \to \rho\rangle}$$

This rule formalises the idea that when $A <: A'$ and A and A' are object types, the methods of any object of type A not mentioned in A' are hidden: therefore if a satisfies the premise of any arrow predicate concerning the hidden part, this will never change in contexts of type A', in such a way that the latter premise can be discharged. Clearly, with reference to Example 23, by this rule one can derive $\vdash b{:}[\ell_1{:}\mathit{Int}] : \langle \ell_1{:}\omega \to 1\rangle$, which makes a and b logically indiscernible at type $[\ell_1{:}\mathit{Int}]$.

The soundness with respect to observational equivalence of the system resulting by adding such a rule to the predicate assignment system can be proved by means of a modified realizability interpretation of predicates, but at the time of writing we do not know to what extent it actually solves the problem.

References

1. M. Abadi and L. Cardelli. *A Theory of Objects*. Springer, 1996.
2. S. Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 51:1–77, 1991.
3. S. van Bakel. Intersection Type Assignment Systems. *Theoretical Computer Science*, 151(2):385–435, 1995.
4. S. van Bakel and U. de'Liguoro. Logical semantics of the first order sigma-calculus. *Lecture Notes in Computer Science*, 2841:202–215, 2003.
5. H. P. Barendregt, M. Coppo, and M. Dezani. A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic*, 48:931–940, 1983.
6. V. Breazu-Tannen, T. Coquand, C. A. Gunter, and A. Scedrov. Inheritance as implicit coercion. *Information and Computation*, 93:172–221, 1991.
7. K. B. Bruce and G. Longo. A modest model of records, inheritance and bounded quantification. *Information and Computation*, 87:196–240, 1990.
8. K. B. Bruce and J. C. Mitchell. Per models of subtyping, recursive types and higher-order polymorphism. In *Proc. of POPL*, 1992.
9. U. de'Liguoro. Characterizing convergent terms in object calculi via intersection types. *Lecture Notes in Computer Science*, 2004:315–328, 2001.
10. U. de'Liguoro. Subtyping in logical form. In *ITRS'02*, ENTCS 70. Elsevier, 2002.
11. A. Gordon and G. Rees. Bisimilarity for first-order calculus of objects with subtyping. In *Proc. of POPL'96*, pages 386–395, 1996.
12. J. C. Mitchell. *Foundations for Programming Languages*. MIT Press, 1996.

A Some proofs

We will conclude this paper by giving details of proofs of some of the results obtained in this paper; we will normally only state the non-trivial issues.

The following Lemma is needed in the proof of Lemma 40, and states what can be concluded from derivable type-statements. The list should in fact be longer, but almost all are directly implied in Lemma 37, using part 37.1; we have listed a few as illustration of that fact.

Lemma 35 (Type generation lemma). 1. If $E \vdash [\ell_i = \varsigma(x_i^{A_i})b_i \ ^{(i \in I)}]{:}A$, then $A_i = A_j$, for all $1 \leq i, j \leq n$, and $A_1 <: A$.
2. If $a = [\ell_i = \varsigma(x_i^{A})b_j \ ^{(i \in I)}]$ and $E \vdash a{:}C$ for some C, then $A <: C$ and $E \vdash a{:}A$.
3. If $a = [\ell_i = \varsigma(x_i^{A})b_j \ ^{(i \in I)}].\ell$, and $E \vdash a{:}C$ for some C, then $\ell = \ell_j$ for some $j \in I$, $A_j <: C$, and $E \vdash a{:}A$ where $A = [\ell_i{:}A_i \ ^{(i \in I)}]$.

$\vdots$

Lemma 36. *If* $E, x{:}A \vdash b{:}B$, *and* $\Gamma \vdash a{:}A$, *then* $E \vdash b\{x \leftarrow a\}{:}B$.

The essential properties of the predicate assignment system, on which the subsequent treatment relies, are stated in next Lemma.

Lemma 37 (Generation lemma). Let $\tau \in \mathcal{P}_S$.

1. If $\Gamma \vdash a{:}B : \tau$, then $\overline{\Gamma} \vdash a{:}B$, and these derivations have the same structure.

2. If $\Gamma, x{:}A{:}\sigma \vdash a{:}B : \tau$, and $C <: A$, then also $\Gamma, x{:}C{:}\sigma \vdash a{:}B : \tau$.
3. If $\Gamma \vdash [\ell_i = \varsigma(x_i^A)b_i \;^{(i \in I)}]{:}B : \tau$, then $A <: B$, $\tau = \langle \ell_j{:}\rho \to \mu \rangle$ for some $j \in I$, $\overline{\Gamma} \vdash [\ell_i = \varsigma(x_i^A)b_i \;^{(i \in I)}]{:}A$ and $\Gamma, x_j{:}A{:}\rho \vdash b_j{:}A_j : \tau$, where $A = [\ell_i{:}A_i \;^{(i \in I)}]$.
4. If $\Gamma \vdash a.\ell{:}B : \tau$, then there exists σ, $A = [\ell_i{:}A_i \;^{(i \in I)}]$, such that $\ell = \ell_j$ for some $j \in I$, $A_j <: B$, $\Gamma \vdash a{:}A : \langle \ell_j{:}\sigma \to \tau \rangle$ and $\Gamma \vdash a{:}A : \sigma$.
5. If $\Gamma \vdash a.\ell \Leftarrow \varsigma(y^A)b{:}B : \tau$, then $A <: B$, $\tau = \langle \ell_j{:}\rho \to \mu \rangle$ for some $j \in I$, $\overline{\Gamma} \vdash a{:}A$, and $\Gamma, y{:}A{:}\rho \vdash b{:}A_j : \mu$, where $A = [\ell_i{:}A_i \;^{(i \in I)}]$.
6. If $\Gamma \vdash \lambda x^C.a{:}B : \tau$, then there exists ρ, μ, D such that $\tau = \rho \to \mu$, $\Gamma, x{:}C{:}\rho \vdash a{:}D : \mu$ and $C \to D <: B$.
7. If $\Gamma \vdash a(b){:}B : \tau$, then there exists $\sigma, C, A <: B$ such that $\Gamma \vdash a{:}C \to A : \sigma \to \tau$ and $\Gamma \vdash b{:}C : \sigma$.
8. If $\Gamma \vdash \mathsf{fold}(X, a){:}A : \sigma$, then there exist B, τ such that $\mu X.B <: A$, $\sigma = \mu(\tau)$, and $\Gamma \vdash a{:}B\{X \hookleftarrow \mu X.B\}$.
9. If $\Gamma \vdash \mathsf{unfold}(a){:}A : \sigma$, then exist X, B such that $B\{X \hookleftarrow \mu X.B\} <: A$, and $\Gamma \vdash a{:}\mu X.B : \mu(\sigma)$.

Proof. Straightforward. ∎

Lemma 38 (Substitution lemma). If $\Gamma, x{:}A{:}\sigma \vdash b{:}B : \tau$ and $\Gamma \vdash a{:}A : \sigma$, then $\Gamma \vdash b\{x \hookleftarrow a\}{:}B : \tau$.

Proof: By straightforward induction on the structure of derivations, of which we show only the interesting cases.

$(\mathsf{Val}\ x)$: Then either:

$(b = x)$: Then $\sigma \leq \tau$. Since $x\{x \hookleftarrow a\} = a$, the result then follows from the second assumption and Lemma 1.

$(b = y \neq x)$: Since $\Gamma, x{:}A{:}\sigma \vdash y{:}B : \tau$, and $x \notin \mathit{FV}(y)$, also $\Gamma \vdash y{:}B : \tau$.

(ω) : Then $\overline{\Gamma}, x{:}A \vdash b{:}B$. By Lemma 36, $\overline{\Gamma} \vdash b\{x \hookleftarrow a\}{:}B$, and by rule (ω), $\Gamma \vdash b\{x \hookleftarrow a\}{:}B : \omega$.

$(\wedge I)$: Then $\tau = \wedge_{\underline{n}} \tau_i$, and, for $1 \leq i \leq n$, $\Gamma, x{:}A{:}\sigma \vdash b{:}\tau_i$. By induction, $\Gamma \vdash b\{x \hookleftarrow A\}{:}\tau_i$, and, by rule $(\wedge I)$, $\Gamma \vdash b\{x \hookleftarrow a\}{:}\wedge_{\underline{n}} \tau_i$. ∎

We use this Lemma to show the following result.

Theorem 39 (Subject Reduction). If $\Gamma \vdash a{:}A : \rho$, and $a \;\to\; a'$, then $\Gamma \vdash a'{:}A : \rho$.

Proof: By induction on the definition of the reduction relation $\;\to\;$. We only show one case, that does not depend on Lemma 38; the others follow easily. Assume $\rho \in \mathcal{P}_{\mathrm{S}}$.

1. $[\ell_i = \varsigma(x_i^C)b_i \;^{(i \in I)}].\ell_j \Leftarrow \varsigma(y^B)b \;\to\; [\ell_i = \varsigma(x_i^C){b_i}^{i \in I \backslash j}, \ell_j = \varsigma(y^C)b]$. Let $C = [\ell_i{:}C_i \;^{(i \in I)}]$. By Lemma 37, there exists $B <: A$, $\tau = \langle \ell_j{:}\rho \to \mu \rangle$,

$$\overline{\Gamma} \vdash [\ell_i = \varsigma(x_i^C)b_i \;^{(i \in I)}]{:}B, \text{ and } \Gamma, y{:}B{:}\rho \vdash b{:}B_j : \mu,$$

for some $j \in I$, where $B = [\ell_i{:}B_i \;^{(i \in I)}]$. By Lemma 35, we have $C <: B$ and

$$\frac{\overline{\Gamma}, x_j{:}C \vdash b_j{:}C_j \quad (\forall j \in J)}{\overline{\Gamma} \vdash [\ell_i = \varsigma(x_j^C)b_j \;^{(j \in J)}]{:}C}$$

where $C = [\ell_j{:}C_j\ ^{(j\,\in\,J)}]$; notice that $I \subseteq J$. Notice that, by Lemma 37, there exists a derivation D'' such that $D'' :: \Gamma, y{:}C{:}\sigma \vdash b{:}B_j : \tau$ and $\overline{D''} :: \overline{\Gamma}, y{:}C \vdash b{:}B_j$. We can then construct:

$$\dfrac{\dfrac{\begin{array}{c} \\ \overline{\Gamma}, x_i{:}C \vdash b_i{:}B_i \quad (\forall i\,{\in}\, I\backslash j)\end{array} \qquad \begin{array}{c}\overline{D''} \\ \overline{\Gamma}, y{:}C \vdash b{:}B_j\end{array}}{\overline{\Gamma} \vdash a'{:}C} \qquad \begin{array}{c} D'' \\ \Gamma, y{:}C{:}\sigma \vdash b{:}B_j : \tau\end{array}}{\dfrac{\Gamma \vdash a'{:}C : \sigma{\to}\tau}{\Gamma \vdash a'{:}A : \sigma{\to}\tau}\,(C\,{<:}\,A)}$$

(where $a' = [\ell_i = \varsigma(x_i^C)b_i{}^{i\,\in\,I\backslash j}, \ell_j = \varsigma(y^C)b]$). ■

Lemma 40 (Expansion lemma). If $\Gamma \vdash b\{x \hookleftarrow a\}{:}B : \tau$, and both $\overline{\Gamma}, x{:}A \vdash b{:}B$ and $\overline{\Gamma} \vdash a{:}A$ for some A, then there exist σ such that $\Gamma, x{:}A{:}\sigma \vdash b{:}B : \tau$ and $\Gamma \vdash a{:}A : \sigma$.

Proof: By induction on the structure of terms; we only show some interesting cases. Let $B = [\ell_k{:}B_i\ ^{(k\,\in\,I)}]$, and assume $\tau \,{\in}\, \mathcal{P}_{\mathrm{S}}$.

$(b = y \neq x)$: Since $y\{x \hookleftarrow a\} = y$, we get $\Gamma \vdash y{:}B : \tau$, and, by Weakening, $\Gamma, x{:}A{:}\omega \vdash y{:}B : \tau$. Notice that, from the fact that $\overline{\Gamma} \vdash a{:}A$, we get, by rule (ω), $\Gamma \vdash a{:}A : \omega$.

$(b = c.\ell \Leftarrow \varsigma(y^C)d)$: If $\Gamma \vdash (c.\ell \Leftarrow \varsigma(y^C)d)\{x \hookleftarrow a\}{:}B : \tau$ then by definition of substitution, $\Gamma \vdash c\{x \hookleftarrow a\}.\ell \Leftarrow \varsigma(y^C)d\{x \hookleftarrow a\}{:}B : \tau$. By Lemma 37 this implies $\overline{\Gamma} \vdash c\{x \hookleftarrow a\}.\ell \Leftarrow \varsigma(y^C)d\{x \hookleftarrow a\}{:}B$, so, by Lemma 35,

$$C \,{<:}\, B \text{ and both } \overline{\Gamma} \vdash c\{x \hookleftarrow a\}.\ell \Leftarrow \varsigma(y^C)d\{x \hookleftarrow a\}{:}C \text{ and } \overline{\Gamma} \vdash c\{x \hookleftarrow a\}{:}C,$$

and by Lemma 37, $\Gamma \vdash c\{x \hookleftarrow a\}.\ell \Leftarrow \varsigma(y^C)d\{x \hookleftarrow a\}{:}C : \tau$. Then by rule (Val Update), there are ρ, μ such that $\tau = \langle \ell_j{:}\rho{\to}\mu\rangle$ (so $\ell = \ell_j$), and

$$\overline{\Gamma} \vdash c\{x \hookleftarrow a\}{:}C, \text{ and } \Gamma, y{:}C{:}\rho \vdash d\{x \hookleftarrow a\}{:}C_j : \mu.$$

where $C = [C_i\ ^{(i\,\in\,I)}]$. Then, by induction, there exist σ such that

$$\Gamma \vdash a{:}A : \sigma, \text{ and } \Gamma, x{:}A{:}\sigma, y{:}C{:}\rho \vdash d{:}C_j : \mu.$$

By assumption, $\overline{\Gamma}, x{:}A \vdash c.\ell \Leftarrow \varsigma(y^C)d{:}B$, so, by Lemma 35, also $\overline{\Gamma}, x{:}A \vdash c{:}C$. Then, by rule (Val Update),

$$\Gamma, x{:}A \vdash c.\ell_k \Leftarrow \varsigma(y^C)b{:}C : \tau$$

and $\Gamma, x{:}A \vdash c.\ell_k \Leftarrow \varsigma(y^C)b{:}B : \tau$ follows from rule $({<:})$.

$(b = c(d))$: If $\Gamma \vdash (c(d))\{x \hookleftarrow a\}{:}B : \tau$, then $\Gamma \vdash c\{x \hookleftarrow a\}(d\{x \hookleftarrow a\}){:}B : \tau$, and by Lemma 37 there exists $\rho, C, A \,{<:}\, B$ such that $\Gamma \vdash c\{x \hookleftarrow a\}{:}C{\to}A : \rho{\to}\tau$ and $\Gamma \vdash d\{x \hookleftarrow a\}{:}C : \sigma$. Since by assumption $\overline{\Gamma}, x{:}A \vdash c(d){:}B$, by Lemma 35, $\overline{\Gamma}, x{:}A \vdash c{:}C{\to}A$ and $\overline{\Gamma}, x{:}A \vdash d{:}C$. Then, by induction, there exists σ_1, σ_2 such that $\Gamma, x{:}A{:}\sigma_1 \vdash c{:}C{\to}A : \rho{\to}\tau$ and $\Gamma \vdash a{:}A : \sigma_1$, and $\Gamma, x{:}A{:}\sigma_2 \vdash d{:}C : \rho$ and $\Gamma \vdash a{:}A : \sigma_2$. Then by Weakening and rule (Val Appl) we get $\Gamma, x{:}A{:}\sigma_1{\wedge}\sigma_2 \vdash c(d){:}A : \tau$ and by rule $(\wedge I)$, $\Gamma \vdash a{:}A : \sigma_1{\wedge}\sigma_2$. ■

Theorem 41 (Subject Expansion). If $\Gamma \vdash a{:}A{:}\tau$, and a' is such that $\overline{\Gamma} \vdash a'{:}A$ and $a' \rightarrow a$, then $\Gamma \vdash a'{:}A{:}\tau$.

Proof: By induction on the definition of the reduction relation $\rightarrow$. We only show one case, that does not depend on Lemma 40; assume $\tau \in \mathcal{P}_{\mathrm{S}}$.

1. $[\ell_i = \varsigma(x_i^C)b_i \;^{(i\in I)}].\ell_j \Leftarrow \varsigma(y^B)b \rightarrow [\ell_i = \varsigma(x_i^C)b_i{}^{i\in I\backslash j}, \ell_j = \varsigma(y^C)b]$. If

$$\Gamma \vdash [\ell_i = \varsigma(x_i^C)b_i{}^{i\in I\backslash j}, \ell_j = \varsigma(y^C)b]{:}A{:}\tau$$

then, by Lemma 37, $C <: A$, $\tau = \langle \ell_j{:}\rho \rightarrow \mu \rangle$,
$\overline{\Gamma} \vdash [\ell_i = \varsigma(x_i^C)b_i{}^{i\in I\backslash j}, \ell_j = \varsigma(y^C)b]{:}C$ and $D_j :: \Gamma, x_j{:}C{:}\rho \vdash b_j{:}C_j{:}\tau$, for some $j \in I$, where $C = [\ell_i{:}C_i \;^{(i\in I)}]$.
We have assumed $\overline{\Gamma} \vdash [\ell_i = \varsigma(x_i^C)b_i \;^{(i\in I)}].\ell_j \Leftarrow \varsigma(y^B)b{:}A$, which gives, by Lemma 35, $C <: B <: A$, and $D :: \overline{\Gamma} \vdash [\ell_i = \varsigma(x_i^C)b_i \;^{(i\in I)}]{:}C$
We can now construct:

$$\frac{\begin{array}{c} D \\ \overline{\Gamma} \vdash [\ell_i = \varsigma(x_i^C)b_i \;^{(i\in I)}]{:}C \end{array} \qquad \begin{array}{c} D_j \\ \Gamma, x_j{:}C{:}\rho \vdash b_j{:}C_j{:}\tau \end{array}}{\Gamma \vdash (\; [\ell_i = \varsigma(x_i^C)b_i \;^{(i\in I)}].\ell_j \Leftarrow \varsigma(x^C)b \;){:}C{:}\rho \rightarrow \tau}$$

and the desired result $\Gamma \vdash (\; [\ell_i = \varsigma(x_i^C)b_i \;^{(i\in I)}].\ell_j \Leftarrow \varsigma(x^C)b \;){:}A{:}\rho \rightarrow \tau$ then follows by applying rule $(<:)$.

For $\tau = \wedge_{\underline{n}}\tau_i \; (n \geq 0)$, the proof follows by easy induction. ■

Theorem 42. *If* $E \vdash a \leftrightarrow b : A$ *then* $a \simeq_E^{\mathcal{L}} b : A$.

Proof. By structural induction over the derivation of $E \vdash a \leftrightarrow b : A$. Most of the cases are the same as in the proofs of Theorem 18 and 20. Case (Eq Subsumption) follows by Corollary 24. We only show:

(Eq Sub Object) : Then $I \cap J = \emptyset$, $A \equiv [\ell_i{:}B_i \;^{i\in I}]$, $A' \equiv [\ell_i{:}B_i \;^{i\in I\cup J}]$, and

$$\frac{E, x_i{:}A \vdash b_i{:}B_i \; (\forall i \in I) \quad E, x_j{:}A' \vdash b_j{:}B_j \; (\forall j \in J)}{E \vdash [\ell_i = \varsigma(x_i^A)b_i \;^{i\in I}] \leftrightarrow [\ell_i = \varsigma(x_i^{A'})b_i \;^{i\in I\cup J}] : A}$$

Now, if $\sigma \in [\![a'{:}A]\!]_E$, where $a' \equiv [\ell_i = \varsigma(x_i^{A'})b_i \;^{i\in I\cup J}]$, then for some Γ such that $\overline{\Gamma} = E$, we derive $\Gamma \vdash a'{:}A{:}\sigma$; this implies, by Lemma 37, that $\sigma = \langle \ell_k{:}\tau \rightarrow \rho \rangle \in \mathcal{L}_A$ for certain τ, ρ and $k \in I \cup J$ and that $\Gamma, x_k{:}A{:}\tau \vdash b_k{:}B_k{:}\rho$.
Now either $k \in I$ or $k \in J$: in the first case by rule (Val Object) we derive immediately that $\sigma \in [\![a{:}A]\!]_E$, where $a \equiv [\ell_i = \varsigma(x_i^A)b_i \;^{i\in I}]$. On the other hand, the case $k \in J$, namely $k \notin I$, is impossible, since then $\langle \ell_k{:}\tau \rightarrow \rho \rangle \notin \mathcal{L}_A$.
This proves that $[\![a'{:}A]\!]_E \subseteq [\![a{:}A]\!]_E$: the proof of the opposite inclusion is similar and easier. ■

Let ϑ be any closed substitution, and $a\vartheta$ be the effect of applying ϑ to a (with usual conventions to avoid free and bound variable clashes).

Given a closed substitution ϑ we say that it respects Γ if for all $x{:}B{:}\tau \in \Gamma$ it is the case that $\vartheta(x) \in [\![\tau]\!]$. By $\vartheta[x_j := c]$ we mean the same as ϑ but for substituting x_j by c.

Theorem 43 (Realizability theorem). If $\Gamma \vdash a{:}A{:}\sigma$ and for all $x{:}B{:}\tau \in \Gamma$ it is the case that $\vartheta(x) \in [\![\tau]\!]$, then $a\vartheta \in [\![\sigma]\!]$.

Proof. By induction on the derivation of $\Gamma \vdash a{:}A{:}\sigma$. We only show the interesting cases.

(unfold) : Then the derivation ends with

$$\frac{\Gamma \vdash a{:}\mu X.A{:}\mu(\sigma)}{\Gamma \vdash \mathsf{unfold}(a){:}A\{X \hookleftarrow \mu X.A\}{:}\sigma}\ (\sigma \in \mathcal{P}_\mathrm{s})$$

By induction $a \xrightarrow{*} \mathsf{fold}(\mu X.A, b)$ for some $b \in [\![\sigma]\!]$; since $\mathsf{unfold}(a) \xrightarrow{*} b$, we are done by Lemma 30.

$(\mathsf{Val\ Object})$**:** The derivation ends with

$$\frac{\overline{\Gamma}, x_i{:}A \vdash b_i{:}B_i\ (\forall i \in I) \quad \Gamma, x_j{:}A{:}\sigma \vdash b_j{:}B_j{:}\tau}{\Gamma \vdash [\ell_i = \varsigma(x_i^A)b_i\ ^{(i \in I)}]{:}A{:}\langle \ell_j{:}\sigma \rightarrow \tau\rangle}\ (j \in I)$$

Then $a \equiv [\ell_i = \varsigma(x_i^A)b_i\ ^{(i \in I)}]$ which is a value; since substitutions preserve values, we get $a\vartheta \Downarrow$. That $\ell_j \in \mathit{Label}(A)$ follows from the side-condition of the rule. For any $c^A \in [\![\sigma]\!]$ we have that $\vartheta[x_j := c]$ respects $\Gamma, x_j{:}A{:}\sigma$ and

$$a\vartheta.\ell_j(c) \equiv b(\vartheta[x_j := c]) \in [\![\tau]\!]$$

follows by induction.

$(\mathsf{Val\ Select})$: The derivation ends with

$$\frac{\Gamma \vdash a{:}A{:}\langle \ell_j{:}\sigma \rightarrow \tau\rangle \quad \Gamma \vdash a{:}A{:}\sigma}{\Gamma \vdash a.\ell_j{:}B_j{:}\tau}$$

By induction $a\vartheta \Downarrow v$, for some value v, $\ell_j \in \mathit{Label}(A)$ and $a\vartheta.\ell_j(c) \in [\![\tau]\!]$ for any $c^A \in [\![\sigma]\!]$; since $a\vartheta \in [\![\sigma]\!]$ (by induction again) we have that $v \in [\![\sigma]\!]$ by Lemma 30 (first part) so that:

$$(a.\ell_j)\vartheta \xrightarrow{*} a\vartheta.\ell_j(v) \in [\![\tau]\!],$$

and we conclude by Lemma 30 (second part).

$(\mathsf{Val\ Update}_1)$: The derivation ends with

$$\frac{\Gamma \vdash a{:}A{:}\langle \ell_j{:}\omega \rightarrow \rho\rangle \quad \Gamma, y{:}A{:}\sigma \vdash b{:}B_j{:}\tau}{\Gamma \vdash (a.\ell_j \Leftarrow \varsigma(y^A)b){:}A{:}\langle \ell_j{:}\sigma \rightarrow \tau\rangle}$$

By induction $a\vartheta \in [\![\langle \ell_j{:}\omega \rightarrow \rho\rangle]\!]$, which implies that $a\vartheta \Downarrow$ and that $\ell_j \in \mathit{Label}(A)$, therefore $(a.\ell_j \Leftarrow \varsigma(y^A)b)\vartheta \Downarrow$ as well. Given any $c^A \in [\![\sigma]\!]$, $\vartheta[y := c]$ respects $\Gamma, y{:}A{:}\sigma$, so that we conclude by induction

$$(a.\ell_j \Leftarrow \varsigma(y^A)b)\vartheta.\ell_j(c) \equiv b\vartheta[y := c] \in [\![\tau]\!].$$

$(\mathsf{Val\ Update}_2)$: the last inference is an instance of the rule:

$$\frac{\overline{\Gamma} \vdash (a.\ell_j \Leftarrow \varsigma(y^A)b){:}A \quad \Gamma \vdash a{:}A : \langle \ell_i{:}\sigma \to \tau \rangle \quad i \neq j}{\Gamma \vdash (a.\ell_j \Leftarrow \varsigma(y^A)b){:}A : \langle \ell_i{:}\sigma \to \tau \rangle}$$

By induction we know that $a\vartheta \in [\![\langle \ell_j{:}\omega \to \rho \rangle]\!]$, which implies that $a\vartheta \Downarrow$: hence $(a.\ell_j \Leftarrow \varsigma(y^A)b)\vartheta \Downarrow$. Moreover, since $i \neq j$, $(a.\ell_j \Leftarrow \varsigma(y^A)b)\vartheta.\ell_i(c) \equiv a\vartheta.\ell_i(c)$ which is in $[\![\tau]\!]$ when $c^A \in [\![\sigma]\!]$ by the inductive hypothesis. ■

Theorem 44 (Logical Equivalence and Observational Equivalence). Suppose that for any value v of ground type K we have exactly a non-trivial predicate $\kappa_v \in \mathcal{L}_K$, that this predicates are distinct for different values and that $\vdash v{:}K : \kappa_v$ is assumed for each v. Then for any a^A and b_B, if $a \simeq^{\mathcal{L}} b : A$ then $a \simeq^{\mathcal{O}}_A b$.

Proof. Towards a contradiction, assume that $a \simeq^{\mathcal{L}} b : A$ and that there exists some ground context $\_{:}A \vdash C[\_]{:}K$ such that $C[a] \Downarrow v$ and not $C[b] \Downarrow v$. By Theorem 33 it follows that there exists some non-trivial $\tau \in \mathcal{L}_K$ such that $C[a] \vdash K{:}\tau$ is derivable, and by the assumptions $\tau = \kappa_v$. By Lemma 40 we know that there exist some $\sigma \in \mathcal{L}_A$ such that $a \vdash A{:}\sigma$ and $x{:}A{:}\sigma \vdash C[x]{:}K : \tau$ (or equivalently $\_{:}A{:}\sigma \vdash C[\_]{:}K : \tau$); since $\sigma \in [\![a{:}A]\!] = [\![b{:}A]\!]$ by the absurd hypothesis, Lemma 38 implies that $\vdash C[b]{:}K : \tau$ is also derivable: now if $C[b] \Uparrow$, then this contradicts Corollary 33; if instead $C[b] \Downarrow v'$ for some value $v' \not\equiv v$ then $\vdash v'{:}K : \kappa_v$ by Theorem 18 which is impossible. ■