HAL
open science

Logical equivalence for subtyping object and recursive types

Steffen van Bakel, Ugo De'Liguoro

To cite this version:

Steffen van Bakel, Ugo De'Liguoro. Logical equivalence for subtyping object and recursive types. [Research Report] RR-5985, INRIA. 2006, pp.51. inria-00097235

HAL Id: inria-00097235

https://inria.hal.science/inria-00097235v1

Submitted on 21 Sep 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Logical equivalence for subtyping object and recursive types

Steffen van Bakel — Ugo de'Liguoro

N° ????

Septembre 2006

Thème COM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--????--FR+ENG

Logical equivalence for subtyping object and recursive types

Steffen van Bakel* , Ugo de'Liguoro †

Thème COM — Systèmes communicants
Projets MIMOSA

Rapport de recherche n° ???? — Septembre 2006 — 51 pages

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

Key-words: intersection types, varsigma calculus, sub-typing, logical equivalence

* On sabbatical leave from Department of Computing, Imperial College London, 180 Queen's Gate London SW7 2BZ, U.K. svb@doc.ic.ac.uk

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

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Introduction

Subtyping is a prominent feature of the 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 code of type B whenever A is a subtype of B, written $A <: B$.

In typed calculi, equations can be expressed between terms of the same type; since terms may have several types because of subsumption, it is commonly postulated that if $a = b : A$ (a and b are equal at type A) and $A <: B$ then $a = b : B$ (but not vice-versa): call this *equational subsumption*. In the realm of object calculi, object types are essentially *interfaces*, and subtyping *interface restriction*. 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 separate them. This provides an interpretation of subtyping: $A <: B$ should hold if any pair of terms contextually equivalent at type A cannot be separated at B.

This is semantically understood in two ways, according to the existing literature: either by means of coercions, or by inclusion of partial equivalence relations (see [20] Ch. 10 for a gentle introduction to these approaches, where coercions are called "conversion functions", and PER semantics "subset interpretation of types").

According to coercion semantics introduced in [12], the relation $A <: B$ is witnessed by the existence of a definable function from A to B, coercing values of one type into the other. Given that, each term and its type is translated into a term typeable in a system without subtyping at all. Since such a translation depends on the typing derivation of the original term, a coherence theorem is needed to prove that different typing derivations yield equivalent translations. Object and recursive types are not the concern of [12]; unfortunately when dealing with such form of polymorphic types the coercion approach has some serious disadvantages. Indeed there is no clear interpretation for object-types coercions: these cannot be reduced to function and (recursive) record types, for which coercion semantics can be easily defined, because of the self-reference which is an essential feature of objects. Furthermore it is a term-model semantics, where term complexity grows because of the translation step. Last but not least, coercion semantics does not reflect the actual implementation practice of object-oriented languages, where subtyping is just type casting which does not affect the object code.

PER semantics interprets types as binary relations over a structure of untyped realisers, and terms as (equivalence classes of) realisers. In [13] subtyping was interpreted as relation inclusion for the first time. To interpret object and recursive types, however, one needs to ensure the existence of certain fixed points of type functors, which is a quite difficult task, as

the relevant functors are not necessarily continuous, not even monotonic. The problem has been confronted by means of metrics and Banach fixed-point theorem (see [15, 6, 3, 14]), and by restricting to the category of complete uniform PERs over a realizability structure which is an inverse limit. Based on these, in [1] Ch. 14 a CUPER semantics of the ς-calculus is proposed, and the soundness of the whole system studied in the book is established. Unfortunately the adopted solution for modelling object types as (continuous) unions of fixed points of type functors is not very natural and quite complex in nature. Hence the remark that "there are obvious difficulties to extract from the (PER) models and justify a finitary programming logic" of [6] applies *a fortiori* to the case of object-calculi, where the call for such a logic is compelling.

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 [4]. 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 interpreted by the same set. 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 from the language $\mathcal{L}_A$ associated to A.

We treat three kinds of entities, namely terms, types and predicates, and define a formal system to derive judgements of the shape $a{:}A{:}\sigma$. The system is built in such a way that, if we forget about any one of these three kinds of entities, what remains is still a meaningful assignment system. Indeed, if we forget about predicates we obtain the first order object calculus called $\mathsf{FOb}_{1<:\mu}$ in [1], but for a minor difference (we do not use fold, unfold operators in the term syntax, and consider as isomorphic all the unfoldings of a recursive type).

If instead we erase all types, we get a sort of "intersection type" assignment system for the untyped ς-calculus (essentially that one used in [16] to characterise the convergence of untyped ς-terms). The predicates system has features different from the type system: objects are treated as records, deducing a predicate $\langle \ell{:}\sigma{\rightarrow}\phi\rangle$ about a single method labelled by ℓ, that can be put in conjunction with any other similar predicate, possibly with a different premise of the arrow. This works because the meaning of $a{:}\langle \ell{:}\sigma{\rightarrow}\phi\rangle$ is not that the self variable has type σ; rather it claims that σ is a precondition of the method ℓ, which yields a value $a.\ell$ satisfying the post-condition ϕ if σ holds for a. Therefore to conclude that $a.\ell{:}\phi$ one needs to show that both $a{:}\langle \ell{:}\sigma{\rightarrow}\phi\rangle$ and $a{:}\sigma$ for some σ.

Semantically speaking, this is Kamin's self-application interpretation of objects, accounting for the interpretation of self reference in case of method call and of method overriding. Eventually, this produces the effect that converging terms are characterised by

predicates, whereas types that ensure, for example, error freeness (no "message not understood" error can occur with typed terms at run time), cannot discriminate diverging terms. We claim that the resulting system is a kind of extended Curry-style assignment system, determining a model for the ς-calculus, which is an extension of the filter model for the pure Lambda Calculus of [11]. This depends on the fact that a notion of implication is defined over predicates which is written $\sigma \leq \tau$. This way we build a logic whose filters of formulae provide a denotation to terms, coinciding with the sets of their properties as expressed by the logical formulae (see [4], where this is framed as a form of Stone duality between categories of semi-lattices and of Scott domains).

The third possibility is to forget about terms; then we interpret the judgement $A{:}\sigma$ as: "the predicate σ makes sense of terms of type A", namely $\sigma \in \mathcal{L}_A$ for closed A. This logic of types, again inspired to domain logic, is our tool to treat object and recursive types. The formal system formalises the concept that both object and recursive types are some kind of fixed point, which is constructed starting with the trivial predicate ω, and iterating the proper rules determined by the structure of type expressions. We observe that this definition of the languages of object and recursive types is inductive, and that it makes sense without any consideration about the invariance of the object types nor (and more importantly) about the variance of the occurrences in A of the type variable X within the recursive type $\mu X.A$.

The importance of the interplay between these assignment systems (actually integrated in a unique system) emerges when treating program equivalence. The system induces an equivalence relation $a \simeq b : A$, which, when restricted to closed terms and types, expresses: the sets of predicates of type A that can be assigned to a and to b coincide. Observing that the derivability of $a{:}A{:}\sigma$ always implies that $\sigma \in \mathcal{L}_A$, this formalises the above idea of relativizing logical semantics to types.

To verify that this is a sound theory of the first order object calculus, we prove that $\vdash_{\mathrm{T}} a \leftrightarrow b : A$ (the equational theory of $\mathsf{FOb}_{1<:\mu}$) implies that $a \simeq b : A$, which in turn implies that $a \simeq^{\mathcal{O}}_{A} b$, namely that a and b cannot be separated by any context $\_ : A \vdash C[\_] : K$ of any ground type K. This Morris-style contextual equivalence is called the *observational equivalence* in [18] and is a maximal consistent theory of the first order object calculus.

To establish the latter results we have to relate our theory of predicates assignment to terms and types to the theory of subtyping. First we establish that if $A <: B$ then $\mathcal{L}_A \supseteq \mathcal{L}_B$: hence the logical theory associated to A is finer than the theory of B, so that any pair of terms which are indiscernible according to A, are such in the coarser theory of B but not vice-versa. This is indeed our interpretation of subtyping and equational subsumption.

To prove the inclusion of the logical equivalence in the observational equivalence we use a realizability interpretation of predicates instead of types. This is not surprising since the key property of convergence is not captured by the type system, but only by the predicate

system, as remarked above. We think that the logical equivalence is the theory of a model of the typed calculus, which can be constructed as a filter model. We do not enter into the details of this construction in the present paper, and shortly comment on it in Section *7*.

Overview of contents

The paper is organised as follows: in Section *1* we introduce our variant of the $\mathsf{FOb}_{1<:\mu}$ system. We define predicates, their logic and assignment to types in Section *2*. The assignment system, which essentially puts together the typing system with the logic of predicates is defined and studied in Section *3*. Basic soundness results of the assignment system with respect to the operational semantics are established via subject reduction and (typed) subject expansion in Section *4*. In Section *5* we recall the equational theory of $\mathsf{FOb}_{1<:\mu}$ from [1]; we then formally define the logical equivalence and show that the former is included in the latter. Finally, in Section *6*, we define the observational semantics after [18] and prove the inclusion of the logical semantics in it. We eventually discuss the results presented in the paper, and relate our work to the literature on the subject in Section *7*.

1 The first order object calculus

We consider a first order object calculus which is a variant of the calculus called $\mathsf{FOb}_{1<:\mu}$ in [1]. The difference between that version and the one we treat here is that we consider recursive types and their unfoldings as equivalent with respect to subtyping. Consequently, we do not have any syntax to distinguish among the folded and unfolded version of the same term $\mathsf{fold}(A, a)$ and $\mathsf{unfold}(a)$, which will be written simply a. Indeed semantically they are all equivalent expressions in [1], where fold and unfold do not affect the computational behaviour of terms and are used just to ensure the existence of a minimal type of any given term: a useful property for type reconstruction algorithms which is not the present concern, however.

For the sake of readability the calculus is introduced in two steps: we first define pre-types A, pre-terms a and pre-environments E by means of grammars. These are not necessarily well-formed types, terms and environments, as the latter notions are defined via derivation systems. We will in fact present a number of type assignment systems that are interdependent each other, whose statements (judgements and sequents) have the intended meaning:

$E \vdash_{\mathrm{T}} \diamond$	*E is a well formed environment (an environment for short)*
$E \vdash_{\mathrm{T}} A$	*A is a type within the environment E;*
$E \vdash_{\mathrm{T}} A <: B$	*A is a subtype of B within the environment E;*
$E \vdash_{\mathrm{O}} a : A$	*a is a term of type A in the environment E;*

Definition 1.1 PRE-TYPES, PRE-TERMS, PRE-ENVIRONMENTS. Let $\mathcal{K}$ be a countable set of type constants ranged over by K, and $\mathcal{X}$ a denumerable set of type variables ranged over by X. Let $L = \{\ell_i \mid i \in \mathsf{IN}\}$ be a denumerable set of labels; let $\mathcal{C}$ a countable set of term constants ranged over by c and $\mathcal{V}$ a denumerable set of term variables ranged over by x. The syntax of *pre-types*, *pre-terms* and *pre-environments* is defined by the following grammar:

Pre-Types:	$A, B ::= X \mid K \mid \mathsf{Top} \mid [\ell_i{:}B_i\ ^{(i \in I)}] \mid A{\rightarrow}B \mid \mu X.A$
Pre-Terms:	$a, b ::= x \mid c \mid \lambda x^A.a \mid a(b) \mid [\ell_i = \varsigma(x_i^A)b_i\ ^{(i \in I)}] \mid a.\ell \mid a.\ell \Leftarrow \varsigma(x^A)b$
Pre-Contexts:	$E ::= \emptyset \mid E, X \mid E, X <: A \mid E, x{:}A$

where I ranges over finite subsets of IN, and $\ell_i, \ell \in L$.

The notion of free and bound occurrences of variables, as well as substitution, are defined as usual.

Definition 1.2 FREE AND BOUND, SUBSTITUTION. We say that X is *free* in A if it does not occur within the scope of μX; it is *bound* otherwise. Similarly, x is *free* in a if it does not occur within the scope of some λx^A nor of some $\varsigma(x^A)$; it is *bound* otherwise. We use *fv*(A) and *fv*(a) to denote the sets of free variables occurring in A and a respectively; *bv*(A) and *bv*(a) denote the sets of the bound variables.

By $A\{X \leftarrow\!\!\dashv B\}$ and $a\{x \leftarrow\!\!\dashv b\}$ we denote the *substitution* of X and x by B and b in A and a respectively, up to renaming of bound variables to avoid variable clashes.

Syntactic equality, up to the renaming of bound variables, is denoted by $\equiv$.

Definition 1.3 TERMINOLOGY. *i*) An *object pre-term* has the shape $[\ell_i = \varsigma(x_i^A)b_i\ ^{(i \in I)}]$, namely a finite collection of methods $\varsigma(x_i^A)b_i$ labelled by distinct ℓ_i; the actual order of methods is immaterial, which justifies the set-theoretic notation $(i \in I)$;

ii) In $\ell_i = \varsigma(x_i^A)b_i$, the variable x_i^A is the *self* variable, and b_i is the *body* of the *method* ℓ_i; if $x_i \notin$ *fv*(b_i) then ℓ_i is more properly seen as a *field*;

iii) $a.\ell$ denotes the *invocation* of method ℓ of the object a, if a evaluates to an object having such a method;

iv) $a.\ell \Leftarrow \varsigma(x^A)b$ is the *overriding* of method ℓ in the object to which a evaluates, if any;

v) Functional abstraction and application are represented as usual in typed λ-calculi, via $\lambda x^A.a$ and $a(b)$.

Term-operational semantics does not depend on types, and can be defined directly over pre-terms.

Definition 1.4 REDUCTION. *i*) *Evaluating contexts* are term expressions with a hole $[\_]$, and are generated by the grammar:

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

We will write $\mathcal{E}[a]$ for filling the hole $[\_]$ in $\mathcal{E}$ with a.

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

$$\begin{array}{rcl}
[\ell_i = \varsigma(x_i^{A_i})b_i{}^{(i\in I)}].\ell_j & \longrightarrow & b_j\{x_j \leftarrowtail [\ell_i = \varsigma(x_i^{A_i})b_i{}^{(i\in I)}]\} \\
[\ell_i = \varsigma(x_i^{A_i})b_i{}^{(i\in I)}].\ell_j \Leftarrow \varsigma(x^A)b & \longrightarrow & [\ell_i = \varsigma(x_i^{A_i})b_i{}^{i\in I\setminus j}, \ell_j = \varsigma(x^A)b] \\
(\lambda x^A.a)(b) & \longrightarrow & a\{x \leftarrowtail b\} \\
a \longrightarrow b & \Rightarrow & \mathcal{E}[a] \longrightarrow \mathcal{E}[b]
\end{array}$$

where in the first two rules $j \in I$ is required.

iii) The relation $\stackrel{*}{\longrightarrow}$ is the reflexive and transitive closure of $\longrightarrow$.

The reduction relation is essentially the same in [18]. It is trivially confluent. Even relaxing Definition *1.4* and taking the closure of $\longrightarrow$ 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 $\Omega_B \longrightarrow \Omega_B$.

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, and v is a *value*. Values are defined as follows:

Definition 1.5 VALUES. A *value* is a closed pre-term belonging to the set defined by the grammar:

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

It is easy to see that $a \rightsquigarrow v$ if and only if $a \stackrel{*}{\longrightarrow} v$. The reduction relation is more general since it is defined for any term (possibly with free variable occurrences). It is even true that

Figure 1: Contexts and Types

$$(\mathsf{Env}\ \emptyset):\ \frac{}{\emptyset \vdash_{\mathrm{T}} \diamond} \qquad (\mathsf{Type\ Const}):\ \frac{E \vdash_{\mathrm{T}} \diamond}{E \vdash_{\mathrm{T}} K} \qquad (\mathsf{Env}\ X):\ \frac{E \vdash_{\mathrm{T}} \diamond}{E, X \vdash_{\mathrm{T}} \diamond}\ (X \notin \mathit{dom}(E)) \qquad (\mathsf{Type}\ X):\ \frac{E', X, E'' \vdash_{\mathrm{T}} \diamond}{E', X, E'' \vdash_{\mathrm{T}} X}$$

$$(\mathsf{Type\ Top}):\ \frac{E \vdash_{\mathrm{T}} \diamond}{E \vdash_{\mathrm{T}} \mathsf{Top}} \qquad (\mathsf{Type\ Object}):\ \frac{E \vdash_{\mathrm{T}} B_i \quad (\forall i \in I)}{E \vdash_{\mathrm{T}} [\ell_i{:}B_i\ ^{(i \in I)}]} \qquad (\mathsf{Type\ Arrow}):\ \frac{E \vdash_{\mathrm{T}} A \quad E \vdash_{\mathrm{T}} B}{E \vdash_{\mathrm{T}} A{\to}B} \qquad (\mathsf{Type\ Rec}):\ \frac{E, X \vdash_{\mathrm{T}} A}{E \vdash_{\mathrm{T}} \mu X.A}$$

normal forms are not necessarily values, but if a is typeable (i.e. a term), having a normal form b, then b is value, as follows by Theorem *1.17* below and the fact that closed normal forms which are not values are not typeable.

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

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

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 subtyping, but it is needed in the presence of type 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 (see below rule (Val Object) for object typing).

The simply typed λ-calculus is a sub-calculus of the Object Calculus: since the only evaluation contexts dealing with abstraction and application have the shape $\mathcal{E}[\_](a)$, it is a lazy λ-calculus, in the sense of [5]. Also objects are "lazy", in the sense that the bodies of the methods are not reducible before selection.

We adopt the alternative notation $a \Downarrow v$ for $a \rightsquigarrow v$, as it is commonly used in the literature of λ-calculus and related systems.

Definition 1.6 CONVERGENCE. Given any closed term a we say that it *converges* to the value v, written $a \Downarrow v$, if $a \xrightarrow{*} v$. Moreover we say that a is convergent, written $a \Downarrow$, if there exists a value v such that $a \Downarrow v$.

We will now repeat the construction of the definition of type assignment.

Definition 1.7 TERMINOLOGY (CONTINUED). *i*) A pre-type of the shape $[\ell_i{:}B_i \quad {}^{(i \in I)}]$ will be used for an object whose methods ℓ_i have *return type* B_i;

ii) $A{\to}B$ is the usual *functional* type and $\mu X.A$ is a *recursive* type;

iii) Top is the *maximal* type w.r.t. the subtyping relation $<:$.

The notions of *inference rule* and *derivation* are as usual. As in [1], we will use a short-hand for rules, and write for example (where $I = \{1, \ldots, n\}$)

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

Definition 1.8 PRE-ENVIRONMENT. A *pre-environment* E is defined via the grammar:

$$E ::= \emptyset \mid E, X \mid E, X <: A \mid E, x{:}A$$

where X is a type variable, A a pre-type, x a term variable.

The *domain* of a pre-environment E is the set of type and term variables occurring in E, and is defined by:

$$\begin{aligned} \mathit{dom}(\emptyset) &= \emptyset, \\ \mathit{dom}(E, X) = \mathit{dom}(E, X <: A) &= \mathit{dom}(E) \cup \{X\}, \\ \mathit{dom}(E, x{:}A) &= \mathit{dom}(E) \cup \{x\}. \end{aligned}$$

Although pre-environments are formally sequences, by abuse of notation we shall treat them as sets and write $X \in E$, $X <: A \in E$ or $x{:}A \in E$ to mean that $X, X <: A$ and $x{:}A$ occur as elements of the sequence E respectively.

Definition 1.9 JUDGEMENTS AND SEQUENTS. A *type judgement* has one of the forms $\diamond$, A, $A <: B$, or $a : A$, where A and B are pre-types and a is a pre-term. A *type sequent* has the form $E \vdash \Theta$ where E is a pre-environment and Θ is a type judgement.

Definition 1.10 THE TYPED OBJECT CALCULUS. A sequent $E \vdash \Theta$ is *derivable* in the calculus of objects if there exists a derivation whose inferences are instances of the rules in figures *1*, *2* and *3*, such that $E \vdash \Theta$ appears in the bottom line.

We say that E is an *environment*, A is a *type* in the environment E and a a *term* (of type A) in the environment E, if $E \vdash_{\mathrm{T}} \diamond$, $E \vdash_{\mathrm{T}} A$ and $E \vdash_{\mathrm{O}} a : A$ are derivable respectively.

We will write $\mathcal{D} :: E \vdash \Theta$ when $\mathcal{D}$ is a derivation whose conclusion is the sequent $E \vdash \Theta$, and will write $E \vdash \Theta$ if there exists a derivation $\mathcal{D}$ such that $\mathcal{D} :: E \vdash \Theta$, i.e. this sequent is derivable.

Figure 2: Subtyping

$$(\textsf{Sub Refl}):\quad \frac{E \vdash_{\mathrm{T}} A}{E \vdash_{\mathrm{T}} A <: A} \qquad (\textsf{Sub Top}):\quad \frac{E \vdash_{\mathrm{T}} A}{E \vdash_{\mathrm{T}} A <: \textsf{Top}} \qquad (\textsf{Sub Trans}):\quad \frac{E \vdash_{\mathrm{T}} A <: B \quad E \vdash_{\mathrm{T}} B <: C}{E \vdash_{\mathrm{T}} A <: C}$$

$$(\textsf{Sub } X):\quad \frac{E', X <: A, E'' \vdash_{\mathrm{T}} \diamond}{E', X <: A, E'' \vdash_{\mathrm{T}} X <: A} \qquad (\textsf{Env } X<:):\quad \frac{E \vdash_{\mathrm{T}} A}{E, X <: A \vdash_{\mathrm{T}} \diamond}\ (X \notin \mathit{dom}(E)) \qquad (\textsf{Type } X<:):\quad \frac{E', X <: A, E'' \vdash_{\mathrm{T}} \diamond}{E', X <: A, E'' \vdash_{\mathrm{T}} X}$$

$$(\textsf{Sub Object}):\quad \frac{E \vdash_{\mathrm{T}} B_i \quad (\forall i \in I)}{E \vdash_{\mathrm{T}} [\ell_i{:}B_i{}^{\,i \in I}] <: [\ell_i{:}B_i{}^{\,i \in J}]}\ (J \subseteq I) \qquad (\textsf{Sub Arrow}):\quad \frac{E \vdash_{\mathrm{T}} A' <: A \quad E \vdash_{\mathrm{T}} B <: B'}{E \vdash_{\mathrm{T}} A \to B <: A' \to B'}$$

$$(\textsf{Sub Rec}_1):\quad \frac{E \vdash_{\mathrm{T}} \mu X.A \quad E \vdash_{\mathrm{T}} A\{X \hookleftarrow \mu X.A\}}{E \vdash_{\mathrm{T}} \mu X.A <: A\{X \hookleftarrow \mu X.A\}} \qquad (\textsf{Sub Rec}_2):\quad \frac{E \vdash_{\mathrm{T}} \mu X.A \quad E \vdash_{\mathrm{T}} A\{X \hookleftarrow \mu X.A\}}{E \vdash_{\mathrm{T}} A\{X \hookleftarrow \mu X.A\} <: \mu X.A}$$

$$(\textsf{Sub Rec}_3):\quad \frac{E \vdash_{\mathrm{T}} \mu X.A \quad E \vdash_{\mathrm{T}} \mu Y.B \quad E, Y, X <: Y \vdash_{\mathrm{T}} A <: B}{E \vdash_{\mathrm{T}} \mu X.A <: \mu Y.B}$$

From now on, when dealing with environments, types and terms, we will assume they are well formed.

Remark 1.11 *i*) For reasons of simplicity we assume that term constants have always constant (and ground) types. These types are just implicitly assumed, and we do not record them anywhere in the syntax.

ii) By comparing rules from figures *1* and *2* it is clear that $E, X <: \textsf{Top} \vdash_{\mathrm{T}} \diamond$ is derivable if and only if $E, X \vdash_{\mathrm{T}} \diamond$ is derivable. Hence we will omit rule (Type Rec$<:$) of [1], since it can be replaced by (Type Rec).

iii) Rules (Sub Rec$_1$) and (Sub Rec$_2$) imply that $\mu X.A$ and $A\{X \hookleftarrow \mu X.A\}$ are isomorphic, which, as noted above, is a departure from system $\textsf{FOb}_{1<:\mu}$. Consequently, there are no $\textsf{fold}(A, a)$ or $\textsf{unfold}(a)$ pre-terms, nor their corresponding typing rules. The original rule (Sub Rec) is our rule (Sub Rec$_3$). This implies the loss of the

minimal type property: it is not true that any well-formed (and typed) term has a minimal type w.r.t. $<:$. This is a problem when designing a type inference algorithm; rather it is the choice of making the syntax more akin to its intended meaning in a foundational setting. Concerning our predicate assignment system (see Section *2*) the choice has some minor consequences: if one does not admit the law of isomorphism of recursive types, as we did in [10], then the only technical problem is that a and $\mathsf{fold}(A, a)$ become distinct entities of types $A\{X \hookleftarrow \mu X.A\}$ and $\mu X.A$ respectively; to catch such a distinction in [10] we introduced predicates of the shape $\mu(\sigma)$ such that if $\sigma \in \mathcal{L}_{A\{X \hookleftarrow \mu X.A\}}$ then $\mu(\sigma) \in \mathcal{L}_{\mu X.A}$. The relevant results in the paper still hold, with the necessary changes in the predicate interpretation (see Definition *3.3*).

The following lemmas state some basic properties of the system. We write $E \vdash_{\mathrm{T}} A =: B$ to abbreviate $E \vdash_{\mathrm{T}} A <: B$ and $E \vdash_{\mathrm{T}} B <: A$.

Lemma 1.12 *i) If $E \vdash_{\mathrm{T}} A$ and $X \notin dom(E)$ then $E \vdash_{\mathrm{T}} A =: \mu X.A$;*
ii) if $E \vdash_{\mathrm{T}} K <: A$ for ground K, then $E \vdash_{\mathrm{T}} A =: K$;
iii) if $E \vdash_{\mathrm{T}} A <: B \to C$ then there exist A', A'' such that $E \vdash_{\mathrm{T}} A =: A' \to A''$ and $E \vdash_{\mathrm{T}} B <: A'$ and $E \vdash_{\mathrm{T}} A'' <: C$;
iv) if $E \vdash_{\mathrm{T}} A <: [\ell_i{:}B_i\ ^{(i \in I)}]$ then there exists some $J \supseteq I$ such that $\vdash_{\mathrm{T}} A =: [\ell_j{:}B_j\ ^{(j \in J)}]$.

Proof: Immediate by inspection of rules. Observe that if $E \vdash_{\mathrm{T}} A$ and $X \notin dom(E)$ then $X \notin fv(A)$; hence $E \vdash_{\mathrm{T}} \mu X.A =: A$ by $(\mathsf{Sub\ Rec}_1)$, $(\mathsf{Sub\ Rec}_2)$, since $A \equiv A\{X \hookleftarrow \mu X.A\}$ in this case. ■

Lemma 1.13 TYPE GENERATION LEMMA. *i) If $E \vdash_{\mathrm{T}} A$ then $E \vdash_{\mathrm{T}} \diamond$;*
ii) if $E, X \vdash_{\mathrm{T}} B$, then $X \notin dom(E)$;
iii) if $E \vdash_{\mathrm{T}} X$ then $X \in dom(E)$;
iv) if $E \vdash_{\mathrm{T}} A$ then $fv(A) \subseteq dom(E)$;
v) if $E \vdash_{\mathrm{T}} A <: B$ then both $E \vdash_{\mathrm{T}} A$ and $E \vdash_{\mathrm{T}} B$;
vi) if $E', X <: A, E'' \vdash_{\mathrm{T}} \diamond$ then $E' \vdash_{\mathrm{T}} A$;
vii) if $E \vdash_{\mathrm{T}} [\ell_i{:}B_i\ ^{(i \in I)}]$ then $E \vdash_{\mathrm{T}} B_i$ for all $i \in I$;
viii) if $E \vdash_{\mathrm{T}} A \to B$ then both $E \vdash_{\mathrm{T}} A$ and $E \vdash_{\mathrm{T}} B$;
ix) if $E \vdash_{\mathrm{T}} \mu X.A$ then $E, X \vdash_{\mathrm{T}} A$;
x) if $E \vdash_{\mathrm{T}} [\ell_i{:}B_i\ ^{(i \in I)}]$ then $E \vdash_{\mathrm{T}} B_i$ for all $i \in I$.

Proof: Straightforward. ■

The following is easy to show:

Figure 3: Typed terms

(Val Const) :
$$\frac{E \vdash_{\mathrm{T}} \diamond}{E \vdash_{\mathrm{O}} c : K} \text{ (for } c \text{ of type } K)$$

(Env x) :
$$\frac{E \vdash_{\mathrm{T}} A}{E, x{:}A \vdash_{\mathrm{O}} \diamond} \; (x \notin \mathit{dom}(E))$$

(Val x) :
$$\frac{E', x{:}A, E'' \vdash_{\mathrm{T}} \diamond}{E', x{:}A, E'' \vdash_{\mathrm{O}} x : A}$$

(Val Fun) :
$$\frac{E, x{:}A \vdash_{\mathrm{O}} a : B}{E \vdash_{\mathrm{O}} \lambda x^A.a : A{\to}B}$$

(Val Appl) :
$$\frac{E \vdash_{\mathrm{O}} a : A{\to}B \quad E \vdash_{\mathrm{O}} b : A}{E \vdash_{\mathrm{O}} a(b) : B}$$

(Val Subsumption) :
$$\frac{E \vdash_{\mathrm{O}} a : A \quad E \vdash_{\mathrm{T}} A <: B}{E \vdash_{\mathrm{O}} a : B}$$

In the subsequent rules, let $A \equiv [\ell_i{:}B_i{}^{\,i \in I}]$:

(Val Object) :
$$\frac{E, x_i{:}A \vdash_{\mathrm{O}} b_i : B_i \quad (\forall i \in I)}{E \vdash_{\mathrm{O}} [\ell_i = \varsigma(x_i^A)b_i{}^{\,(i \in I)}] : A}$$

(Val Select) :
$$\frac{E \vdash_{\mathrm{O}} a : A}{E \vdash_{\mathrm{O}} a.\ell_j : B_j} \; (j \in J)$$

(Val Update) :
$$\frac{E \vdash_{\mathrm{O}} a : A \quad E, x{:}A \vdash_{\mathrm{O}} b : B_j}{E \vdash_{\mathrm{O}} (a.\ell_j \Leftarrow \varsigma(x^A)b) : A} \; (j \in J)$$

Lemma 1.14 TYPED TERMS GENERATION LEMMA. *i) if $E \vdash_{\mathrm{O}} a : A$ then $E \vdash_{\mathrm{T}} A$;*

ii) if $E \vdash_{\mathrm{O}} a : A$ and $x \in \mathit{fv}(a)$ then $x \in \mathit{dom}(E)$;

iii) If $E \vdash_{\mathrm{O}} c : C$ and c is a constant of some ground type K, then $C \equiv K$;

iv) If $E \vdash_{\mathrm{O}} x : C$ then there exists an A such that $x{:}A$ occurs in E and $E \vdash_{\mathrm{T}} A <: C$;

v) If $E \vdash_{\mathrm{O}} [\ell_i = \varsigma(x_i^{A_i})b_i{}^{\,(i \in I)}] : C$, then there exist $A \equiv [\ell_i{:}B_i{}^{\,(i \in I)}]$ such that $E \vdash_{\mathrm{T}} A <: C$ and, for all $i \in I$, $A \equiv A_i$ and $E, x_i{:}A \vdash_{\mathrm{O}} b_i : B_i$;

vi) If $E \vdash_{\mathrm{O}} a.\ell : C$ then there exists an $A \equiv [\ldots, \ell{:}B, \ldots]$ such that $E \vdash_{\mathrm{T}} A$, $E \vdash_{\mathrm{T}} B <: C$ and $E \vdash_{\mathrm{O}} a : A$;

vii) If $E \vdash_{\mathrm{O}} (a.\ell \Leftarrow \varsigma(x^D)b) : C$ then $E \vdash_{\mathrm{T}} D <: C$ and there exists an $A \equiv [\ldots, \ell{:}B, \ldots]$ such that
$E \vdash_{\mathrm{T}} A <: D$, $E \vdash_{\mathrm{O}} a : A$ and $E, x{:}D \vdash_{\mathrm{O}} b : B$;

viii) If $E \vdash_{\mathrm{O}} \lambda x^A.a : C$, then there exists B such that $E, x{:}A \vdash_{\mathrm{O}} a : B$, and $E \vdash_{\mathrm{T}} A{\to}B <: C$;

ix) If $E \vdash_{\mathrm{O}} a(b) : C$, then there exist A, B such that $E \vdash_{\mathrm{O}} a : A{\to}B$, $E \vdash_{\mathrm{O}} b : A$ and $E \vdash_{\mathrm{T}} B <: C$.

Proof: By induction on the structure of derivations. ■

The following property is standard, and allows to generalise derivable results.

Lemma 1.15 WEAKENING. *i) if $E \vdash_{\mathrm{T}} A$ and $X \notin dom(E)$, then $E, X \vdash_{\mathrm{T}} A$; if also $E \vdash_{\mathrm{T}} C$ then $E, X <: C \vdash_{\mathrm{T}} A$;*
ii) if $E \vdash_{\mathrm{T}} A <: B$ and $X \notin dom(E)$ then $E, X \vdash_{\mathrm{T}} A <: B$;
iii) if $E \vdash_{\mathrm{T}} A <: B$, $X \notin dom(E)$ and $E \vdash_{\mathrm{T}} C$ then $E, X <: C \vdash_{\mathrm{T}} A <: B$;
iv) if $E \vdash_{\mathrm{O}} a : A$, $E \vdash_{\mathrm{T}} B$ and $x \notin dom(E)$, then $E, x{:}B \vdash_{\mathrm{O}} a : A$.

Proof: By induction on derivations. We just remark that, e.g. in case of *(i)*, if $E \vdash_{\mathrm{T}} A$ then $E \vdash_{\mathrm{T}} \diamond$ by Lemma *1.13*, so that $E, X \vdash_{\mathrm{T}} \diamond$ because $X \notin dom(E)$ is the side condition of rule $(\mathsf{Env}\ X)$. It follows that the derivation of $E, X \vdash_{\mathrm{T}} A$ is essentially the same as the given derivation of $E \vdash_{\mathrm{T}} A$, but for the sub-derivation of the sequent $E, X \vdash_{\mathrm{T}} \diamond$. In case of $E, X <: C \vdash_{\mathrm{T}} A$ we need the hypothesis $E \vdash_{\mathrm{T}} C$ because of the premise of rule $(\mathsf{Env}\ X{<:})$. Note that $X \notin dom(E)$ and $E \vdash_{\mathrm{T}} C$ imply $X \notin fv(C)$ by *(iv)* of Lemma *1.13*.

The proofs of the other items are similar. ■

Lemma 1.16 SUBSTITUTION LEMMA FOR $\vdash_{\mathrm{O}}$. *If $E, x{:}A \vdash_{\mathrm{O}} b : B$ and $E \vdash_{\mathrm{O}} a : A$ then $E \vdash_{\mathrm{O}} b\{x \leftarrow a\} : B$.*

Proof: By induction on the structure of derivations using Lemma *1.14*. The proof is similar to that of Lemma *4.2*. ■

Using Lemma *1.16*, we can prove the following theorem:

Theorem 1.17 SUBJECT REDUCTION FOR $\vdash_{\mathrm{O}}$. *If $E \vdash_{\mathrm{O}} a : A$ and $a \longrightarrow b$, then $E \vdash_{\mathrm{O}} b : A$.*

Proof: By induction over the definition of $\longrightarrow$ and using Lemma *1.14*; the proof is similar to that of Theorem *4.3* but simpler. ■

Theorem *1.17* is the most relevant result about typed ς-calculus in [1]. It implies that no "message not understood" error can occur in the evaluation (formally in the reduction) of any well-typed term (a term in our terminology). In fact, for such an error to occur in the reduction of a it should be the case that $a \xrightarrow{*} b.\ell$ or $a \xrightarrow{*} (b.\ell \Leftarrow \varsigma(x^A)d)$, where b is some object not including a method labelled by ℓ. But since such a $b.\ell$ or $b.\ell \Leftarrow \varsigma(x^A)d$ has no type, because of *(v)*, *(vi)* and *(vii)* of Lemma *1.14*, Theorem *1.17* says that a has no type as well.

2 Typed Predicates and Languages

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

Term properties are formalised by predicates, which in turn are classified by types. Predicates are transparently intersection types for a λ-calculus with records, and come from [16]. The essential difference is that the set of predicates is stratified into languages (see [17, 9]), in such a way that whenever a predicate can be deduced for a (closed) term a, it belongs to the language $\mathcal{L}_A$ associated with the (closed) type A.

Much in the style of [8], in this section we will present a notion of *strict intersection types*, called *strict predicates* here. Using these, we will define in the next section a notion of *predicate assignment*, which will consists basically of associating a predicate to a typed term.

Definition 2.1 PREDICATES. The set $\mathcal{P}$ of *predicates*, ranged over by $\sigma, \tau, \ldots$ and its subset $\mathcal{P}_{\mathrm{s}}$ of *strict predicates* ranged over by $\phi, \psi, \ldots$, are defined through the grammar:

$$\begin{aligned} \phi &::= \kappa \mid \omega \mid (\sigma \to \phi) \mid \langle \ell{:}\phi \rangle \\ \sigma, \tau &::= \phi \mid (\sigma \wedge \tau) \end{aligned}$$

where κ ranges over a countable set of atoms, and $\ell \in L$ is any (method) label.

Since $\wedge$ is commutative and associative w.r.t. the equivalence introduced below in Definition *2.2*, we omit brackets and write $\bigwedge_{i \in I} \sigma_i$ for $\sigma_1 \wedge \ldots \wedge \sigma_n$. Also, rather than $\langle \ell_1{:}\phi_1 \rangle \wedge \cdots \wedge \langle \ell_n{:}\phi_n \rangle$ where the ℓ_i are pair-wise distinct, we will write $\langle \ell_i{:}\phi_i \ {}^{(i \in I)} \rangle$. By definition, any predicate $\sigma \in \mathcal{P}$ is such that $\sigma \equiv \bigwedge_{i \in I} \phi_i$ for some non empty finite I and certain $\phi_i \in \mathcal{P}_{\mathrm{s}}$. We shall use ϕ, ψ possibly with apices and indices for elements of $\mathcal{P}_{\mathrm{s}}$, while $\sigma, \tau \in \mathcal{P} \supseteq \mathcal{P}_{\mathrm{s}}$ (with similar decorations) may be strict or not.

Predicates are strict intersection types in the sense of [7, 8] but for the fact that the type constant ω is no longer treated as the empty intersection, which allows for occurrences of ω at the right of arrows. Also record predicates $\langle \ell{:}\phi \rangle$ are added. With respect to ordinary intersection types, which have been introduced by several authors in a series of papers (see e.g. [8] for references), the occurrence of intersection $\wedge$ is not allowed at the right of an arrow; this is a technical choice and a departure from [9], making the proof theory of the system more manageable, since it allows for a more syntax directed treatment of the assignment system, without loss of expressivity.

Atomic predicates κ are intended to describe elements of atomic types in the domain of interpretation, but they can be used also to denote subsets of such values (e.g. the odd or even

integers in the interpretation of *Int*), allowing for a limited form of abstract interpretation. $\sigma{\to}\phi$ is the property of functions sending elements satisfying σ into elements satisfying ϕ. $\langle\ell{:}\phi\rangle$ is the property of records having values that satisfy ϕ associated with the field ℓ. Predicates ω and $\sigma{\wedge}\tau$ mean '*truth*' and '*conjunction*' respectively. It should be noted that arbitrary conjunctive predicates like $(\sigma{\to}\phi){\wedge}\langle\ell{:}\psi\rangle$ are allowed by the above definition, although never derived for any type nor for any term by the assignment systems.

To build a logic of predicates we need a notion of implication, written $\sigma \leq \tau$ (read as: "σ implies τ"), which is a reflexive and transitive relation on predicates, as defined below. Also, as in [8], but differently w.r.t. [7], we define $\leq$ to be *contra-variant* in arrow types.

Definition 2.2 PREDICATE PRE-ORDER. The relation $\leq$ over predicates is defined as the last pre-order such that for any $\sigma, \tau \in \mathcal{P}$ and $\phi, \psi \in \mathcal{P}_{\mathrm{S}}$:

i) $\sigma \leq \omega$;
ii) $(\sigma{\wedge}\tau)$ is the meet of σ and τ;
iii) $(\sigma{\to}\omega) \leq (\omega{\to}\omega)$;
iv) $\sigma \geq \tau, \phi \leq \psi \;\Rightarrow\; (\sigma{\to}\phi) \leq (\tau{\to}\psi)$;
v) $\phi \leq \psi \;\Rightarrow\; \langle\ell{:}\phi\rangle \leq \langle\ell{:}\psi\rangle$, for any $\ell \in L$.

Finally $\sigma = \tau \iff \sigma \leq \tau \leq \sigma$, and we write $\sigma < \tau$ if $\sigma \leq \tau$ and $\sigma \neq \tau$. A predicate is *trivial* if it is equivalent to ω.

The relation $\leq$ differs from that considered in [11], in that there $\omega{\to}\omega = \omega$, whereas here we only allow $\omega{\to}\omega < \omega$. This is natural in the present context of lazy evaluation, where an abstraction should always have a conjunction of arrow predicates, hence different from ω, even if it does not return a result. Since strict intersection types are essentially representatives of equivalence classes of type in [11], in [8], $\omega{\to}\omega$ is not a type; any term typeable by that type in [11] is typeable only by ω in [8].

Lemma 2.3 For any $\sigma, \tau \in \mathcal{P}$, and $\phi, \psi \in \mathcal{P}_{\mathrm{S}}$:

i) the connective $\wedge$ is monotonic w.r.t. $\leq$;
ii) σ is trivial if and only if $\sigma \equiv \bigwedge_{i\in I} \omega$ for any (finite) I;
iii) $\sigma{\to}\phi \leq \omega{\to}\omega$ and $\langle\ell{:}\phi\rangle \leq \langle\ell{:}\omega\rangle$ for any $\ell \in L$;
iv) $I \supseteq J, \forall j \in J.\ \phi_j \leq \psi_j \;\Rightarrow\; \langle\ell_i{:}{\phi_i}^{i\in I}\rangle \leq \langle\ell_j{:}{\psi_j}^{j\in J}\rangle$.

Proof: To see *(i)* let $\sigma \leq \sigma'$ and $\tau \leq \tau'$; then by the definition of $\wedge$ as the meet w.r.t. $\leq$ we have that $\sigma{\wedge}\tau \leq \sigma \leq \sigma'$ and $\sigma{\wedge}\tau \leq \tau \leq \tau'$, which implies that $\sigma{\wedge}\tau \leq \sigma'{\wedge}\tau'$.

By inspection of the axioms of $\leq$ it is evident that if $\omega \not\equiv \phi \in \mathcal{P}_{\mathrm{S}}$ then $\phi \neq \omega$; hence *(ii)* follows by the fact that $\sigma \equiv \bigwedge_{i\in I} \phi_i$ for some $\phi_i \in \mathcal{P}_{\mathrm{S}}$ and that if either $\sigma_i \neq \omega$ for $i = 1, 2$, then $\sigma_1{\wedge}\sigma_2 \neq \omega$ for any σ_i.

Figure 4: Predicates to Types Assignment System

$$(\mathsf{Env}\ \emptyset):\ \frac{}{\emptyset \vdash_{\mathrm{PT}} \diamond} \qquad (\mathsf{Env}\ X):\ \frac{\Delta \vdash_{\mathrm{PT}} \diamond}{\Delta, X{:}\sigma \vdash_{\mathrm{PT}} \diamond}\,(X \notin \mathit{dom}(\Delta)) \qquad (\mathsf{Env}\ X{<:}):\ \frac{\Delta \vdash_{\mathrm{PT}} A : \tau}{\Delta, X{:}\sigma <: A \vdash_{\mathrm{PT}} \diamond}\,(X \notin \mathit{dom}(\Delta), \sigma \leq \tau)$$

$$(\mathsf{Type\ Const}):\ \frac{\Delta \vdash_{\mathrm{PT}} \diamond}{\Delta \vdash_{\mathrm{PT}} K{:}\kappa} \qquad (\mathsf{Type}\ X):\ \frac{\Delta', X{:}\sigma, \Delta'' \vdash_{\mathrm{PT}} \diamond}{\Delta', X{:}\sigma, \Delta'' \vdash_{\mathrm{PT}} X{:}\phi}\,(\sigma \leq \phi) \qquad (\mathsf{Type}\ X{<:}):\ \frac{\Delta', X{:}\sigma <: A, \Delta'' \vdash_{\mathrm{PT}} \diamond}{\Delta', X{:}\sigma <: A, \Delta'' \vdash_{\mathrm{PT}} X{:}\phi}\,(\sigma \leq \phi)$$

$$(\mathsf{Type\ Arrow}):\ \frac{\Delta \vdash_{\mathrm{PT}} A{:}\sigma \quad \Delta \vdash_{\mathrm{PT}} B{:}\phi}{\Delta \vdash_{\mathrm{PT}} A{\to}B{:}\sigma{\to}\phi} \qquad (\mathsf{Type\ Object}), A \equiv [\ell_i{:}B_i{}^{(i\,\in\,I)}]:\ \frac{\Delta \vdash_{\mathrm{PT}} A{:}\sigma \quad \Delta \vdash_{\mathrm{PT}} B_j{:}\phi}{\Delta \vdash_{\mathrm{PT}} A{:}\langle \ell_j{:}\sigma{\to}\phi\rangle}\,(j \in I)$$

$$(\mathsf{Type\ Rec}):\ \frac{\Delta \vdash_{\mathrm{PT}} \mu X.A{:}\sigma \quad \Delta, X{:}\sigma \vdash_{\mathrm{PT}} A{:}\phi}{\Delta \vdash_{\mathrm{PT}} \mu X.A{:}\phi} \qquad (\omega):\ \frac{\Delta \vdash_{\mathrm{PT}} \diamond \quad \widehat{\Delta} \vdash_{\mathrm{T}} A}{\Delta \vdash_{\mathrm{PT}} A{:}\omega} \qquad (\wedge):\ \frac{\Delta \vdash_{\mathrm{PT}} A{:}\phi_i \quad (\forall i \in I)}{\Delta \vdash_{\mathrm{PT}} A{:}\bigwedge_{i\in I}\phi_i}$$

The proof of *(iii)* is straightforward. About *(iv)* we note that $\langle \ell_i{:}\phi_i{}^{i\in I}\rangle \equiv \bigwedge_{i\in I} \langle \ell_i{:}\phi_i\rangle$, and that the meet operation $\wedge$ is monotonic w.r.t. $\leq$ (part *(i)* of this Lemma). ■

The first part of this lemma implies that to be a trivial predicate is decidable. The subsequent part says that $\omega{\to}\omega$ and $\langle \ell{:}\omega\rangle$ are the largest non trivial predicates among arrow and record predicates (with a certain label ℓ) respectively. The last part claims that, with respect to $\leq$, record predicates mirror record subtyping in width and in depth.

Lemma 2.4 *i) If $\sigma \leq \tau{\to}\phi$, then there are I and σ_i, ψ_i for every $i \in I$, such that $\sigma = \bigwedge_{i\in I}(\sigma_i{\to}\psi_i)$ and there exists $J \subseteq I$ such that both $\bigwedge_{j\in J}\sigma_j \geq \tau$ and $\bigwedge_{j\in J}\psi_j \leq \phi$;*

ii) if $\sigma \leq \langle \ell_j{:}\psi_j{}^{j\in J}\rangle$ then there exists $I \supseteq J$ and ϕ_i for every $i \in I$, such that $\sigma = \langle \ell_i{:}\phi_i{}^{i\in I}\rangle$ and $\phi_j \leq \psi_j$ for all $j \in J$;

iii) For all σ, τ, $\sigma \leq \tau$ if and only if there are σ_i $(i \in I), \tau_j$ $(j \in J)$ such that $\sigma = \bigwedge_{i\in I}\sigma_i$, $\tau = \bigwedge_{j\in J}\tau_j$, and, for every $j \in J$, there is an $i \in I$ such that $\sigma_i \leq \tau_j$.

Proof: By induction on the definition of $\leq$ (*2.2*). Note that the statements would become false with $\equiv$ in place of $=$ since equation $\sigma{\wedge}\omega = \sigma$ trivially holds for any σ. ■

Definition 2.5 PREDICATES CONTEXTS. *i*) *Predicate pre-environments* are defined by the following grammar:

$$\Delta ::= \emptyset \mid \Delta, X{:}\sigma \mid \Delta, X{:}\sigma <: A$$

where X is a type variable, σ is a predicate, and A is a pre-type.

ii) By $\widehat{\Delta}$ we denote the pre-environment E obtained from Δ by erasing all predicates:

$$\begin{aligned} \widehat{\emptyset} &\equiv \emptyset, \\ \widehat{\Delta, X{:}\sigma} &\equiv \widehat{\Delta}, X, \\ \widehat{\Delta, X{:}\sigma <: A} &\equiv \widehat{\Delta}, X <: A. \end{aligned}$$

Set $dom(\Delta) = dom(\widehat{\Delta})$.

iii) A *predicate environment* is a predicate pre-environment Δ such that $\Delta \vdash_{\mathrm{PT}} \diamond$ is derivable in the system of Figure *4*, which we call the *predicates to types assignment system*.

iv) We extend the relation $\leq$ as defined on predicates to predicate environments by: $\Delta \leq \Delta'$ if and only if, for every $X{:}\sigma' \in \Delta'$ or $X{:}\sigma' <: A \in \Delta'$ there exists $X{:}\sigma \in \Delta$ or $X{:}\sigma <: A \in \Delta$ respectively, such that $\sigma \leq \sigma'$.

Definition 2.6 ASSIGNMENT OF PREDICATES TO TYPES: LANGUAGES. Let Δ be a predicate environment, A a type and σ a predicate. We write $\Delta \vdash_{\mathrm{PT}} A : \sigma$ if this statement can be derived using the rules of Figure *4*.

Given a closed type A we define the *language* of A as the set $\mathcal{L}_A = \{\sigma \in \mathcal{P} \mid \emptyset \vdash_{\mathrm{PT}} A : \sigma\}$.

Notice that, in the definition of the system as in Figure *4*, $\sigma, \tau \in \mathcal{P}$ while $\phi \in \mathcal{P}_{\mathrm{S}}$: otherwise one could assign to types predicates which are not in $\mathcal{P}$ at all. As stated in Definition *1.1*, the basic type K in rule (Type Const) ranges over a countable set $\mathcal{K}$ of type constants: the assignment of the atoms κ to their types K is assumed to be fixed by a signature we do not make explicit.

As it is apparent from Definition *2.6*, we are essentially interested in closed types and their languages. The reason why we introduce environments Δ in the system is for a proper handling of assumptions of predicates assigned to type variables, which may occur in recursive types. This parallels the usage of environments E in the type system for $\vdash_{\mathrm{T}}$.

The logical interpretation of types we are proposing is as collections of predicates, closed under conjunction (by rule $(\wedge)$) and logical implication (by the admissibility of rule $(\leq)$ in Lemma *2.14*): that is types are seen as propositional theories.

Since predicates are properties of terms, which in turn are polymorphic-typed entities, the soundness criterion we have in mind for the judgements $A{:}\sigma$ is that there exists some term of type A satisfying σ: this is what we mean by saying that σ makes sense of entities of

type A. We observe that this is the very departure of the present work from the endogenous logic of [4]: while there the logical interpretation of polymorphism is not considered (but for the limited case of recursive types), so that the logical and denotational interpretations of non isomorphic types are pair-wise disjoint, the present construction allows for proper inclusions and non-empty intersections of languages.

Example 2.7 Let $A \equiv [\ell_0{:}\mathit{Int}, \ell_1{:}\mathit{Int}]$, and suppose that $\mathsf{O}, \mathsf{E} \in \mathcal{L}_{\mathit{Int}}$ are the predicates of being odd and even integer respectively. Then we can derive $\vdash_{\mathrm{PT}} A : \langle \ell_0{:}\omega{\to}\mathsf{O}\rangle$ as follows (while omitting some obvious inferences):

$$\dfrac{\dfrac{\dfrac{}{\emptyset \vdash_{\mathrm{PT}} \diamond} \quad \vdash_{\mathrm{T}} A}{\vdash_{\mathrm{PT}} A : \omega}\,(\omega) \qquad \vdash_{\mathrm{PT}} \mathit{Int} : \mathsf{O}}{\vdash_{\mathrm{PT}} A : \langle \ell_0{:}\omega{\to}\mathsf{O}\rangle}\,(\text{Type Object})$$

Once this is given we can derive more complex statements like:

$$\dfrac{\vdash_{\mathrm{PT}} A : \langle \ell_0{:}\omega{\to}\mathsf{E}\rangle \qquad \dfrac{\vdash_{\mathrm{PT}} A : \langle \ell_0{:}\omega{\to}\mathsf{O}\rangle \qquad \vdash_{\mathrm{PT}} \mathit{Int} : \mathsf{O}}{\vdash_{\mathrm{PT}} A : \langle \ell_1{:}\langle \ell_0{:}\omega{\to}\mathsf{O}\rangle{\to}\mathsf{O}\rangle}\,(\text{Type Object})}{\vdash_{\mathrm{PT}} A : \langle \ell_0{:}\omega{\to}\mathsf{E}, \ell_1{:}\langle \ell_0{:}\omega{\to}\mathsf{O}\rangle{\to}\mathsf{O}\rangle}\,(\wedge)$$

where the derivation of $\vdash_{\mathrm{PT}} A : \langle \ell_0{:}\omega{\to}\mathsf{E}\rangle$ is similar to that of $\vdash_{\mathrm{PT}} A : \langle \ell_0{:}\omega{\to}\mathsf{O}\rangle$.

The predicate $\langle \ell_0{:}\omega{\to}\mathsf{E}, \ell_1{:}\langle \ell_0{:}\omega{\to}\mathsf{O}\rangle{\to}\mathsf{O}\rangle$ is satisfied by any object having at least methods labelled by ℓ_0 and ℓ_1 (that we identify with the respective methods), where ℓ_0 returns an even number, no matter which is the actual state of the object; ℓ_1 returns and odd integer provided that ℓ_0 does. This makes sense, however, and so it is not contradictory, since the second conjunct of the predicate is only a conditional. Moreover, this is essential for handling method overriding: see Example *4.4*.

Concerning recursive types, consider the following derivation:

$$\dfrac{\dfrac{\vdash_{\mathrm{T}} \mu X.X{\to}X}{\vdash_{\mathrm{PT}} \mu X.X{\to}X : \omega}\,(\omega) \qquad \dfrac{X{:}\omega \vdash_{\mathrm{PT}} X : \omega \qquad X{:}\omega \vdash_{\mathrm{PT}} X : \omega}{X{:}\omega \vdash_{\mathrm{PT}} X{\to}X : \omega{\to}\omega}\,(\text{Type Arrow})}{\vdash_{\mathrm{PT}} \mu X.X{\to}X : \omega{\to}\omega}\,(\text{Type Rec})$$

From this it is then not difficult to see that $(\omega{\to}\omega){\to}(\omega{\to}\omega) \in \mathcal{L}_{\mu X.X\to X}$; but also the unbalanced unfolds (on the predicate side) $(\omega{\to}\omega){\to}\omega$ and $\omega{\to}(\omega{\to}\omega)$ are in $\mathcal{L}_{\mu X.X\to X}$, e.g.:

$$\dfrac{\dfrac{\vdash_{\mathrm{PT}} \mu X.X{\to}X : \omega{\to}\omega \qquad \dfrac{\dfrac{X{:}\omega{\to}\omega \vdash_{\mathrm{PT}} X : \omega{\to}\omega \qquad \dfrac{X{:}\omega{\to}\omega \vdash_{\mathrm{PT}} \diamond \qquad \dfrac{}{X \vdash_{\mathrm{T}} X}}{X{:}\omega{\to}\omega \vdash_{\mathrm{PT}} X : \omega}\,(\omega)}{X{:}\omega{\to}\omega \vdash_{\mathrm{PT}} X{\to}X : (\omega{\to}\omega){\to}\omega}\,(\mathsf{Type\ Arrow})}{\vdash_{\mathrm{PT}} \mu X.X{\to}X : (\omega{\to}\omega){\to}\omega}\,(\mathsf{Type\ Rec})$$

■

We can naturally link derivations in $\vdash_{\mathrm{PT}}$ to those in $\vdash_{\mathrm{T}}$ via erasure of predicates.

Lemma 2.8 ERASING. *If $\Delta \vdash_{\mathrm{PT}} \diamond$, then $\widehat{\Delta} \vdash_{\mathrm{T}} \diamond$. Similarly, if $\Delta \vdash_{\mathrm{PT}} A : \sigma$, then $\widehat{\Delta} \vdash_{\mathrm{T}} A$.*

Proof: In both cases the proof is an easy induction on the structure of derivations. All cases are trivial, except for when the derivation ends by rule $(\wedge)$ or $(\mathsf{Type\ Rec})$; then the result follows by induction. ■

Lemma 2.9 *i) If $\Delta, X{:}\tau \vdash_{\mathrm{PT}} B : \sigma$, then $X \notin dom(\Delta)$;*

ii) If $\Delta_0, \Delta_1 \vdash_{\mathrm{PT}} B : \sigma$, such that no free type variable in B is declared in Δ_1, then $\Delta_0 \vdash_{\mathrm{PT}} B : \sigma$,

Proof: Easy. ■

The next lemma states some standard properties of $\vdash_{\mathrm{PT}}$, that follow immediately from the rules in Figure *4*.

Lemma 2.10 TYPE PREDICATE GENERATION LEMMA. *Let $\mathcal{D} :: \Delta \vdash_{\mathrm{PT}} A : \sigma$. If $A = \mathsf{Top}$, then $\sigma = \omega$; otherwise, either $\sigma = \omega$, or:*

$(\sigma = \bigwedge_{i\in I}\phi_i)$: *Then, for all $i \in I$ there exist $\mathcal{D}_i :: \Delta \vdash_{\mathrm{PT}} A : \phi_i$, sub-derivations of $\mathcal{D}$;*

$(\omega \neq \sigma \in \mathcal{P}_{\mathrm{S}})$: *a) If $\mathcal{D} :: \Delta \vdash_{\mathrm{PT}} K : \sigma$, then σ is atomic;*

b) if $\mathcal{D} :: \Delta \vdash_{\mathrm{PT}} X : \sigma$ then $X{:}\tau \in \Delta$ and $\tau \leq \sigma$;

c) if $\mathcal{D} :: \Delta \vdash_{\mathrm{PT}} A : \sigma$ and $X \in fv(A)$ then $X{:}\tau \in \Delta$, for some τ;

d) if $\mathcal{D} :: \Delta \vdash_{\mathrm{PT}} A{\to}B : \sigma$ then there exist τ, ϕ such that $\sigma = \tau{\to}\phi$ and $\mathcal{D}_1 :: \Delta \vdash_{\mathrm{PT}} A : \tau$ and $\mathcal{D}_2 :: \Delta \vdash_{\mathrm{PT}} B : \phi$, sub-derivations of $\mathcal{D}$;

e) if $\mathcal{D} :: \Delta \vdash_{\mathrm{PT}} [\ell_i{:}B_i\ ^{(i\,\in\,I)}] : \sigma$ *then for some* $j \in I$, $\sigma = \langle \ell_j{:}\tau{\to}\psi \rangle$ *and there exists both* $\mathcal{D}' :: \Delta \vdash_{\mathrm{PT}} [\ell_i{:}B_i\ ^{(i\,\in\,I)}] : \tau$ *and* $\mathcal{D}'' :: \Delta \vdash_{\mathrm{PT}} B_j : \psi$, *sub-derivations of* $\mathcal{D}$*;*

f) if $\mathcal{D} :: \Delta \vdash_{\mathrm{PT}} \mu X.A : \sigma$, *then there exists* τ *and* $\mathcal{D}' :: \Delta \vdash_{\mathrm{PT}} \mu X.A : \tau$ *and* $\mathcal{D}'' :: \Delta, X{:}\tau \vdash_{\mathrm{PT}} A : \sigma$, *sub-derivations of* $\mathcal{D}$.

Proof: By straightforward induction on the structure of derivations. ■

By *2.10* of the above Lemma it follows immediately that the rule:

$$(\mathsf{Type\ Rec}') : \quad \frac{\Delta \vdash_{\mathrm{PT}} \mu X.A{:}\sigma \quad \Delta, X{:}\sigma \vdash_{\mathrm{PT}} A{:}\tau}{\Delta \vdash_{\mathrm{PT}} \mu X.A{:}\tau}$$

is admissible, where τ is not necessarily strict.

A useful corollary of the previous lemma is stated below:

Lemma 2.11 *If* $\mathcal{D} :: \Delta \vdash_{\mathrm{PT}} [\ell_i{:}B_i\ ^{(i\,\in\,I)}] : \sigma$ *and* $\sigma \neq \omega$ *then for some* $J \subseteq I$, $\sigma = \bigwedge_{j\in J} \sigma_j$ *and for each* j *there exists* H_j *such that* $\sigma_j = \langle \ell_j{:}\tau_h{\to}\psi_h\ ^{(h\,\in\,H_j)} \rangle$ *and* $\mathcal{D}_h :: \Delta \vdash_{\mathrm{PT}} [\ell_i{:}B_i\ ^{(i\,\in\,I)}] : \tau_h$ *and* $\mathcal{D}'_h :: \Delta \vdash_{\mathrm{PT}} B_j : \psi_h$ *for all* $h \in H_j$*; moreover, all these are sub-derivations of* $\mathcal{D}$.

Proof: Immediate by Lemma *2.10*. ■

The next lemma shows that, for object types A and C, if σ is a predicate that we can assign to A, which is a super-type of C, then also σ can also be assigned to C.

Lemma 2.12 *Let* $A \equiv [\ell_i{:}B_i\ ^{(i\,\in\,I)}]$, $C \equiv [\ell_j{:}B_j\ ^{(j\,\in\,J)}]$ *for some* $J \supseteq I$ *and suppose* $\widehat{\Delta} \vdash_{\mathrm{T}} C <: A$. *Then, for any* σ, *if* $\Delta \vdash_{\mathrm{PT}} A : \sigma$, *then* $\Delta \vdash_{\mathrm{PT}} C : \sigma$.

Proof: If $\sigma = \omega$, the thesis is trivial by rule (ω). For $\sigma \neq \omega$, we reason by induction over the structure derivations. Now, $\widehat{\Delta} \vdash_{\mathrm{T}} C <: A$ implies that both C and A are well formed types under $\widehat{\Delta}$.

By Lemma *2.10 (e)* we know that σ is (equivalent to) an intersection of record types of the shape $\langle \ell_i{:}\tau{\to}\phi \rangle$ for some $i \in I \subseteq J$, and that both $\Delta \vdash_{\mathrm{PT}} A : \tau$, for some τ, and $\Delta \vdash_{\mathrm{PT}} B_i : \phi$ are derivable in sub-derivations. Therefore, by induction, we know that $\Delta \vdash_{\mathrm{PT}} C : \tau$. Reconstructing the derivation as the one for $\Delta \vdash_{\mathrm{PT}} A : \sigma$, we obtain $\Delta \vdash_{\mathrm{PT}} C : \sigma$. ■

Lemma 2.13 *If both* $\Delta_0, \Delta_1 \vdash_{\mathrm{PT}} A : \sigma$ *and* $\widehat{\Delta_0} \vdash_{\mathrm{T}} A$, *then* $\Delta_0 \vdash_{\mathrm{PT}} A : \sigma$.

Proof: By induction on the structure of derivations. Since derivations in the Predicates to Types Assignment System mirror derivations in the type and subtype system of the object

calculus in almost all cases, we focus on those rules which are not just a decoration of the type formation rules.

(ω) : Obvious, since we can use $\widehat{\Delta_0} \vdash_{\mathrm{T}} A$ as the premise of the same rule.

$(\wedge)$: By induction.

(Type Top) : Trivial, as the environment Δ does not play any role in this case.

(Type Object) : Then $A \equiv [\ell_i{:}B_i\ ^{(i\,\in\,I)}]$, and $\sigma \equiv \langle \ell_j{:}\tau{\to}\phi\rangle$ for some , $j \in I$, and both $\Delta_0, \Delta_1 \vdash_{\mathrm{PT}} A : \tau$ and $\Delta_0, \Delta_1 \vdash_{\mathrm{PT}} B_j : \phi$. By Lemma *1.13(x)*, we have $\widehat{\Delta_0} \vdash_{\mathrm{T}} B_i$ for all $i \in I$; by induction we have also $\Delta_0 \vdash_{\mathrm{PT}} A : \tau$ and $\Delta_0 \vdash_{\mathrm{PT}} B_j : \phi$, and we get $\Delta_0 \vdash_{\mathrm{PT}} A : \langle \ell_j{:}\tau{\to}\phi\rangle$ by rule (Type Object).

(Type Arrow) : Then $A \equiv B{\to}C, \sigma \equiv \tau{\to}\phi$ and both $\Delta_0, \Delta_1 \vdash_{\mathrm{PT}} B : \tau$ and $\Delta_0, \Delta_1 \vdash_{\mathrm{PT}} C : \phi$. By induction both $\Delta_0 \vdash_{\mathrm{PT}} B : \tau$ and $\Delta_0 \vdash_{\mathrm{PT}} C : \phi$, hence $\Delta_0 \vdash_{\mathrm{PT}} B{\to}C : \tau{\to}\phi$ by rule (Type Arrow).

(Type Rec) : Then $A \equiv \mu X.B$ and both $\Delta_0, \Delta_1 \vdash_{\mathrm{PT}} \mu X.B : \tau$ and $\Delta, X{:}\tau \vdash_{\mathrm{PT}} B : \sigma$. By induction $\Delta_0 \vdash_{\mathrm{PT}} \mu X.B : \tau$; by Lemma *1.13(ix)*, $\widehat{\Delta_0} \vdash_{\mathrm{T}} \mu X.B$ implies $\widehat{\Delta_0}, X \vdash_{\mathrm{T}} B$. Since no free type variable in B is declared in Δ_1, we obtain $\Delta_0, X{:}\tau \vdash_{\mathrm{PT}} B : \sigma$ by Lemma *2.9*. Using the assumption $\Delta_0 \vdash_{\mathrm{PT}} \mu X.B : \tau$, we get the desired $\Delta_0 \vdash_{\mathrm{PT}} \mu X.B : \sigma$ by admissibility of rule (Type Rec′). ■

Lemma 2.14 The following rules are admissible:

(Relevance) :

$$\frac{\Delta, X{:}\tau, \Delta' \vdash_{\mathrm{PT}} A : \sigma \quad \Delta, \Delta' \vdash_{\mathrm{PT}} \diamond}{\Delta, \Delta' \vdash_{\mathrm{PT}} A : \sigma}\ (X \notin \mathit{fv}(A))$$

(Weak) :

$$\frac{\Delta \vdash_{\mathrm{PT}} A : \sigma}{\Delta, X{:}\tau \vdash_{\mathrm{PT}} A : \sigma}\ (X \notin \mathit{dom}(\Delta))$$

(Cut) :

$$\frac{\Delta, X{:}\sigma \vdash_{\mathrm{PT}} B : \tau \quad \Delta \vdash_{\mathrm{PT}} A : \sigma}{\Delta \vdash_{\mathrm{PT}} B\{X \hookleftarrow A\} : \tau}$$

(Cut<:) :

$$\frac{\Delta \vdash_{\mathrm{PT}} A : \sigma \quad \Delta, X{:}\sigma <: C \vdash_{\mathrm{PT}} B : \tau \quad \widehat{\Delta} \vdash_{\mathrm{T}} A <: C}{\Delta \vdash_{\mathrm{PT}} B\{X \hookleftarrow A\} : \tau}$$

$(\leq)$:

$$\frac{\Delta \vdash_{\mathrm{PT}} A : \sigma \quad \Delta' \vdash_{\mathrm{PT}} \diamond}{\Delta' \vdash_{\mathrm{PT}} A : \tau}\ (\sigma \leq \tau, \Delta' \leq \Delta)$$

Proof: By easy induction on the structure of derivations. The proof of admissibility of (Relevance) and (Weak) parallels the arguments in Lemma *1.15*. In the cases of (Cut) and (Cut<:) we use Lemma *4.2*. For rule $(\leq)$, the proof is much as that for Lemma *3.13*. ■

Lemma 2.15 For any pre-environment Δ, pre-type $\mu X.A$, and predicate σ:

$$\Delta \vdash_{\mathrm{PT}} \mu X.A : \sigma \Longleftrightarrow \Delta \vdash_{\mathrm{PT}} A\{X \hookleftarrow \mu X.A\} : \sigma.$$

Proof: $(\Rightarrow)$: for $\sigma = \omega$, the thesis is trivial; if σ is an intersection, the thesis follows by induction; else, by Lemma *2.10(f)*, there exists τ such that $\Delta \vdash_{\mathrm{PT}} \mu X.A : \tau$ and $\Delta, X{:}\tau \vdash_{\mathrm{PT}} A : \sigma$. The thesis now follows from rule (Cut).

$(\Leftarrow)$: Let k be the number of free occurrences of X in A. If $k = 0$ then $A\{X \hookleftarrow \mu X.A\} \equiv A$ and $\Delta, X : \omega \vdash_{\mathrm{PT}} A : \sigma$ would follow by the hypothesis and (Weak); since trivially $\Delta \vdash_{\mathrm{PT}} \mu X.A : \omega$, the thesis follows by rule (Type Rec$'$).
Suppose that $k > 0$. For all $1 \leq i \leq k$ there exists an environment Δ_i and a predicate σ_i such that $\Delta_i \vdash_{\mathrm{PT}} \mu X.A : \sigma_i$, and these are sub-derivations of the given derivation of $\Delta \vdash A\{X \hookleftarrow \mu X.A\} : \sigma$. Since $\mathit{fv}(\mu X.A) = \mathit{fv}(A) \setminus \{X\} = \mathit{fv}(A\{X \hookleftarrow \mu X.A\})$ we can freely assume that $\Delta_i = \Delta$ for all i. By replacing each of these derivations by the derivation of $\Delta, X : \bigwedge_{j=1}^{k} \sigma_j \vdash X : \sigma_i$ in the given derivation we construct a derivation of $\Delta, X : \bigwedge_{j=1}^{k} \sigma_j \vdash A : \sigma$; on the other hand from $\Delta_i \vdash_{\mathrm{PT}} \mu X.A : \sigma_i$ for all i we have by rule $(\wedge I)$ a derivation of $\Delta \vdash \mu X.A : \bigwedge_{j=1}^{k} \sigma_j$. Then we conclude by rule (Type Rec$'$).

■

We will now show that $\vdash_{\mathrm{PT}}$ is downwards closed for $<:$.

Theorem 2.16 *The following rule is admissible:*

$$(:>) : \quad \frac{\Delta \vdash_{\mathrm{PT}} B : \sigma \quad \widehat{\Delta} \vdash_{\mathrm{T}} A <: B}{\Delta \vdash_{\mathrm{PT}} A : \sigma}$$

Proof: If $\sigma = \omega$, then by Lemma *1.13(v)*, $\widehat{\Delta} \vdash_{\mathrm{T}} A <: B$ implies $\widehat{\Delta} \vdash_{\mathrm{T}} A$, and $\Delta \vdash_{\mathrm{PT}} A : \omega$ is derivable by rule (ω). If σ is an intersection, the result follows by induction. Otherwise, for σ strict, we reason by a principal induction on the derivation of $\widehat{\Delta} \vdash_{\mathrm{T}} A <: B$ and a secondary induction on the derivation of $\Delta \vdash_{\mathrm{PT}} B : \sigma$.

(Sub Refl), (Sub Trans) : The first case is trivial and the second one follows by induction and the transitivity of $\leq$.

(Sub Top) : Then $B \equiv \mathsf{Top}$ and $\sigma = \omega$ by Lemma *2.10*. As above, by Lemma *1.13(v)*, $\widehat{\Delta} \vdash_{\mathrm{T}} A <: B$ implies $\widehat{\Delta} \vdash_{\mathrm{T}} A$, so $\Delta \vdash_{\mathrm{PT}} A : \omega$ is derivable by rule (ω).

(Sub X) : Then $A \equiv X$ and $\widehat{\Delta} \equiv \widehat{\Delta'}, X <: B, \widehat{\Delta''}$ for some Δ', Δ'', and $\widehat{\Delta}$ is a well-formed environment. By Lemma *1.13(vi)* this implies $\widehat{\Delta'} \vdash_{\mathrm{T}} B$. Therefore, by Lemma *2.13* and the assumption, we have $\Delta' \vdash_{\mathrm{PT}} B : \sigma$. Now, since $\sigma \leq \sigma$, by rule (Env $X<:$), we have $\Delta', X{:}\sigma <: B \vdash_{\mathrm{T}} \diamond$, and, by rule (Type $X<:$), $\Delta', X{:}\sigma <: B \vdash_{\mathrm{PT}} X : \sigma$, and the thesis follows by rule (Weak).

(Sub Object) : This is an immediate consequence of Lemma *2.12*.

(Sub Arrow) : Then $A \equiv A' \to A''$, $B \equiv B' \to B''$ and both $\widehat{\Delta} \vdash_{\mathrm{T}} B' <: A'$ and $\widehat{\Delta} \vdash_{\mathrm{T}} A'' <: B''$. Then $\sigma = \rho \to \psi$, $\Delta \vdash_{\mathrm{PT}} B' : \rho$ and $\Delta \vdash_{\mathrm{PT}} B'' : \psi$. By induction $\Delta \vdash_{\mathrm{PT}} A'' : \psi$; on the other hand, since $\Delta \vdash_{\mathrm{PT}} A' : \omega$, so $\Delta \vdash_{\mathrm{PT}} A' \to A'' : \omega \to \psi$. Since $\omega \to \psi \leq \rho \to \psi$, we get $\Delta \vdash_{\mathrm{PT}} A' \to A'' : \sigma$ by rule $(\leq)$.

(Sub Rec$_1$), (Sub Rec$_2$) : These follow by Lemma *2.14* and *2.15*.

(Sub Rec$_3$) : Then $A \equiv \mu X.A'$, $B \equiv \mu Y.B'$. By Lemma *2.10(f)*, there exists σ' and sub-derivations for $\Delta \vdash_{\mathrm{PT}} \mu Y.B' : \sigma'$ and $\Delta, Y{:}\sigma' \vdash_{\mathrm{PT}} B' : \sigma$. By secondary induction, $\Delta \vdash_{\mathrm{PT}} \mu X.A : \sigma'$. Since $X \notin \mathit{dom}(\Delta)$ we have $\Delta, Y{:}\sigma', X{:}\sigma', X <: Y \vdash_{\mathrm{T}} \diamond$. On the other hand, applying rule (Weak) to $\Delta, Y{:}\sigma' \vdash_{\mathrm{PT}} B' : \sigma$ gives $\Delta, Y{:}\sigma', X{:}\sigma' \vdash_{\mathrm{PT}} B' : \sigma$. But we know that $\widehat{\Delta}, X, Y, X <: Y \vdash_{\mathrm{T}} A' <: B'$ is the premise of the last inference in the derivation for $\widehat{\Delta} \vdash_{\mathrm{T}} A <: B$, so by the principal induction $\Delta, Y{:}\sigma', X{:}\sigma' \vdash_{\mathrm{PT}} A' : \sigma$. Now $Y \notin \mathit{fv}(A')$ (otherwise, since $X \not\equiv Y$, $Y \in \mathit{fv}(\mu X.A')$ so that we need $Y \in \mathit{dom}(\Delta)$ for deducing $\Delta \vdash_{\mathrm{PT}} \mu X.A : \sigma'$ by Lemma *2.10(c)*, which is not the case), hence, by Lemma *2.142.14*, $\Delta, X{:}\sigma' \vdash_{\mathrm{PT}} A' : \sigma$. From this and $\Delta \vdash_{\mathrm{PT}} \mu X.A : \sigma'$ we conclude $\Delta \vdash_{\mathrm{PT}} \mu X.A : \sigma$. ■

Corollary 2.17 LANGUAGES AND SUBTYPING. *If $\vdash_{\mathrm{T}} A <: B$ then both A and B are closed types, and $\mathcal{L}_A \supseteq \mathcal{L}_B$. In particular if $\vdash_{\mathrm{T}} \mu X.A$ then $\mathcal{L}_{\mu X.A} = \mathcal{L}_{A\{X \hookleftarrow \mu X.A\}}$.*

Proof: That A and B are closed is an immediate consequence of *(iv)* and *(v)* of Lemma *1.13*, hence $\mathcal{L}_A$ and $\mathcal{L}_B$ are well defined. Then the first part of the thesis follows by Theorem *2.16*. This implies the second part together with the fact that $\vdash_{\mathrm{T}} \mu X.A =: A\{X \hookleftarrow \mu X.A\}$. ■

3 The Assignment System

We now come to the definition of *predicate assignment*, where we associate predicates to typed terms.

Definition 3.1 BASES AND TERM PRE-ENVIRONMENTS. *i*) A *basis* Γ is defined inductively by:

$$\Gamma ::= \emptyset \mid \Gamma, x{:}A{:}\sigma$$

where x is a term variable, A is a pre-type, and σ is a predicate. The *domain* of a basis is defined by: $\mathit{dom}(\Gamma, x{:}A{:}\sigma) = \{x\} \cup \mathit{dom}(\Gamma)$, and $\widehat{\Gamma, x{:}A{:}\sigma} = \widehat{\Gamma}, x{:}A$.

ii) We extend $\leq$ to environments by: $\Gamma \leq \Gamma'$ if and only if, for every $x{:}A{:}\sigma' \in \Gamma'$ there exists $x{:}A{:}\sigma \in \Gamma$ such that $\sigma \leq \sigma'$.

iii) A *term pre-environment* is a pair $\Delta;\Gamma$ such that Δ is a predicate pre-environment and Γ is a basis. We also define $\mathit{dom}(\Delta;\Gamma) = \mathit{dom}(\Delta) \cup \mathit{dom}(\Gamma)$ and $\widehat{\Delta;\Gamma} = \widehat{\Delta}, \widehat{\Gamma}$ (where the comma represents the concatenation of the two sequences).

Definition 3.2 ASSIGNMENT SYSTEM. An *assignment judgement* is a triple $a{:}A : \sigma$ expressing the assignment of a predicate σ to the (pre) term a of (pre) type A. An *assignment sequent* has the shape $\Delta;\Gamma \vdash_{\mathrm{T}} a{:}A : \sigma$ where $\Delta;\Gamma$ is a term pre-environment.

The *Assignment System* to derive assignment sequents is defined in Figure *5*.

We say that a term pre-environment $\Delta;\Gamma$ is a *term environment* if $\Delta;\Gamma \vdash_{\mathrm{PT}} \diamond$ is derivable in the assignment system.

The judgement $a{:}A{:}\sigma$ tells at the same time that a is of type A, and that it satisfies the predicate σ: hence this implies that $A{:}\sigma$ (as this is witnessed by a), which is in fact ensured by the formal system. Since ω is the trivial predicate, the judgement $a{:}A{:}\omega$ is the same as $a{:}A$, i.e. a has type A. This is why we use $\vdash_{\mathrm{O}}$ in the definition of $\vdash_{\mathrm{P}}$. However, we could avoid this explicit composition of systems, at the price of doubling all rules relating predicates to term and type structure, adding an instance of each such rule with all trivial predicates at the right end of the sequents. E.g. in the case of (Val Appl) we would have:

$$(\mathsf{Val\ Appl}\ \omega):\quad \frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A{\to}B : \omega \quad \Delta;\Gamma \vdash_{\mathrm{P}} b{:}A : \omega}{\Delta;\Gamma \vdash_{\mathrm{P}} a(b){:}B : \omega}$$

The actual formulation of rule (ω) avoids such verbose introductions of the trivial predicate, by preserving its meaning of "being a typeable term".

In the case of (Val Object) and of the $(\mathsf{Val\ Update}_i)$, different formulations are reported in the Introduction, which are only apparently stronger of the corresponding rules in Figure *5*: take all the predicates ϕ and ψ to be ω in the discarded parts, just to ensure typeability.

The side condition $\sigma \neq \omega$ of rule $(\mathsf{Val\ Update}_1)$ is decidable by *(ii)* of Lemma *2.3*. It is immaterial w.r.t. subject reduction and subject expansion Theorems *4.3* and *4.6* respectively, in the next section. Indeed, as far as we are concerned with these properties of the system, it could be replaced by the weaker

$$(\mathsf{Val\ Update}'_1):\quad \frac{\widehat{\Delta;\Gamma} \vdash_{\mathrm{O}} a : A \quad \Delta;\Gamma, y{:}A{:}\tau \vdash_{\mathrm{P}} b{:}B_j : \phi}{\Delta;\Gamma \vdash_{\mathrm{P}} (a.\ell_j \Leftarrow \varsigma(y^A)b){:}A : \langle \ell_j{:}\tau{\to}\phi \rangle}\ (j \in I)$$

This can be understood from the observation that the rule expresses that the newly derived predicate does *not* depend on any predicate for a at all; the only thing we need for subject reduction and expansion is that a is well-formed of type A.

On the other hand it is essential for Theorem *6.5* to hold, as well as for the subsequent results. These say that any (closed) term a satisfies a non-trivial predicate if and only if it is convergent, namely reduces to a value. Now the predicate $\langle \ell_j{:}\tau{\to}\phi\rangle$ is non trivial, but $a.\ell_j \Leftarrow \varsigma(y^A)b$ converges only if a does. It is a remarkable fact that this is a combined effect of the type system and of the predicate system: indeed for $a.\ell_j \Leftarrow \varsigma(y^A)b$ to be convergent we also need that a reduces to an object term, having a label ℓ_j. The type system ensures that if a converges, then this will be the case; the assumption that $a{:}A{:}\sigma$ for $\sigma \neq \omega$ implies that a actually converges, even if σ is discarded in the conclusion. This will be formally proved in section *6*.

We are now in place to give a semantics to predicates. It consists in assigning to each predicate σ a set $[\![\sigma]\!]$ of closed terms of the appropriate (closed) types. It is usually called a *realizability* interpretation, in the sense that each term in $[\![\sigma]\!]$ is a realiser of σ, namely an evidence that σ holds of something. A proper reading of Theorem *6.5* is as a soundness theorem for the realizability interpretation. This interpretation should be compared with the interpretation of intersection types into saturated sets in [19].

We write a^A for a closed term a of a closed type A, i.e. such that $\emptyset \vdash_{\mathrm{O}} a : A$ (abbreviated by $\vdash_{\mathrm{O}} a : A$). In the next definition, the set of labels of A is defined as: $\mathit{Label}(A) = \{\ell_i \mid i \in I\}$ 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, $\ell_j \in \mathit{Label}(A)$ and $a \Downarrow [\ell_i = \varsigma(x_i^A)b_i \ {}^{(i \in I)}]$, then $a.\ell(c)$ abbreviates $b_j\{x_j \leftarrowtail c\}$, for any c^A.

Definition 3.3 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:

i) $[\![\omega]\!] = \{a^A \mid A \text{ is a closed type}\}$;
ii) $[\![\sigma{\wedge}\tau]\!] = [\![\sigma]\!] \cap [\![\tau]\!]$;
iii) $[\![\kappa]\!] = \{a^K \mid \kappa \in \mathcal{L}_K \ \&\ \exists v\ [\ \vdash_{\mathrm{P}} v{:}K : \kappa \ \&\ a \Downarrow v\ ]\}$;
iv) $[\![\sigma{\to}\phi]\!] = \{a^{A\to B} \mid \exists x, b\ [\ a \Downarrow (\lambda x^A.b) \ \&\ \forall c^A \in [\![\sigma]\!]\ [\ b\{x \leftarrowtail c\} \in [\![\phi]\!]\ ]\ ]\}$;
v) $[\![\langle \ell{:}\phi\rangle]\!]$ is defined according to the shapes of ϕ:
- $[\![\langle \ell{:}\omega\rangle]\!] = \{a^A \mid \ell \in \mathit{Label}(A) \ \&\ a \Downarrow\}$
- $[\![\langle \ell{:}\sigma{\to}\psi\rangle]\!] = \{a^A \mid \ell \in \mathit{Label}(A) \ \&\ a \Downarrow \ \&\ \forall c^A \in [\![\sigma]\!]\ [\ a.\ell(c) \in [\![\psi]\!]\ ]\}$;
- $[\![\langle \ell{:}\langle \ell'{:}\psi\rangle\rangle]\!] = \emptyset$.

The clause $[\![\langle \ell{:}\langle \ell'{:}\psi\rangle\rangle]\!] = \emptyset$ is consistent with the fact that $\langle \ell{:}\langle \ell'{:}\psi\rangle\rangle$ cannot be assigned to any (well typed) term.

Lemma 3.4 *If $\sigma \neq \omega$ (i.e. it is non trivial), then any $a^A \in [\![\sigma]\!]$ converges.*

Proof: By induction on the definition of $[\![\sigma]\!]$. ■

Lemma 3.5 *If $a^A \in [\![\sigma]\!]$ then for any b^A if $a \xrightarrow{*} b$ or $b \xrightarrow{*} a$ then $b^A \in [\![\sigma]\!]$.*

Proof: By induction on the length of the reduction sequence, of which we only show the proof for the relation $\longrightarrow$. This is proven by induction on the definition of reduction and by cases on σ, of which we show one case:

($a \equiv (\lambda x^C a')a''$ *and* $\sigma \equiv \tau \rightarrow \psi$) : then $(\lambda x^C a')a'' \Downarrow v$ for some v (an abstraction) with certain properties; but this is true if and only if $b \equiv a'\{x \leftarrow a''\} \Downarrow v$, since the reduction is deterministic: hence $b \in [\![\tau \rightarrow \phi]\!]$ if and only if $a \in [\![\tau \rightarrow \phi]\!]$. ■

Lemma 3.6 *If $\sigma \leq \tau$ then $[\![\sigma]\!] \subseteq [\![\tau]\!]$.*

Proof: By easy induction on the definition of $\leq$ using Lemma *2.4*. E.g. suppose $\langle \ell{:}\sigma \rightarrow \phi \rangle \leq \langle \ell{:}\tau \rightarrow \psi \rangle$ because $\tau \leq \sigma$ and $\phi \leq \psi$. By Definition *3.3*, for any $a^A \in [\![\langle \ell{:}\sigma \rightarrow \phi \rangle]\!]$ we have $a \Downarrow$ and $\ell \in \mathit{Label}(A)$; suppose that $c^A \in [\![\tau]\!]$: by induction $c^A \in [\![\sigma]\!]$, so that by hypothesis $a.\ell(c) \in [\![\phi]\!]$. The thesis follows since $[\![\phi]\!] \subseteq [\![\psi]\!]$ again by induction. ■

We turn to the proof theoretic study of our systems, and formulate the link between $\vdash_{\mathrm{PT}}$ and $\vdash_{\mathrm{T}}$, and $\vdash_{\mathrm{P}}$ and $\vdash_{\mathrm{O}}$, which is a conservativity result.

Lemma 3.7 ASSIGNMENT ERASING. *If $\Delta; \Gamma \vdash_{\mathrm{PT}} \diamond$ and $\Delta; \Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$, then $\widehat{\Delta; \Gamma} \vdash_{\mathrm{T}} \diamond$ and $\widehat{\Delta; \Gamma} \vdash_{\mathrm{O}} a : A$.*

Proof: By induction on the structure of derivations using Lemma *2.8*. ■

The following lemma links $\vdash_{\mathrm{P}}$ and $\vdash_{\mathrm{PT}}$.

Lemma 3.8 *If $\Delta; \Gamma, x{:}A{:}\sigma \vdash_{\mathrm{P}} b{:}B : \tau$ then both $\Delta \vdash_{\mathrm{PT}} A : \sigma$ and $\Delta \vdash_{\mathrm{PT}} B : \tau$.*

Proof: By straightforward induction on derivations. It is a consequence of the fact that the assignment $\Delta \vdash_{\mathrm{O}} A : \sigma$ of a predicate to a type under the assumptions in Δ for each judgement $a{:}A : \sigma$ occurring in a derivation is checked by the appropriate rule. ■

We will now show that also $\vdash_{\mathrm{P}}$ is downwards closed for $<:$.

Theorem 3.9 *The following rule is admissible:*

$$(:>):\quad \frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}C : \sigma \quad \widehat{\Delta;\Gamma} \vdash_{\mathrm{O}} a : A \quad \widehat{\Delta} \vdash_{\mathrm{T}} A <: C}{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma}$$

Proof: By induction on the structure of derivations in $\vdash_{\mathrm{P}}$, using Theorem *2.16*. E.g. if the derivation ends by rule (Val Fun):

$$\frac{\Delta;\Gamma, x{:}C'{:}\tau' \vdash_{\mathrm{P}} b{:}D : \psi}{\Delta;\Gamma \vdash_{\mathrm{P}} \lambda x^{C'}.b{:}C'{\to}D : \tau'{\to}\psi}$$

where $\widehat{\Delta} \vdash_{\mathrm{T}} A <: C \equiv C'{\to}D$. By Lemma *1.12(iii)*, $A = A'{\to}A''$, $E \vdash_{\mathrm{T}} C' <: A'$ and $E \vdash_{\mathrm{T}} A'' <: D$. By induction $\Delta;\Gamma, x{:}C'{:}\tau' \vdash_{\mathrm{P}} b{:}A'' : \psi$ hence $\Delta;\Gamma \vdash_{\mathrm{P}} \lambda x^{C'}.b{:}A'{\to}A'' : \tau'{\to}\psi$. Take any σ' such that $\Delta \vdash_{\mathrm{T}} A'{:}\sigma'$ and $\tau' \leq \sigma'$ (which exists, and at worst is ω), then we get $\Delta;\Gamma \vdash_{\mathrm{P}} \lambda x^{C'}.b{:}A'{\to}A'' : \sigma'{\to}\psi$. Since $\sigma'{\to}\psi \leq \tau'{\to}\psi$, we conclude by $(\leq)$. ■

Lemma 3.10 *The following rule is admissible:*

$$\frac{\Delta;\Gamma, x{:}A{:}\tau, \Gamma' \vdash_{\mathrm{P}} a{:}C : \sigma \quad \Delta \vdash_{\mathrm{O}} A' : \tau' \quad \widehat{\Delta} \vdash_{\mathrm{T}} A' <: A}{\Delta;\Gamma, x{:}A'{:}\tau', \Gamma' \vdash_{\mathrm{P}} a{:}C : \sigma}\ (\tau' \leq \tau)$$

Proof: Easy induction on the structure of derivations. Informally, each time there is an instance of the rule (Env x) with conclusion $\Delta;\Gamma, x{:}A{:}\tau, \Gamma'' \vdash_{\mathrm{P}} x{:}A : \tau$ in the derivation for $\Delta;\Gamma, x{:}A{:}\tau, \Gamma' \vdash_{\mathrm{P}} a{:}C : \sigma$, we replace it by an instance of $(<:)$ whose premises are $\Delta;\Gamma, x{:}A'{:}\tau', \Gamma'' \vdash_{\mathrm{P}} x{:}A : \tau'$, $\widehat{\Delta} \vdash_{\mathrm{T}} A' <: A$ and $\Delta \vdash_{\mathrm{O}} A : \tau'$, where the conclusion is $\Delta;\Gamma, x{:}A{:}\tau', \Gamma'' \vdash_{\mathrm{P}} x{:}A : \tau$. ■

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

Lemma 3.11 PREDICATE GENERATION LEMMA. *If $\mathcal{D} :: \Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$, then either: $\sigma = \omega$, or,*

- $(\sigma = \bigwedge_{i\in I}\phi_i)$: *Then, for all $1 \leq i \leq n$ there exist $\mathcal{D}_i :: \Delta \vdash_{\mathrm{P}} a{:}A : \phi_i$, sub-derivations of $\mathcal{D}$;*
- $(\sigma \equiv \phi \in \mathcal{P}_{\mathrm{S}} \setminus \{\omega\})$: *a) if $\Delta;\Gamma \vdash_{\mathrm{P}} c{:}A : \phi$ then c is a constant of some ground type K, with $A \equiv K$, $\phi \equiv \kappa$ and $\emptyset \vdash_{\mathrm{PT}} K : \kappa$;*
  - *b) if $\Delta;\Gamma \vdash_{\mathrm{P}} x{:}A : \phi$ then $\Delta \vdash_{\mathrm{PT}} A : \phi$ and there exists $x{:}C{:}\sigma \in \Gamma$ such that $\widehat{\Delta} \vdash_{\mathrm{T}} C <: A$ and $\sigma \leq \phi$;*

c) *if* $\Delta;\Gamma \vdash_{\mathrm{P}} [\ell_i = \varsigma(x_i^{A_i})b_i\ ^{(i\in I)}]{:}A : \phi$ *then there exists* $C \equiv [\ell_i{:}B_i\ ^{(i\in I)}]$ *for certain* B_i*, such that* $A_i \equiv C$ *for all* $i \in I$*, and there exists* $J \subseteq I$ *such that* $A \equiv [\ell_j{:}B_j\ ^{(j\in J)}]$*; moreover* $\phi \equiv \langle \ell_k{:}\tau{\to}\psi\rangle$ *where* $k \in J$*,* $\Delta;\Gamma, x_k{:}C{:}\tau \vdash_{\mathrm{P}} b_k{:}B_k : \psi$*, while for all* $i \in I \setminus \{k\}$*,* $\widehat{\Delta;\Gamma}, x_i{:}C \vdash_{\mathrm{O}} b_i : B_i$*;*

d) *if* $\Delta;\Gamma \vdash_{\mathrm{P}} a.\ell{:}A : \phi$ *then there exists* $C \equiv [\ldots,\ell{:}B,\ldots]$ *such that* $\widehat{\Delta} \vdash_{\mathrm{T}} B <: A$*, and* τ *such that* $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}C : \langle \ell{:}\tau{\to}\phi\rangle$ *and* $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}C : \tau$*;*

e) *if* $\Delta;\Gamma \vdash_{\mathrm{P}} (a.\ell_j \Leftarrow \varsigma(x^C)b){:}A : \sigma$ *then* $\widehat{\Delta;\Gamma} \vdash_{\mathrm{T}} C <: A$ *and* $C \equiv [\ell_i{:}B_i\ ^{(i\in I)}]$ *for certain* B_i*, with* $j \in I$ *and either:*

- *there are* τ, ψ*,* $\rho \neq \omega$ *such that* $\phi \equiv \langle \ell_j{:}\tau{\to}\psi\rangle$*,* $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}C : \rho$*, and* $\Delta;\Gamma, x{:}C{:}\tau \vdash_{\mathrm{P}} b{:}B_j : \psi$*; or*
- *there exist* ψ *and* $k \neq j$ *such that* $\phi \equiv \langle \ell_k{:}\psi\rangle$*,* $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}C : \phi$*, and* $\widehat{\Delta;\Gamma}, x{:}C \vdash_{\mathrm{O}} b : B_j$*;*

f) *if* $\Delta;\Gamma \vdash_{\mathrm{P}} \lambda x^A.a{:}C : \phi$*, then there exists* B *such that* $\widehat{\Delta} \vdash_{\mathrm{T}} A{\to}B <: C$*, and* $\phi \equiv \tau{\to}\psi$*, and* $\Delta;\Gamma, x{:}A{:}\tau \vdash_{\mathrm{P}} a{:}B : \psi$*;*

g) *if* $\Delta;\Gamma \vdash_{\mathrm{P}} a(b){:}C : \phi$*, then there exist* A, B *such that* $\widehat{\Delta} \vdash_{\mathrm{T}} B <: C$*, and there is* τ *such that both* $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A{\to}B : \tau{\to}\phi$ *and* $\Delta;\Gamma \vdash_{\mathrm{P}} b{:}A : \tau$*.*

Proof: By induction on the structure of derivations. We observe that in all clauses we use $\equiv$ instead of $=:$ (among types) and $=$ (among predicates). This is possible since these are all existential statements of derivability, and do not necessarily refer to sub-derivations of the given one: hence we can choose types and predicates of the right form as we need. In particular in clause *(c)* we have $\phi \equiv \langle \ell_k{:}\tau{\to}\psi\rangle$ instead of $\phi \geq \langle \ell_k{:}\tau{\to}\psi\rangle$ as one might expect. This is a consequence of the fact that languages are upward closed w.r.t. $\leq$ (by lemma *2.14*), and that $E \vdash_{\mathrm{T}} A <: A$ for any A, so that if $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$ and $\sigma \leq \tau$ then $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \tau$ by $(<:)$. Similar remarks apply to all other clauses.

■

Remark 3.12 The last lemma, together with Lemmas *1.14* and *2.10* forms the basic tool for reconstructing types and predicates in terms of the structure of the subject, namely the term. In particular Lemma *2.10* applies the same technique to the "subject" A in the judgement $A : \sigma$.

Their proofs are just backward readings of the derivation rules, and as such are very simple inductions over derivations which we omit.

All the implications in these lemmas are actually equivalences: indeed the opposite implications follow by direct application of (possibly more than one) derivation rules.

Although the relation $\leq$ is only used for variables, we can show the following lemma.

Lemma 3.13 The rule

$$(\leq):\quad \frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma \quad \Delta';\Gamma' \vdash_{\mathrm{T}} \diamond}{\Delta';\Gamma' \vdash_{\mathrm{P}} a{:}A : \tau}\ (\sigma \leq \tau, \Delta' \leq \Delta, \Gamma' \leq \Gamma)$$

is admissible.

Proof: $(\tau \in \mathcal{P}_{\mathrm{S}})$: By induction on the structure of the derivation for $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$.

$(\mathsf{Val}\ x)$: Then $a = x$, $\sigma \in \mathcal{P}_{\mathrm{S}}$, and there exists τ such that $\Gamma = \Gamma_1, x{:}A{:}\rho, \Gamma_2$, $\rho \leq \sigma$ and $\Delta;\Gamma \vdash_{\mathrm{PT}} \diamond$. Since $\Gamma' \leq \Gamma$, there exists $x{:}A{:}\rho' \in \Gamma'$ such that $\rho' \leq \rho$. Notice that then $\rho' \leq \tau$, and by Lemma *2.14* also $\Delta';\Gamma' \vdash_{\mathrm{PT}} \diamond$. Then, by rule $(\mathsf{Val}\ x)$, $\Delta',\Gamma' \vdash_{\mathrm{P}} x{:}A : \tau$.

$(<:)$: Then $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}B : \sigma$, $\widehat{\Delta} \vdash_{\mathrm{T}} B <: A$, and $\Delta \vdash_{\mathrm{PT}} A : \sigma$, for some B. We get $\Delta';\Gamma' \vdash_{\mathrm{P}} a{:}B : \tau]$ by induction; since $\widehat{\Delta'} \subseteq \widehat{\Delta}$, by weakening $\widehat{\Delta'} \vdash_{\mathrm{T}} B <: A$; and by Lemma *2.14* $\Delta' \vdash_{\mathrm{PT}} A : \tau$, Then, by rule $(<:)$ we get $\Delta';\Gamma' \vdash_{\mathrm{P}} a{:}A : \tau$.

$(\mathsf{Val\ Fun})$: Then $\sigma = \rho{\to}\phi$, $a = \lambda x^A.a'$, and $\Delta;\Gamma, x{:}A{:}\rho \vdash_{\mathrm{P}} a'{:}B : \phi$ for some B. Since $\sigma \leq \tau \in \mathcal{P}_{\mathrm{S}}$, $\tau = \rho'{\to}\phi'$ with $\rho' \leq \rho, \phi \leq \phi'$, so, by induction, $\Delta';\Gamma', x{:}A{:}\rho' \vdash_{\mathrm{P}} a'{:}B : \phi'$. Then, by rule $(\mathsf{Val\ Fun})$, $\Delta';\Gamma' \vdash_{\mathrm{P}} \lambda x^A.a'{:}A{\to}B : \rho'{\to}\phi'$.

$(\mathsf{Val\ Appl})$: Then $a = a_1 a_2$, and there exists ρ such that $\Delta;\Gamma \vdash_{\mathrm{P}} a_1{:}A{\to}B : \rho{\to}\sigma$ and $\Delta;\Gamma \vdash_{\mathrm{P}} a_2{:}A : \rho$. Since $\rho{\to}\sigma \leq \rho{\to}\tau$, by induction, $\Delta';\Gamma' \vdash_{\mathrm{P}} a_1{:}A{\to}B : \rho{\to}\tau$ and $\Delta';\Gamma' \vdash_{\mathrm{P}} a_2{:}A : \rho$. Then, by rule $(\mathsf{Val\ Appl})$, also $\Delta';\Gamma' \vdash_{\mathrm{P}} a(b){:}B : \tau$.

$(\mathsf{Val\ Object})$, $(\mathsf{Val\ Update}_1)$: As for rule $(\mathsf{Val\ Fun})$.

$(\mathsf{Val\ Select})$: As for rule $(\mathsf{Val\ Appl})$.

$(\mathsf{Val\ Update}_2)$, $(\wedge I)$: By induction, using *(ii)* and *(iii)* of Lemma *2.4* respectively.

(ω) : By assumption, $\Delta';\Gamma' \vdash_{\mathrm{PT}} \diamond$; since $\widehat{\Delta',\Gamma'} \subseteq \widehat{\Delta;\Gamma}$, by weakening also $\widehat{\Delta';\Gamma'} \vdash_{\mathrm{O}} a : A$. Then, by rule (ω), $\Delta';\Gamma' \vdash_{\mathrm{P}} a{:}A : \omega$.

$(\tau = \bigwedge_{j\in J}\tau_j)$: Assume, without loss of generality, that $\sigma = \bigwedge_{i\in I}\sigma_i$; then, by *(iii)* of Lemma *2.4*, for every $j \in J$ there is an $i \in I$ such that $\sigma_i \leq \tau_j \in \mathcal{P}_{\mathrm{S}}$. The result follows by the first part of the proof, and applying rule $(\wedge I)$.

■

4 Subject Reduction and Expansion

In this section we will show that predicate assignment as defined above is not only preserved by reduction, but also by expansion, i.e. if $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$ and we can relate a to a' via the reduction system, then also $\Delta;\Gamma \vdash_{\mathrm{P}} a'{:}A : \sigma$.

The following lemma concerns properties of the Object Calculus whose proofs are straightforward inductions over derivations.

Lemma 4.1 *i) Let* $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$*, and* $\Gamma' = \{x{:}A{:}\tau \in \Gamma \mid x \in \mathit{fv}(a)\}$*, then* $\Delta;\Gamma' \vdash_{\mathrm{P}} a{:}A : \sigma$*,*
ii) if $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$*, and* $x \in \mathit{fv}(a)$*, then* $x \in \mathit{dom}(\Gamma)$*.*

Proof: Easy. ■

In the remaining part of this section we show that predicate assignment is closed for reduction and expansion. First we establish a substitution lemma.

Lemma 4.2 SUBSTITUTION LEMMA FOR $\vdash_{\mathrm{P}}$. *If* $\Delta;\Gamma, x{:}A{:}\sigma \vdash_{\mathrm{P}} b{:}B : \tau$ *and* $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$*, then*
$\Delta;\Gamma \vdash_{\mathrm{P}} b\{x \hookleftarrow a\}{:}B : \tau$*.*

Proof: By straightforward induction on the structure of derivations, of which we show only the interesting cases; the others follow by easy induction.

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

$(b = x)$: Then $\sigma \leq \tau$. Since $x\{x \hookleftarrow a\} = a$, the result follows from the second assumption and Lemma *3.13*.

$(b = y \neq x)$: Since $y\{x \hookleftarrow a\} = y$, and $\Delta;\Gamma, x{:}A{:}\sigma \vdash_{\mathrm{P}} y{:}B : \tau$, by Lemma *4.1 (i)* we obtain $\Delta;\Gamma \vdash_{\mathrm{P}} y{:}B : \tau$.

(ω) : Then $\widehat{\Delta;\Gamma}, x{:}A \vdash_{\mathrm{O}} b : B$. Since by Lemma *1.16*, $\widehat{\Delta;\Gamma} \vdash_{\mathrm{O}} b\{x \hookleftarrow a\} : B$, we can apply rule (ω) to get $\Delta;\Gamma \vdash_{\mathrm{P}} b\{x \hookleftarrow a\}{:}B : \omega$.

$(\wedge I)$: Then $\tau = \bigwedge_{i\in I}\tau_i$, and, for $i \in I$, $\Delta;\Gamma, x{:}A{:}\sigma \vdash_{\mathrm{P}} b{:}B : \tau_i$. Then $\Delta;\Gamma \vdash_{\mathrm{P}} b\{x \hookleftarrow a\}{:}B : \tau_i$ follows by induction, and, by rule $(\wedge I)$, $\Delta;\Gamma \vdash_{\mathrm{P}} b\{x \hookleftarrow a\}{:}B : \bigwedge_{i\in I}\tau_i$. ■

We use this lemma to show the following result.

Theorem 4.3 SUBJECT REDUCTION FOR $\vdash_{\mathrm{P}}$. *If* $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$*, and* $a \longrightarrow a'$*, then*
$\Delta;\Gamma \vdash_{\mathrm{P}} a'{:}A : \sigma$*.*

Proof: By induction on the length of the reduction sequence, of which we only show the base case, which is by definition on the reduction relation $\longrightarrow$. First we deal with $\omega \neq \sigma \in \mathcal{P}_{\mathrm{S}}$.

$((\lambda x^D.a)(b) \longrightarrow a\{x \hookleftarrow b\})$: by Lemma *3.11*, there exist B, C, E, τ such that both $\widehat{\Delta} \vdash_{\mathrm{T}} B <: A$ and $\widehat{\Delta} \vdash_{\mathrm{T}} D{\to}E <: C{\to}B$, and such that both $\Delta;\Gamma, x{:}D{:}\tau \vdash_{\mathrm{P}} a{:}E : \phi$, and $\Delta;\Gamma \vdash_{\mathrm{P}} b{:}C : \tau$. Notice that $C <: D$, so also $\Delta;\Gamma \vdash_{\mathrm{P}} b{:}D : \tau$; the result then follows from Lemma *4.2* and rule $(<:)$.

$([\ell_i = \varsigma(x_i^{A_i})b_i\ ^{(i\in I)}].\ell_j \longrightarrow b_j\{x_j \leftarrowtail [\ell_i = \varsigma(x_i^{A_i})b_i\ ^{(i\in I)}]\}) :$ as the previous part.

$([\ell_i = \varsigma(x_i^{D_i})b_i\ ^{(i\in I)}].\ell_j \Leftarrow \varsigma(x^C)b \longrightarrow [\ell_i = \varsigma(x_i^{D_i})b_i{}^{i\in I\backslash j}, \ell_j = \varsigma(x^C)b]) :$

by Lemma *3.11*, $j \in I$, $\widehat{\Delta} \vdash_{\mathrm{T}} C <: A$ and $C \equiv [\ell_i{:}C_i\ ^{(i\in I)}]$ and either:

- there are $\tau, \psi, \rho \neq \omega$ such that $\sigma \equiv \langle \ell_j{:}\tau{\to}\psi\rangle$, and both
  (1) $\Delta;\Gamma \vdash_{\mathrm{P}} [\ell_i = \varsigma(x_i^{D_i})b_i\ ^{(i\in I)}]{:}C : \rho$, and
  (2) $\Delta;\Gamma, x_j{:}C{:}\tau \vdash_{\mathrm{P}} b_j{:}C_j : \psi$.
  Then from (1), by Lemma *1.13*, we have that there exist $E \equiv [\ell_i{:}E_i\ ^{(i\in I)}]$ such that $\widehat{\Delta;\Gamma} \vdash_{\mathrm{T}} E <: C$, and, for all $i \in I$, $E \equiv D_i$ and $\widehat{\Delta;\Gamma}, x_i{:}E \vdash_{\mathrm{O}} b_i : E_i$. Notice that, by Lemma *3.10*, since $E <: C$ we also have $\Delta;\Gamma, x_j{:}E{:}\tau_j \vdash_{\mathrm{P}} b_j{:}E_j : \psi_j$. Then the result follows by rules (Val Object)

$$\frac{\Delta;\Gamma, x_j{:}E{:}\tau \vdash_{\mathrm{P}} b_j{:}E_j : \psi \quad \widehat{\Delta;\Gamma}, x_i{:}E \vdash_{\mathrm{O}} b_i : E_i \quad (\forall i \in I\backslash j)}{\Delta;\Gamma \vdash_{\mathrm{P}} [\ell_i = \varsigma(x_i^{D_i})b_i{}^{i\in I\backslash j}, \ell_j = \varsigma(x^C)b]{:}E : \langle \ell_j{:}\tau{\to}\psi\rangle}\ (j \in I)$$

  and $(<:)$, since $E <: C <: A$.
- $\sigma \equiv \langle \ell_j{:}\phi\rangle$, for some $j \in I$, and we have both (1) $\Delta;\Gamma \vdash_{\mathrm{P}} [\ell_i = \varsigma(x_i^{D_i})b_i\ ^{(i\in I)}]{:}C : \sigma$, and (2) $\widehat{\Delta;\Gamma}, x{:}C \vdash_{\mathrm{O}} b : C_k$ for some $j \neq k \in I$. From (1), by Lemma *3.11*, there exist $E \equiv [\ell_i{:}E_i\ ^{(i\in I)}]$ such that $\widehat{\Delta;\Gamma} \vdash_{\mathrm{T}} E <: C$ and $E \equiv D_i$ and $\widehat{\Delta;\Gamma}, x_i{:}E \vdash_{\mathrm{O}} b_i : E_i$ for all $i \in I\backslash j$, and there exists τ, ψ such that $\phi \equiv \tau{\to}\psi$, and $\Delta;\Gamma, x_k{:}E{:}\tau \vdash_{\mathrm{P}} b_k{:}E_k : \psi$. Also, from (2), we get by Lemma *2.16* and $(<:)$ that $\widehat{\Delta;\Gamma}, x{:}E \vdash_{\mathrm{O}} b : E_j$.
  Then the result follows by rules (Val Object)

$$\frac{\Delta;\Gamma, x_k{:}E{:}\tau \vdash_{\mathrm{P}} b_k{:}E_k : \psi \quad \widehat{\Delta;\Gamma}, x_i{:}E \vdash_{\mathrm{O}} b_i : E_i \quad (\forall i \in I\backslash k, j) \quad \widehat{\Delta;\Gamma}, x{:}E \vdash_{\mathrm{O}} b : E_j}{\Delta;\Gamma \vdash_{\mathrm{P}} [\ell_i = \varsigma(x_i^{D_i})b_i{}^{i\in I\backslash j}, \ell_j = \varsigma(x^E)b]{:}E : \langle \ell_j{:}\tau{\to}\psi\rangle}\ (j \in I)$$

  and $(<:)$, since $E <: C <: A$.

$(a \longrightarrow b \ \Rightarrow\ \mathcal{E}[a] \longrightarrow \mathcal{E}[b]) :$ By induction on the structure of evaluating contexts.

For $\sigma = \omega$, the result follows from Lemma *3.7*, Theorem *1.17* and rule (ω). For $\sigma = \bigwedge_{i\in I}\sigma_i$, the result follows by the strict case, and rule $(\wedge I)$. ■

Example 4.4 To better appreciate the importance of this standard result in the present setting, let us look at the following example.

Suppose that $A \equiv [\ell_0{:}\mathit{Int}, \ell_1{:}\mathit{Int}]$ as in Example *2.7*, where the predicates in $\mathcal{L}_A$ to be used below have been derived. Moreover let $a \equiv [\ell_0 = \varsigma(x^A)1, \ell_1 = \varsigma(x^A)x.\ell_0]$ (using a constant 1 of type *Int*), so that we have $\vdash_{\mathrm{O}} a : A$ is derivable in the Object Calculus. Then

(ignoring the obvious parts):

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

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

$$\dfrac{\vdash_{\mathrm{T}} a{:}A{:}\langle \ell_0{:}\omega{\to}\mathsf{O}, \ell_1{:}\langle \ell_0{:}\omega{\to}\mathsf{O}\rangle{\to}\mathsf{O}\rangle \qquad y{:}A{:}\omega \vdash_{\mathrm{P}} 2{:}\mathit{Int} : \mathsf{E}}{\vdash_{\mathrm{P}} (a.\ell_0 \Leftarrow \varsigma(y^A)2){:}A : \langle \ell_0{:}\omega{\to}\mathsf{E}, \ell_1{:}\langle \ell_0{:}\omega{\to}\mathsf{O}\rangle{\to}\mathsf{O}\rangle}\,(\textsf{Val Update}_i, i = 1, 2, \wedge I)$$

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 *4.3* 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 $\not\vdash 2{:}\mathit{Int}{:}\mathsf{O}$, so by contraposition $\not\vdash (a.\ell_0 \Leftarrow \varsigma(y^A)2).\ell_1{:}\mathit{Int}{:}\mathsf{O}$. As a matter of fact, rule (Val Select) is not applicable, since $\not\vdash (a.\ell_0 \Leftarrow \varsigma(y^A)2){:}A{:}\langle \ell_0{:}\omega{\to}\mathsf{O}\rangle$.

On the other hand, the following odd-looking assignment is legal as well:

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

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

$$\dfrac{\vdash_{\mathrm{P}} a{:}A : \langle \ell_0{:}\omega{\to}\mathsf{O}, \ell_1{:}\langle \ell_0{:}\omega{\to}\mathsf{E}\rangle{\to}\mathsf{E}\rangle \qquad y{:}A{:}\omega \vdash_{\mathrm{P}} 2{:}\mathit{Int} : \mathsf{E}}{(a.\ell_0 \Leftarrow \varsigma(y^A)2) \vdash_{\mathrm{O}} A : \langle \ell_0{:}\omega{\to}\mathsf{E}, \ell_1{:}\langle \ell_0{:}\omega{\to}\mathsf{E}\rangle{\to}\mathsf{E}\rangle}\,(\textsf{Val Update}_i, i = 1, 2, \wedge I)$$

which is what we expected.

We now come to the proof that predicate assignment is closed for subject expansion as well. With respect to the subject reduction property there is an asymmetry, since the expansion property does not hold, in general, in $\vdash_{\mathrm{O}}$. It might seem to be contradictory w.r.t. the conservativity established in Lemma *3.7*, but it is not: remember that we assign predicates to typeable terms, while the property which we are going to establish concerns the predicates, and not the types. Therefore, we do not just assume that $a' \longrightarrow a$, rather also that both have the same type A, and show that for any predicate σ such that $a{:}A{:}\sigma$ can be derived in a suitable environment, also $a'{:}A{:}\sigma$ can be derived in the same environment. This could be called a *typed subject expansion* property.

We need the following lemma.

Lemma 4.5 EXPANSION LEMMA FOR $\vdash_{\mathrm{P}}$. *Assume* $\Delta;\Gamma \vdash_{\mathrm{P}} b\{x \hookleftarrow a\}{:}B : \tau$, *and both* $\widehat{\Delta;\Gamma}, x{:}A \vdash_{\mathrm{O}} b : B$ *and* $\widehat{\Delta;\Gamma} \vdash_{\mathrm{O}} a : A$ *for some* A. *Then there exist* σ *such that* $\Delta;\Gamma, x{:}A{:}\sigma \vdash_{\mathrm{P}} b{:}B : \tau$ *and* $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$.

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 $\omega \neq \tau \in \mathcal{P}_{\mathrm{S}}$.

$(b = x)$: Since $x\{x \hookleftarrow a\} = a$, we get $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}B : \tau$. From $\widehat{\Delta;\Gamma}, x{:}A \vdash_{\mathrm{O}} x : B$, by Lemma *3.11*, we get $A <: B$. Then, from $\widehat{\Delta;\Gamma} \vdash_{\mathrm{O}} a : A$ and $A <: B$, by Theorem *3.9*, we get $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \tau$. Take $\sigma = \tau$, then also by rule $(<:)$, we have $\Delta;\Gamma, x{:}A{:}\tau \vdash_{\mathrm{P}} x{:}B : \tau$.

$(b = y \neq x)$: Since $y\{x \hookleftarrow a\} = y$, we get $\Delta;\Gamma \vdash_{\mathrm{P}} y{:}B : \tau$, and, by Lemma *4.1*, $\Delta;\Gamma, x{:}A{:}\omega \vdash_{\mathrm{P}} y{:}B : \tau$. Notice that, from the fact that $\widehat{\Delta;\Gamma} \vdash_{\mathrm{O}} a : A$, we get, by rule (ω), $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \omega$.

$(b = c.\ell \Leftarrow \varsigma(y^C)d)$: If $\Delta;\Gamma \vdash_{\mathrm{P}} (c.\ell \Leftarrow \varsigma(y^C)d)\{x \hookleftarrow a\}{:}B : \tau$ then, by the definition of substitution, $\Delta;\Gamma \vdash_{\mathrm{P}} c\{x \hookleftarrow a\}.\ell \Leftarrow \varsigma(y^C)d\{x \hookleftarrow a\}{:}B : \tau$. From $\widehat{\Delta;\Gamma}, x{:}A \vdash_{\mathrm{O}} c.\ell \Leftarrow \varsigma(y^C)d : B$, by Lemma *1.13* we have both $\widehat{\Delta;\Gamma}, x{:}A \vdash_{\mathrm{O}} c : B$ and $\widehat{\Delta;\Gamma}, x{:}A \vdash_{\mathrm{O}} d : D$, for some D.
Also by Lemma *3.11*, $\widehat{\Delta} \vdash_{\mathrm{T}} C <: B$ and $C \equiv [\ell_i{:}C_i\ ^{(i \in I)}]$ with $j \in I$ and either:

- $\tau \equiv \langle \ell_j{:}\tau' \rightarrow \psi \rangle$, and we have $\Delta;\Gamma \vdash_{\mathrm{P}} c\{x \hookleftarrow a\}{:}C : \rho$ and, for some ρ, μ, τ', ψ, $\Delta;\Gamma, x{:}C{:}\mu \vdash_{\mathrm{P}} d\{x \hookleftarrow a\}{:}D : \psi$. By induction, $\Delta;\Gamma, x{:}A{:}\sigma_1 \vdash_{\mathrm{P}} c{:}C : \alpha$ and $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}D : \sigma_1$ for some σ_1.
- $\tau \equiv \langle \ell_j{:}\phi \rangle$, and both $\Delta;\Gamma \vdash_{\mathrm{P}} c\{x \hookleftarrow a\}{:}C : \tau$ and $\Delta;\Gamma, x{:}C{:}\mu \vdash_{\mathrm{P}} d\{x \hookleftarrow a\}{:}D : \psi$ for some ϕ, ψ. By induction, there exists σ_2 such that $\Delta;\Gamma, x{:}A{:}\sigma_2 \vdash_{\mathrm{P}} d{:}C : \alpha$ and $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}D : \sigma_2$.

In either case, take $\sigma = \sigma_1 \wedge \sigma_2$, then, by either rule (Val Update$_1$) or (Val Update$_2$), by Lemma *4.1* we get $\Delta; \Gamma, x{:}A{:}\sigma \vdash_{\mathrm{P}} (c.\ell \Leftarrow \varsigma(y^C)d){:}B : \tau$ and by rule $(\wedge I)$, $\Delta; \Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$, and the result follows from rule $(<:)$.

$(b = c(d))$: If $\Delta; \Gamma \vdash_{\mathrm{P}} c\{x \leftarrowtail a\}(d\{x \leftarrowtail a\}){:}B : \tau$, and, by Lemma *3.11* there exists ρ, C, $A <: B$ such that $\Delta; \Gamma \vdash_{\mathrm{P}} c\{x \leftarrowtail a\}{:}C \rightarrow A : \rho \rightarrow \tau$ and $\Delta; \Gamma \vdash_{\mathrm{P}} d\{x \leftarrowtail a\}{:}C : \sigma$. From the assumption $\widehat{\Delta; \Gamma}, x{:}A \vdash_{\mathrm{O}} c(d) : B$, by Lemma *1.13*, $\widehat{\Delta; \Gamma}, x{:}A \vdash_{\mathrm{O}} c : C \rightarrow A$ and $\widehat{\Delta; \Gamma}, x{:}A \vdash_{\mathrm{O}} d : C$. Then, by induction, there exists σ_1, σ_2 such that $\Delta; \Gamma, x{:}A{:}\sigma_2 \vdash_{\mathrm{P}} d{:}C : \rho$ and $\Delta; \Gamma, x{:}A{:}\sigma_1 \vdash_{\mathrm{P}} c{:}C \rightarrow A : \rho \rightarrow \tau$ and $\Delta; \Gamma \vdash_{\mathrm{P}} a{:}A : \sigma_1$, and $\Delta; \Gamma \vdash_{\mathrm{P}} a{:}A : \sigma_2$. Then by Lemma *4.1* and rule (Val Appl) we get $\Delta; \Gamma, x{:}A{:}\sigma_1 \wedge \sigma_2 \vdash_{\mathrm{P}} c(d){:}A : \tau$ and by rule $(\wedge I)$, $\Delta; \Gamma \vdash_{\mathrm{P}} a{:}A : \sigma_1 \wedge \sigma_2$.

■

Theorem 4.6 SUBJECT EXPANSION FOR $\vdash_{\mathrm{P}}$. *If $\Delta; \Gamma \vdash_{\mathrm{P}} a{:}A : \tau$, and a' is such that $\widehat{\Delta; \Gamma} \vdash_{\mathrm{O}} a' : A$ and $a' \longrightarrow a$, then $\Delta; \Gamma \vdash_{\mathrm{P}} a'{:}A : \tau$.*

Proof: By induction on the length of the reduction sequence, of which we only show the base case, which is by definition on the reduction relation $\longrightarrow$. Most cases depend straightforwardly on Lemma *4.5*; we show one case, that does not. First we deal with $\omega \neq \tau \in \mathcal{P}_{\mathrm{S}}$.

(i) : $[\ell_i = \varsigma(x_i^{B_i})b_i{}^{(i \in I)}].\ell_j \Leftarrow \varsigma(x_j^C)b_j \longrightarrow [\ell_i = \varsigma(x_i^{B_i})b_i{}^{i \in I \setminus j}, \ell_j = \varsigma(y^C)b]$. If

$$\Delta; \Gamma \vdash_{\mathrm{P}} [\ell_i = \varsigma(x_i^{B_i})b_i{}^{i \in I \setminus j}, \ell_j = \varsigma(y^C)b]{:}A : \tau$$

then, by Lemma *3.11*, there exist $D \equiv [\ell_i{:}D_i{}^{(i \in I)}]$ such that $\widehat{\Delta} \vdash_{\mathrm{T}} D <: A$, $D \equiv C$, and for all $i \in I$, $B_i \equiv D$. Then $\tau \equiv \langle \ell_k{:}\sigma \rightarrow \psi \rangle$ for some $\sigma, \psi, k \in I$, and, for all $i \in I \setminus j$, $\Delta, \Gamma, x_i{:}D \vdash_{\mathrm{O}} b_i : D_i$, and $\Delta, \Gamma, y{:}D \vdash_{\mathrm{O}} b : D_i$. By Lemma *1.13*, the assumption

$$\widehat{\Delta; \Gamma} \vdash_{\mathrm{O}} [\ell_i = \varsigma(x_i^D)b_i{}^{(i \in I)}].\ell_j \Leftarrow \varsigma(y^D)b : A$$

gives $D <: A$, and $\widehat{\Delta; \Gamma} \vdash_{\mathrm{O}} [\ell_i = \varsigma(x_i^D)b_i{}^{(i \in I)}] : D$, and for all $i \in I$, $\widehat{\Delta; \Gamma}, x_i{:}D \vdash_{\mathrm{O}} b_i : D_i$, so, in particular, $\widehat{\Delta; \Gamma}, x_j{:}D \vdash_{\mathrm{O}} b_j : D_j$.

If $j = k$ and $h \neq j$, we can now construct:

$$\frac{\Delta; \Gamma, x_h{:}D{:}\omega \vdash_{\mathrm{P}} b_h{:}D_h : \omega \quad \widehat{\Delta; \Gamma}, x_i{:}D \vdash_{\mathrm{O}} b_i : D_i \quad (\forall i \in I \setminus \{h\})}{\Delta; \Gamma \vdash_{\mathrm{P}} [\ell_i = \varsigma(x_i^{D_i})b_i{}^{(i \in I)}]{:}D : \langle \ell_h{:}\omega \rightarrow \omega \rangle} \text{(Val Object)}$$

from which, using rule (Val Update$_1$):

$$\frac{\Delta; \Gamma \vdash_{\mathrm{P}} [\ell_i = \varsigma(x_i^{D_i})b_i{}^{(i \in I)}]{:}D : \langle \ell_h{:}\omega \rightarrow \omega \rangle \quad \Delta; \Gamma, x_j{:}D{:}\tau \vdash_{\mathrm{P}} b_j{:}D_j : \psi}{\Delta; \Gamma \vdash_{\mathrm{P}} ([\ell_i = \varsigma(x_i^{D_i})b_i{}^{(i \in I)}].\ell_j \Leftarrow \varsigma(y^D)b){:}D : \langle \ell_k{:}\sigma \rightarrow \psi \rangle} \text{(Val Update}_1\text{)}$$

If instead $j \neq k$, we use rule (Val Update$_2$):

$$\dfrac{\dfrac{\Delta;\Gamma, x_k{:}D{:}\sigma \vdash_{\mathrm{P}} b_k{:}D_j : \psi \quad \widehat{\Delta;\Gamma}, x_i{:}D \vdash_{\mathrm{O}} b_i : D_i \quad (\forall i \in I\backslash k)}{\Delta;\Gamma \vdash_{\mathrm{P}} [\ell_i = \varsigma(x_i^{D_i})b_i\ ^{(i\,\in\, I)}]{:}D : \langle \ell_k{:}\sigma{\to}\psi\rangle} \quad \widehat{\Delta;\Gamma}, y{:}D \vdash_{\mathrm{O}} b : D_j}{\Delta;\Gamma \vdash_{\mathrm{P}} ([\ell_i = \varsigma(x_i^{D_i})b_i\ ^{(i\,\in\, I)}].\ell_j \Leftarrow \varsigma(y^D)b){:}D : \langle \ell_k{:}\sigma{\to}\psi\rangle}$$

and the result follows by applying rule $(<:)$.

For $\tau = \omega$, the result follows from Lemma *3.7*, Theorem *1.17* and rule (ω). For $\tau = \bigwedge_{i\in I}\tau_i$ $(n \geq 0)$, the proof follows by easy induction. ■

5 The Logical Equivalence

In [1] an equational theory of the object calculus is presented, whose first order sub-theory is generated by the rules of Figure *6* (omitting the term folding-unfolding rules, which do not make sense for the Object Calculus we consider here).

Definition 5.1 THE EQUATIONAL THEORY OF OBJECTS [1]. The *Equational Theory of Objects* is a theory of equations of the shape $a \leftrightarrow b : A$, where a and b are pre-terms and A is a pre-type; these are derived as the right-hand side of sequents $E \vdash_{\mathrm{T}} a \leftrightarrow b : A$, where E is a (pre)-environment of the Object Calculus. The rules are given in Figure *6*, plus α-congruence:

$$(\mathsf{Eq}\ \alpha) : \quad \dfrac{E \vdash_{\mathrm{O}} a : A \quad a \equiv_\alpha b}{E \vdash_{\mathrm{T}} a \leftrightarrow b : A}$$

where $a \equiv_\alpha b$ if and only if b is obtained from a by renaming bound variables and avoiding capture of any free variable in a.

Proposition 5.2 If $E \vdash_{\mathrm{T}} a \leftrightarrow b : A$ then both $E \vdash_{\mathrm{O}} a : A$, $E \vdash_{\mathrm{O}} b : A$ and $E \vdash_{\mathrm{T}} A$. Hence E is an environment, a and b are terms and A is a type under the assumptions in E.

Proof: By induction over derivations. In rule (Eq Refl), which is the base case, the premise immediately implies the thesis. Cases of (Eq Symm) and (Eq Trans) are immediate consequences of the induction hypothesis.

By rule $(\mathsf{Eq}\ \alpha)$, we can assume that bound variables do not appear in the environment E, so that the choice of their names is arbitrary and does not conflict with the choice of free variable names.

In case of Eval rules use Theorem *1.17*.

(Eq Subsumption) and (Eq Top) follow by the induction hypothesis and (Val Subsumption). For rule (Eq Sub Object) use Theorem *2.16* and the compatibility of $\vdash_{\mathrm{P}}$ with respect to $\vdash_{\mathrm{O}}$, stated in Lemma *3.7*.

All the Cong rules follow by induction.

Eventually, if $E \vdash_{\mathrm{O}} a : A$ then $E \vdash_{\mathrm{T}} A$ by Lemma *1.14(i)*; this in turn implies $E \vdash_{\mathrm{T}} \diamond$ by *(i)* of Lemma *1.13*. ■

Remark 5.3 This notion of equality includes (typed) convertibility, as is clear from rules (Eval Beta), (Eq Select) and (Eq Update), but it does not coincide with it: in fact '$\leftrightarrow$' is a congruence whereas '$\longrightarrow$' is not closed under arbitrary contexts; more importantly, this is a consequence of subtyping and precisely of rule (Eq Sub Object) (see Example *5.5*).

The assignment system of Definition *3.2* 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 σ such that $A{:}\sigma$. More precisely, and taking into account environments and their extensions into bases, we can formalise this idea as follows:

Definition 5.4 LOGICAL EQUIVALENCE. Let a be any pre-term; we define $\mathcal{P}(E, a, A)$ as the set of predicates of type A that can be assigned to a when $E \vdash_{\mathrm{O}} a : A$:

$$\mathcal{P}(E, a, A) = \{\sigma \mid \exists \Delta; \Gamma \, [\, \widehat{\Delta;\Gamma} \equiv E \ \&\ \Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma \,] \,\}.$$

We then say that the pre-terms a and b are *logically equivalent* at A and environment E if they can be assigned the same set of predicates of type A with respect to E:

$$a \simeq_E b : A \iff \mathcal{P}(E, a, A) = \mathcal{P}(E, b, A).$$

Example 5.5 As in Example *4.4*, let $A \equiv [\ell_0{:}\mathit{Int}, \ell_1{:}\mathit{Int}]$, and $a \equiv [\ell_0 = \varsigma(x_0^A)1, \ell_1 = \varsigma(x_1^A)x.\ell_0]$. Further consider:

$$b \equiv [\ell_0 = \varsigma(x_1^A)1, \ell_1 = \varsigma(x_1^A)1].$$

In [1], Section 7.6.2 it is argued that they 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 $\Delta; \vdash_{\mathrm{P}} 1{:}\mathit{Int} : 1$, then $\Delta; \vdash_{\mathrm{P}} b{:}A : \langle \ell_1{:}\omega \to 1 \rangle$ but $\Delta; \nvdash a{:}A{:}\langle \ell_1{:}\omega \to 1 \rangle$. Indeed (omitting again the Δ's and all those parts of the derivation justifying the assignment of the predicate to a type):

$$\dfrac{\begin{array}{c}\diagup\!\!\!\!\!\!\diagdown \\ x_1{:}A{:}\omega \vdash_{\mathrm{P}} 1{:}\mathit{Int} : 1\end{array}}{\vdash_{\mathrm{P}} b{:}A : \langle \ell_1{:}\omega \to 1 \rangle}\ (\text{Val Object})$$

Replacing b by a would not yield a valid derivation. The best we can do in the case of a is instead:

$$\dfrac{\dfrac{x_1{:}A{:}\langle \ell_0{:}\omega{\to}1\rangle \vdash_{\mathrm{P}} x_1{:}A : \langle \ell_0{:}\omega{\to}1\rangle \qquad x_1{:}A{:}\langle \ell_0{:}\omega{\to}1\rangle \vdash_{\mathrm{P}} x_1{:}A : \omega}{x_1{:}A{:}\langle \ell_0{:}\omega{\to}1\rangle \vdash_{\mathrm{P}} x_1.\ell_0{:}\mathit{Int} : 1}\ (\text{Val Select})}{\vdash_{\mathrm{P}} a{:}A : \langle \ell_1{:}\langle \ell_0{:}\omega{\to}1\rangle{\to}1\rangle}\ (\text{Val Object})$$

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

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

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

can be obtained by rule (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 $a \simeq_{\emptyset} b : [\ell_0{:}\mathit{Int}]$ 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 with respect to $\leq$).

We relate here logical equivalence to the equational theory of the Object Calculus.

Lemma 5.6 Logic equivalence is a congruence; more precisely:

$$a \simeq_{E,E'} a' : A \ \&\ b \simeq_{E,x:A,E'} b' : B \ \Rightarrow\ b\{x \leftarrow a\} \simeq_{E,E'} b'\{x \leftarrow a'\} : B.$$

Proof: This is just a rephrasing of the substitution property stated in Lemma *4.2*. ∎

We want to establish that equality in the Object Calculus implies logical equivalence, proving that what we have seen in the Example *5.5* actually holds in general. Corollary *2.17* is a first evidence of the consistency of the predicate assignment system with respect to the subtyping relation. It is however not enough.

Lemma 5.7 Let $A \equiv [\ell_i{:}B_i\ ^{i\in I}]$, *and* $A' \equiv [\ell_k{:}B_k\ ^{k\in I\cup J}]$, *where* $I \cap J = \emptyset$. *If*

$$E \vdash_{\mathrm{T}} [\ell_i = \varsigma(x_i^A)b_i\ ^{i\in I}] \leftrightarrow [\ell_k = \varsigma(x_k^{A'})b_k\ ^{k\in I\cup J}] : A$$

is the conclusion of rule (Eq Sub Object) *under the premises*

$$E, x_i{:}A \vdash_{\mathrm{O}} b_i : B_i \quad (\forall i \in I) \quad \textit{and} \quad E, x_j{:}A' \vdash_{\mathrm{O}} b_j : B_j \quad (\forall j \in J),$$

then $[\ell_i = \varsigma(x_i^A)b_i\ ^{i\in I}] \simeq_E [\ell_k = \varsigma(x_k^{A'})b_k\ ^{k\in I\cup J}] : A$.

Proof: Since $I \subseteq I \cup J$, by rule (Sub Object) we have that $E \vdash_{\mathrm{T}} A'{<:}A$. Moreover if $i \in I$, and $\Delta \vdash_{\mathrm{O}} A : \langle \ell_i{:}\tau{\to}\psi\rangle$ then we claim that:

$$\Gamma; \Delta, x_i{:}A{:}\tau \vdash_{\mathrm{P}} b_i{:}B_i : \psi \iff \Gamma; \Delta, x_i{:}A'{:}\tau \vdash_{\mathrm{P}} b_i{:}B_i : \psi. \tag{1}$$

Indeed the $\Rightarrow$ implication an instance of Lemma *3.10*. To prove $\Leftarrow$ we remark that $\Delta \vdash_{\mathrm{O}} \langle \ell_i{:}\tau{\to}\psi\rangle : A$ implies $\Delta \vdash_{\mathrm{O}} A : \tau$ by *(e)* of Lemma *2.10*, so that the hypothesis $E, x_i{:}A \vdash_{\mathrm{O}} b_i : B_i$ implies that the assumption $x_i{:}A'{:}\tau$ can be weakened to $x_i{:}A{:}\tau$, and we get $\Gamma; \Delta, x_i{:}A{:}\tau \vdash_{\mathrm{P}} b_i{:}B_i : \psi$ as desired.

Let $a \equiv [\ell_i = \varsigma(x_i^A)b_i\ ^{i\in I}]$ and $a' \equiv [\ell_k = \varsigma(x_k^{A'})b_k\ ^{k\in I\cup J}]$. Suppose that $\sigma \in \mathcal{P}(E, a, A)$ and assume without loss of generality that $\sigma \in \mathcal{P}_{\mathrm{S}} \setminus \{\omega\}$ (otherwise either it is ω, the trivial case, or it is a conjunction of predicates in $\mathcal{P}_{\mathrm{S}}$ and we reason similarly for each conjunct): hence $\sigma \equiv \langle \ell_i{:}\tau{\to}\psi\rangle$ for some $i \in I$ and τ, ψ such that $\Gamma; \Delta, x_i{:}A{:}\tau \vdash_{\mathrm{P}} b_i{:}B_i : \psi$, where $\widehat{\Gamma; \Delta} \equiv E$. By this and Lemma *3.8* we know that $\Delta \vdash_{\mathrm{O}} A : \langle \ell_i{:}\tau{\to}\psi\rangle$, therefore by (*1*) we have $\Gamma; \Delta, x_i{:}A'{:}\tau \vdash_{\mathrm{P}} b_i{:}B_i : \psi$, which easily implies $\Gamma; \Delta \vdash_{\mathrm{P}} a'{:}A : \sigma$, i.e. $\sigma \in \mathcal{P}(E, a', A)$.

Vice-versa, let $\sigma \in \mathcal{P}(E, a', A)$. From $\Gamma; \Delta \vdash_{\mathrm{P}} a'{:}A : \sigma$ it follows that $\Delta \vdash_{\mathrm{O}} A : \sigma$ that is, if $\sigma \equiv \langle \ell_i{:}\tau{\to}\psi\rangle$ as before, we know that $i \in I$ and $\Delta \vdash_{\mathrm{O}} A : \tau$. By *(c)* of Lemma *3.11* it must be the case that $\Gamma; \Delta, x_i{:}A'{:}\tau \vdash_{\mathrm{P}} b_i{:}B_i : \psi$, whence we get $\Gamma; \Delta \vdash_{\mathrm{P}} a{:}A : \sigma$ by (*1*) and rule (Val Object).

■

Theorem 5.8 *If* $E \vdash_{\mathrm{T}} a \leftrightarrow b : A$ *then* $a \simeq_E b : A$.

Proof: The proof is by induction over the derivation of $E \vdash_{\mathrm{T}} a \leftrightarrow b : A$. In case the derivation ends by any rule among (Eval Beta), (Eval Select) and (Eval Update) the thesis follows by Theorems *4.3* and *4.6*. Those cases in which the derivation ends by rules that establish the congruence properties of $\leftrightarrow$, namely (Abs Cong), (App Cong), (Object Cong), (Sel Cong) and (Update Cong), the result follows from Lemma *5.6*. If the last rule is (Eq Top) then $A \equiv$ Top and we observe that, by Lemma *2.10*, $\mathcal{P}(E, a, \mathsf{Top})$ and $\mathcal{P}(E, b, \mathsf{Top})$ contain exactly all predicates equivalent to ω, so they coincide.

If the derivation ends by rule (Eq Sub Object), the thesis follows by Lemma *5.7*. The case remains in which the derivation ends by rule (Eq Subsumption):

$$\frac{E \vdash_{\mathrm{T}} a \leftrightarrow b : A \quad E \vdash_{\mathrm{T}} A <: B}{E \vdash_{\mathrm{T}} a \leftrightarrow b : B}$$

Let $\tau \in \mathcal{P}(E, a, B)$: then $\Delta; \Gamma \vdash_{\mathrm{P}} a{:}B : \tau$ for some $\Delta; \Gamma$ such that $\widehat{\Delta; \Gamma} \equiv E$. Since the premise $E \vdash_{\mathrm{T}} a \leftrightarrow b : A$ implies that both $E \vdash_{\mathrm{O}} a : A$ and $E \vdash_{\mathrm{O}} b : A$ are derivable and $E \vdash_{\mathrm{T}} A <: B$ implies that $\widehat{\Delta} \vdash_{\mathrm{T}} A <: B$, Lemma *3.9* applies, so that $\Delta; \Gamma \vdash_{\mathrm{P}} a{:}A : \tau$ is derivable, so $\tau \in \mathcal{P}(E, a, A)$. Also, by induction, $\mathcal{P}(E, a, A) = \mathcal{P}(E, b, A)$, that is $\Delta'; \Gamma' \vdash_{\mathrm{P}} b{:}A : \tau$ is derivable for certain $\Delta'; \Gamma'$ such that $\widehat{\Delta'; \Gamma'} \equiv E$. Then $\Delta'; \Gamma' \vdash_{\mathrm{P}} b{:}B : \tau$ follows by rule $(<:)$, that is $\tau \in \mathcal{P}(E, b, B)$ as desired.

■

Remark 5.9 The converse of Theorem *5.8* does not hold. To see a counter example consider:

$$d \equiv [\ell_0 = \varsigma(x_0^A)1, \ell_1 = \varsigma(x_1^A)\lambda y^{\mathit{Int}}.y],$$
$$e \equiv [\ell_0 = \varsigma(x_0^A)1, \ell_1 = \varsigma(x_1^A)\lambda y^{\mathit{Int}}.(x_1.\ell_0 \Leftarrow \varsigma(z^A)y).\ell_0]$$

where $A \equiv [\ell_0{:}\mathit{Int}, \ell_1{:}\mathit{Int}{\to}\mathit{Int}]$. It is easy to see that both $\vdash_{\mathrm{O}} d : A$ and $\vdash_{\mathrm{O}} e : A$, but clearly $\not\vdash d \leftrightarrow e : A$. Their behaviour is however the same: this is clear for the field ℓ_0; concerning ℓ_1 we observe that both $d.\ell_1$ and $e.\ell_1$ are the identity over *Int*. Indeed the side-effect which occurs when applying $e.\ell_1$ to any integer cannot be observed, since the *self* of e gets lost in the computation, being the $\mathsf{FOb}_{1<:\mu}$-calculus functional.

We argue that $d \simeq e : A$ (a detailed proof can be obtained by means of Lemma *3.11*). In fact it is not difficult to see that $\mathcal{P}(\emptyset, d, A) \subseteq \mathcal{P}(\emptyset, e, A)$. For the opposite inclusion, if $\vdash_{\mathrm{P}} e{:}A : \langle \ell_1{:}\sigma{\to}\phi \rangle$ then it must be obtained by deriving

$$x_1{:}A'{:}\sigma \vdash_{\mathrm{P}} \lambda y^{\mathit{Int}}.(x_1.\ell_0 \Leftarrow \varsigma(z^A)y).\ell_0{:}\mathit{Int}{\to}\mathit{Int} : \phi,$$

for some subtype A' of A: since $\sigma \in \mathcal{L}_A$ and $\vdash_{\mathrm{O}} e : A$ we can freely assume that A' is A, and that the last rule of this derivation is (Val Object). From the fact that $\phi \in \mathcal{L}_{\mathit{Int}\to\mathit{Int}}$ we also know that, if it is not trivial, then it is an arrow, namely $\phi \equiv \tau{\to}\kappa$ for certain $\tau, \kappa \in \mathcal{L}_{\mathit{Int}}$. Going backward in the derivation we arrive at $x_1{:}A{:}\sigma, y{:}\mathit{Int}{:}\tau \vdash_{\mathrm{P}} (x_1.\ell_0 \Leftarrow \varsigma(z^A)y).\ell_0{:}\mathit{Int} : \kappa$ which is the conclusion of (Val Select). By this we know that in the derivation we must have both

$$x_1{:}A{:}\sigma, y{:}\mathit{Int}{:}\tau \vdash_{\mathrm{P}} x_1.\ell_0 \Leftarrow \varsigma(z^A)y{:}A : \langle \ell_0{:}\sigma'{\to}\kappa \rangle \tag{2}$$

and $x_1{:}A{:}\sigma, y{:}\mathit{Int}{:}\tau \vdash_{\mathrm{P}} x_1.\ell_0 \Leftarrow \varsigma(z^A)y{:}A : \sigma'$ for some σ'. As a matter of fact we do not need the information encoded by σ', which can be simply ω, so that we are left to derive

(*2*) by means of rule (Val Update$_1$). This amounts to assume that σ is non trivial (e.g. take $\langle \ell_0{:}\omega \rangle$), and to show $x_1{:}A{:}\sigma, y{:}\mathit{Int}{:}\tau, z{:}A{:}\omega \vdash_{\mathrm{P}} y{:}\mathit{Int} : \kappa$, for which $\tau \leq \kappa$ is necessary and sufficient. But this very last fact also suffices to derive

$$x_1{:}A{:}\sigma, y{:}\mathit{Int}{:}\tau \vdash_{\mathrm{P}} \lambda y^{\mathit{Int}}.y{:}\mathit{Int}{\to}\mathit{Int} : \tau{\to}\kappa,$$

from which it is immediate to obtain $\vdash_{\mathrm{P}} d{:}A : \langle \ell_1{:}\sigma{\to}\phi \rangle$ by rule (Val Object) as desired.

We end this remark by observing that the logical complexity of the theory of logical equivalence is Π^0_2, and we conjecture that it is a Π^0_2-complete one, as it is the case for the theories of filter models of the untyped λ-calculus from which it derives. If this is the case there exists no formal system extending the theory of $\leftrightarrow$ such that it coincides with $\simeq$.

6 Logical Equivalence and Observational Semantics

Observational semantics for the first order calculus has been defined in [18] in Morris-style, called there "contextual equivalence". It consists of *inseparability* by means of contexts of ground type. 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 here.

We claim that, when restricted to closed terms, logical equivalence is included in observational equivalence. To this aim we will establish a computational adequacy result for the logical semantics with respect to convergence, which states that any well-typed term can be assigned a non-trivial predicate if and only if it converges to a value. This is achieved by means of the realizability interpretation of predicates given in Definition *3.3*, proving that the characterisation results of [16] are preserved in the typed context of the first order object calculus.

As also mentioned above, we will write $\_{:}A \vdash_{\mathrm{O}} C[\_] : B$ to express that the closed context $C[\_]$ is typed with B, under the assumption that the "hole _" has type A; $C[a]$ is the result of replacing '_' by a in $C[\_]$. We write a^A stays for a closed term a of a closed type A, i.e. such that $\emptyset \vdash_{\mathrm{O}} a : A$ (abbreviated by $\vdash_{\mathrm{O}} a : A$).

Definition 6.1 OBSERVATIONAL EQUIVALENCE. Two closed terms a and b are called *observationally equivalent at type* A, written $a \simeq^{\mathcal{O}}_A b$, if both a^A and b^A, and

$$\forall C[\_].\_ : A \vdash C[\_] : K \;\;\Rightarrow\;\; (C[a] \Downarrow v \iff C[b] \Downarrow v).$$

for any ground type K and value v of type K.

Remark 6.2 *i*) Typeable values can always be assigned non-trivial predicates.

ii) Definition *6.1* differs from the definition of contextual equivalence in [18] 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 the above definition and in Theorem *6.7* the predicate $a \Downarrow v$ instead of $a \Downarrow$.

Let $a\{x_j \leftarrowtail b_j\}_{j\leq k}$ abbreviate the simultaneous substitution of b_j for x_j in a for all $1 \leq j \leq k$, and similarly $A\{X_i \leftarrowtail B_i\}_{i\leq h}$, substituting each B_i for X_i in A for all $1 \leq i \leq k$.

Definition 6.3 COMPATIBLE SUBSTITUTIONS. Assume $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$, where $\Delta \equiv J_1, \ldots, J_h$ and each judgement J_i either is $X_i{:}\sigma_i$ or $X_i{:}\sigma_i <: D_i$, and $\Gamma \equiv x_1{:}C_1{:}\tau_1, \ldots, x_k{:}C_k{:}\tau_k$. We say that the simultaneous substitutions $\{X_i \leftarrowtail B_i\}_{i\leq h}$ and $\{x_j \leftarrowtail b_j\}_{j\leq k}$ are *compatible* with $\Delta;\Gamma$ if: $\widehat{\Delta} \vdash_{\mathrm{T}} B_i <: D_i$ (in case $X_i{:}\sigma_i <: D_i \in \Delta$), and $\vdash_{\mathrm{O}} B_i : \sigma_i$ for all $1 \leq i \leq h$, and $\vdash_{\mathrm{P}} b_j{:}C_j\{X_i \leftarrowtail B_i\}_{i\leq h} : \tau_j$ for all $1 \leq j \leq k$.

In the above definition, if $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$ is derivable then $\mathit{fv}(A) \cup \mathit{fv}(a) \subseteq \mathit{dom}(\Delta) \cup \mathit{dom}(\Gamma)$: hence compatible substitutions are closing a and A if they replace all free variable occurrences by closed types and terms. We will call such a substitution a *closing* substitution.

Lemma 6.4 *Assume that the closing substitutions* $\{X_i \leftarrowtail B_i\}_{i\leq h}$ *and* $\{x_j \leftarrowtail b_j\}_{j\leq k}$ *are compatible with* $\Delta;\Gamma$*, then:*

$$\textit{if } \Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma \textit{ then } \vdash_{\mathrm{P}} a\{x_j \leftarrowtail b_j\}_{j\leq k}{:}A\{X_i \leftarrowtail B_i\}_{i\leq h} : \sigma$$

Proof: By Lemma *2.14* and Lemma *4.2*. ■

Theorem 6.5 REALIZABILITY THEOREM. *Let* $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$*; assume that the closing substitutions* $\{X_i \leftarrowtail B_i\}_{i\leq h}$ *and* $\{x_j \leftarrowtail b_j\}_{j\leq k}$ *are compatible with* $\Delta;\Gamma$*. If* ${b_j}^{C'_j} \in [\![\tau_j]\!]$ *whenever* $x_j{:}C_j{:}\tau_j \in \Gamma$*, then* ${a'}^{A'} \in [\![\sigma]\!]$*, where* $a' \equiv a\{x_j \leftarrowtail b_j\}_{j\leq k}$*,* $C'_j \equiv C_j\{X_i \leftarrowtail B_i\}_{i\leq h}$ *and* $A' \equiv A\{X_i \leftarrowtail B_i\}_{i\leq h}$*.*

Proof: We write $a\vartheta$ for $a\{x_j \leftarrowtail b_j\}_{j\leq k}$ and $A' \equiv A\Theta$ for $A\{X_i \leftarrowtail B_i\}_{i\leq h}$. The proof proceeds by induction on the derivation for $\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma$. The case $(\mathsf{Val}\ x)$ is trivial; the case $(<:)$ follows by the induction hypothesis and Lemma *3.6*. For the remaining cases:

(Val Fun) : the last inference is an instance of the rule:

$$\frac{\Delta;\Gamma,x{:}A{:}\sigma \vdash_{\mathrm{P}} a{:}B : \phi}{\Delta;\Gamma \vdash_{\mathrm{P}} \lambda x^A.a{:}A{\to}B : \sigma{\to}\phi}$$

$(\lambda x^A.a)\vartheta$ is a value, hence it converges trivially. Since ϑ, Θ are compatible with $\Delta;\Gamma$, given any $b^{A'} \in [\![\sigma]\!]$, the substitutions $\vartheta[x := b]$ (which is the same as ϑ but for its value on x which is b) and Θ are compatible with $\Delta;\Gamma,x{:}A{:}\sigma$. Then, by induction $a\vartheta[x := b] \in [\![\phi]\!]$, and we conclude from $((\lambda x^A.a)\vartheta)b \longrightarrow a\vartheta[x := b]$, Lemma *3.5* and the arbitrary choice of $b^{A'}$.

(Val Appl) : the last inference is an instance of the rule:

$$\frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A{\to}B : \sigma{\to}\phi \quad \Delta;\Gamma \vdash_{\mathrm{P}} b{:}A : \sigma}{\Delta;\Gamma \vdash_{\mathrm{P}} ab{:}B : \phi}$$

By induction $a\vartheta \in [\![\sigma{\to}\phi]\!]$ so that $a\vartheta \Downarrow (\lambda x^A.d)$ for some closed abstraction $\lambda x^A.d$; moreover $b\vartheta \in [\![\sigma]\!]$ by induction, hence $d[x := b\vartheta] \in [\![\phi]\!]$, and we conclude by Lemma *3.5* and

$$(ab)\vartheta \equiv (a\vartheta)(b\vartheta) \stackrel{*}{\longrightarrow} (\lambda x^A.d)(b\vartheta) \longrightarrow d[x := b\vartheta].$$

(Val Object) : the last inference is an instance of the rule:

$$\frac{\Delta;\Gamma,x_j{:}A{:}\sigma \vdash_{\mathrm{P}} b_j{:}B_j : \phi \quad \widehat{\Delta;\Gamma},x_i{:}A \vdash_{\mathrm{O}} b_i : B_i \quad (\forall i \in I \setminus \{j\})}{\Delta;\Gamma \vdash_{\mathrm{P}} [\ell_i = \varsigma(x_i^A)b_i{}^{(i\,\in\,I)}]{:}A : \langle \ell_j{:}\sigma{\to}\phi\rangle}\ (j \in I)$$

Now $a \equiv [\ell_i = \varsigma(x_i^A)b_i{}^{(i\,\in\,I)}]$ is an object term, whose closure is a value: hence $a\vartheta \Downarrow$. That $\ell_j \in \mathit{Label}(A)$ is a side condition of the rule. For any $d^{A'} \in [\![\sigma]\!]$ we have that $\vartheta[x_j := d]$, and Θ are compatible with $\Delta;\Gamma,x_j{:}A{:}\sigma$ and

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

by induction.

(Val Select) : the last inference is an instance of the rule:

$$\frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \langle \ell_j{:}\sigma{\to}\phi\rangle \quad \Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \sigma}{\Delta;\Gamma \vdash_{\mathrm{P}} a.\ell_j{:}B_j : \phi}$$

By induction $a\vartheta \Downarrow v$, for some value v, $\ell_j \in \mathit{Label}(A)$ and $a\vartheta.\ell_j(d) \in [\![\phi]\!]$ for any $d^{A'} \in [\![\sigma]\!]$; since $a\vartheta \in [\![\sigma]\!]$ (by induction again) we have that $v \in [\![\sigma]\!]$ by Lemma *3.5*

so that:

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

and we conclude again by Lemma *3.5*.

$(\mathsf{Val\ Update}_1)$: the last inference has the shape:

$$\frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \tau \quad \Delta;\Gamma, y{:}A{:}\sigma \vdash_{\mathrm{P}} b{:}B_j : \phi}{\Delta;\Gamma \vdash_{\mathrm{P}} (a.\ell_j \Leftarrow \varsigma(y^A)b){:}A : \langle \ell_j{:}\sigma{\to}\phi\rangle}\ (\ell_j \in \mathit{Label}(A), \tau \neq \omega)$$

By induction $a\vartheta \in [\![\tau]\!]$; since $\tau \neq \omega$ this 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 $d^{A'} \in [\![\sigma]\!]$, $\vartheta[y := d], \Theta$ is compatible with $\Delta;\Gamma, y{:}A{:}\sigma$, so that we conclude by induction

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

$(\mathsf{Val\ Update}_2)$: the last inference has the form:

$$\frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \langle \ell_j{:}\phi\rangle \quad \widehat{\Delta;\Gamma}, y{:}A \vdash_{\mathrm{T}} b{:}B_i}{\Delta;\Gamma \vdash_{\mathrm{P}} (a.\ell_i \Leftarrow \varsigma(y^A)b){:}A : \langle \ell_j{:}\phi\rangle}\ (i \neq j)$$

By induction we know that $a\vartheta \in [\![\langle \ell_j{:}\phi\rangle]\!]$, which implies that $a\vartheta \Downarrow$: hence $(a.\ell_j \Leftarrow \varsigma(y^A)b)\vartheta \Downarrow$. It remains to show that $(a.\ell_j \Leftarrow \varsigma(y^A)b)\vartheta \in [\![\langle \ell_j{:}\phi\rangle]\!]$ according to the shapes of ϕ. The case in which $\phi \equiv \omega$ is already proved by the fact that $(a.\ell_j \Leftarrow \varsigma(y^A)b)\vartheta \Downarrow$. The case $\phi \equiv \langle \ell{:}\psi\rangle$ is impossible, since $(a.\ell_j \Leftarrow \varsigma(y^A)b)\vartheta$ is well typed because of the compatibility of ϑ, and no well-typed term might be assigned a predicate of that shape. We are left with the case $\phi \equiv \tau{\to}\psi$: then, since $i \neq j$, we have that $(a.\ell_j \Leftarrow \varsigma(y^A)b)\vartheta.\ell_i(d) \equiv a\vartheta.\ell_i(d)$ is in $[\![\psi]\!]$ when $d^{A'} \in [\![\tau]\!]$ by the inductive hypothesis. ■

Corollary 6.6 CHARACTERISATION OF CONVERGENCE. *Let a^A be any closed term: then $a \Downarrow$ if and only if $\vdash_{\mathrm{P}} a{:}A : \sigma$ for some non-trivial σ.*

Proof: As noted in *(i)* of Remark *6.2* typeable value 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 *4.6*; on the other hand a straightforward induction on the structure of σ shows that if σ is non trivial, then any $a^A \in [\![\sigma]\!]$ converges: by this and Theorem *6.5* we conclude. ■

Corollary *6.6* has important consequences. First it expresses the computational adequacy of the logical equivalence, in the sense that no divergent term can be equated with a term converging to a value. Combining this with the subject reduction and expansion Theorems

4.3 and *4.6*, it says that predicates actually foresee properties of values: since the latter include function and objects, such properties concern behaviour and not just aspects of elementary values such as integers or Booleans.

But it is stronger than the above mentioned theorems, since we know that logical equivalence is not simply convertibility (not even the equational theory of [1]). In fact Corollary *6.6* entails the subsequent Theorem *6.7*, which states the inclusion of the logical equivalence into the observational equivalence, and hence the consistency of the former, and of the whole logic of predicates we are about.

Theorem 6.7 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 these predicates are distinct for different values and that $\vdash_{\mathrm{P}} v{:}K : \kappa_v$ is assumed for each v. Then for any a^A and b^A, if $a \simeq b : A$ then $a \simeq^{\mathcal{O}}_A b$.*

Proof: Let $\_ : A \vdash C[\_] : K$ be any context of ground type K such that $C[a] \Downarrow v$ for some value v. By the hypothesis that $\vdash_{\mathrm{P}} v{:}K : \kappa_v$ and Theorem *4.6*, we have $\kappa_v \in \mathcal{P}(\emptyset, C[a], K)$. On the other hand $a \simeq b : A$ implies $C[a] \simeq C[b] : K$, since logical equivalence is a congruence by Lemma *5.6*, and therefore $\kappa_v \in \mathcal{P}(\emptyset, C[b], K)$, that is $\vdash_{\mathrm{P}} C[b]{:}K : \kappa_v$. By Corollary *6.6* it follows that $C[b] \Downarrow v'$ for some v', whence $\vdash_{\mathrm{P}} v'{:}K : \kappa_v$ by Theorem *4.3*. From the hypothesis that K is a ground type and that κ_v is a characteristic predicate of v, we conclude $v' \equiv v$. ■

By using bisimulation and its coincidence with observational equivalence, in [18] it is shown that, taking a and b as in example *5.5*, $a \simeq^{\mathcal{O}}_{[\ell_1:Int]} b$. 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 $a.\ell_1$ depends on such a value while $b.\ell_1$ does not, becomes apparent; but the overriding of ℓ_0 is inhibited in contexts with the hole of type $[\ell_1{:}Int]$, where ℓ_0 is hidden.

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

Language inclusion alone is not sufficient to account for subtyping of object types, while it is for record types (see [17]): this is the essential reason for the presence of rule (Val Select) in our system. It is reasonable to think that the failure of equivalences like $a \simeq b : [\ell_1{:}Int]$ from Example *5.5* depends on the fact that no rule accounts for the hiding effect of subtyping in the case of object types. One possibility for coping with such a

limitation is the following rule:

$$I \cap J = \emptyset,\ A \equiv [\ell_i{:}B_i{}^{\,i\in I\cup J}],\ A' \equiv [\ell_i{:}B_i{}^{\,i\in J}] :$$
$$\frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \langle \ell{:}\langle \ell_i{:}\sigma_i{}^{\,i\in I}\rangle \wedge \tau \rightarrow \rho\rangle \quad \Delta;\Gamma \vdash_{\mathrm{P}} a{:}A : \langle \ell_i{:}\sigma_i{}^{\,i\in I}\rangle \quad \Delta \vdash_{\mathrm{O}} A' : \langle \ell{:}\tau\rightarrow\rho\rangle}{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A' : \langle \ell{:}\tau\rightarrow\rho\rangle}$$

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 *5.5*, by this rule one can derive $\vdash_{\mathrm{O}} b : [\ell_1{:}\mathit{Int}]{:}\langle \ell_1{:}\omega\rightarrow 1\rangle$, which makes a and b logically indiscernible at type $[\ell_1{:}\mathit{Int}]$.

We have not considered this rule in our assignment system, however. The proof of the soundness of such a rule requires a different definition of the realizability interpretation, and makes the proof theory of the assignment system more involved. On the other hand, it is difficult to say to what extent we obtain a stronger equivalence. Indeed we know in advance that logical equivalence cannot coincide with observational equivalence: the former is indeed the theory of a filter model which is a D_∞ model; we know from [5] that it is not fully abstract with respect to the lazy λ-calculus, which is a sub-calculus of the Object Calculus we consider. Were logical equivalence and observational equivalence the same, a full abstraction property would hold for the model.

7 Conclusions and Related Work

The system and results presented in this paper have been developed through a series of papers, [16], [17], [9], and [10], of which the present work is an extended and revised version.

To summarise our work, it can be seen as an intersection type assignment system (see e.g. [10] and the references there) to typed terms of the first order object calculus from [1]. A similar idea of assignment of logical formulas to terms of a simply typed λ-calculus with recursive types is the endogenous logic in [4], where languages are used to provide a finitary description of the domain interpretation of types, and a denotation of terms, much as it happens for filter models of the type free λ-calculus in [11]. With respect to these antecedents our technical contribution consists both in the consideration of object types and, more importantly, in the treatment of subtyping, for which we build on ideas presented in [17].

Our results substantiate the claim that logical semantics allows for a clean and elegant understanding of subtyping, which is seen as a form of (reverse) inclusion of logical theories, indexed by types. This seems remarkable in presence of arrow, object and recursive types, whose combination is notoriously very difficult to model.

The resulting logical equivalence is consistent with the equivalence axiomatically defined in [1], and operationally sound with respect to the Morris-style semantics studied in [18]. We remark that logic adds to the abstraction represented by types, in that it is able to capture computational properties like convergence and context separability, which is not the case for types. To show this, we resort to the technical tool of realizability interpretation of predicates, which comes from [16] and is a mild extension of known techniques from the λ-calculus. Its relation to types and subtyping, however, is not completely understood and is clearly involved in the treatment of the rule suggested at the end of Section *6*.

The present study rests on the assignment system and its proof theoretical properties, without facing the problem of models. In [17] a filter model construction is proposed in which types are interpreted as the CUPERs (studied in [6, 3, 14] and used in [1] Ch. 14) induced by the indiscernibility relation with respect to the predicates of a language. Terms are interpreted as filters of predicates, but their meaning in a type A is obtained by restricting to $\mathcal{L}_A$. Such a restriction, trivially idempotent, is also continuous, which suggested the interpretation of the restriction operation in terms of retractions over a D_∞ universal model in [9]. Unfortunately retractions do not allow for a sound treatment of subtyping because of their covariant behaviour with respect to both left and right-hand sides of arrow types. Although we think that a model is implicitly described by our system, the analysis of its structure and the comparison with existing denotational models of subtyping deserve further investigation.

Our system provides a program logic for the first order ς-calculus, which is natural to compare with similar proposals in the literature. In [2] a Hoare-style logic for a first order object calculus with subtyping is presented. A close relationship exists between their transition relations and our predicates $\langle \ell_i{:}\sigma_i{\to}\phi_i \ ^{(i \in I)}\rangle$. In fact, even if transition relations are expressed via first order predicate logic whereas our logic is propositional, they specify pre and post conditions of methods in terms of properties of filed values before and after method invocation. We can do the same, since fields are simply methods that do not depend on self variables, so that we can encode their properties by means of predicates of the shape $\langle \ell{:}\omega{\to}\psi\rangle$ (and conjunctions of them); we then put them as the premises σ_i above, and encode post conditions in the ϕ_i. We stress however that our framework is more powerful, because in [2] only field update is permitted, whereas our logic is sound in the presence of the stronger operation of method overriding.

The latter limitation is removed in [21], at the price of losing monotonicity of the transition relations. To handle this difficulty, a notion of specification *Spec*$(A, \vec{B}, \vec{T})$ is introduced as the unique fixed point of the bi-functor $\Phi_{A,\vec{B},\vec{T}}$ induced by the field predicate A, the result predicates $\vec{B}$ and the transition relations $\vec{T}$; such a fixed point does not exist in general, and it can be assured only under suitable conditions (Existence Theorem). We first remark a tight similarity between the definition of *Spec*$(A, \vec{B}, \vec{T})$ and of $[\![\langle \ell{:}\sigma \rightarrow \psi \rangle]\!]$ here, especially for the quantification in the clause $\forall c^A \in [\![\sigma]\!].\ a.\ell(c) \in [\![\psi]\!]$. Then we observe that the realizability interpretation of predicates is inductive, so that $[\![\sigma]\!]$ always exists. Although we do not have a definite answer, it seems reasonable to think that our predicates are particular cases of specifications, enjoying the good properties, which would explain why we do not need an existence theorem at all. The model of the logic proposed in [21] is for the untyped ς-calculus, both functional and imperative, so that there is nothing to remark about subtyping semantics here.

A different approach to the relationship between program logic and subtyping is "behavioural subtyping" as exposed e.g. in [22]. It is based on the "subtype requirement" which says that if $A <: B$ and $\phi(b)$ for all $b{:}B$ then $\phi(a)$ for all $a{:}A$, where $\phi(x)$ is a certain predicate of x. This recalls the fact that $\mathcal{L}_A \supseteq \mathcal{L}_B$ whenever $A <: B$ in our system. However, because of the universal quantifiers in the subtyping requirement, the predicates of [22] are likely to be properties of types rather than of programs, so that they are better seen as a reinforcement of the abstractions expressed by types, rather than as a description of behaviours. The latter is exactly what we gain in our system, as shown via the Realizability Theorem *6.5*.

References

[1] M. Abadi and L. Cardelli. *A Theory of Objects*. Springer, 1996.

[2] M. Abadi and K. Leino. A logic of object oriented programs. In *Proc. of TAPSOFT '97*, volume 1214 of *Lecture Notes in Computer Science*, pages 682–696, 1997.

[3] M. Abadi and G.D. Plotkin. A per model of polymorphism and recursive types. In *Proc. of of LICS'90*, pages 355–365, 1990.

[4] S. Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 51:1–77, 1991.

[5] S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105(2):159–267, 1993.

[6] R. Amadio. Recursion over realizability structures. *Information and Computation*, 91:55–85, 1991.

[7] S. van Bakel. Complete restrictions of the Intersection Type Discipline. *Theoretical Computer Science*, 102(1):135–163, 1992.

[8] S. van Bakel. Intersection Type Assignment Systems. *Theoretical Computer Science*, 151(2):385–435, 1995.

[9] S. van Bakel and U. de'Liguoro. Logical semantics of the first order sigma-calculus. In *Proc. of ICTCS'03*, volume 2841 of *Lecture Notes in Computer Science*, pages 202–215, 2003.

[10] S. van Bakel and U. de'Liguoro. Subtyping object and recursive types logically. In *Proc. of ICTCS'05*, volume 3701 of *Lecture Notes in Computer Science*, pages 66–80, 2005.

[11] 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.

[12] V. Breazu-Tannen, T. Coquand, C. A. Gunter, and A. Scedrov. Inheritance as implicit coercion. *Information and Computation*, 93:172–221, 1991.

[13] K. B. Bruce and G. Longo. A modest model of records, inheritance and bounded quantification. *Information and Computation*, 87:196–240, 1990.

[14] K. B. Bruce and J. C. Mitchell. Per models of subtyping, recursive types and higher-order polymorphism. In *Proc. of POPL '92*, pages 316–327, 1992.

[15] F. Cardone. Relational semantics for recursive types and bounded quantification. *Lecture Notes in Computer Science*, 372:164–178, 1989.

[16] U. de'Liguoro. Characterizing convergent terms in object calculi via intersection types. In *Proc. of TLCA'01*, volume 2004 of *Lecture Notes in Computer Science*, pages 315–328, 2001.

[17] U. de'Liguoro. Subtyping in logical form. In *Proc. of ITRS'02*, volume 70 of *ENTCS*. Elsevier, 2002.

[18] A. Gordon and G. Rees. Bisimilarity for first-order calculus of objects with subtyping. In *Proc. of POPL'96*, pages 386–395, 1996.

[19] J. L. Krivine. *Lambda-calcul, types et modèles*. Masson, 1990.

[20] J. C. Mitchell. *Foundations for Programming Languages*. MIT Press, 1996.

[21] B. Reus and T. Streicher. Semantics and logic of object calculi. *Theoretical Computer Science*, 316:191–213, 2004.

[22] B. H. Riskov and J. M. Wing. A behavioral notion of subtyping. *ACM Transactions of Programming Languages and Systems*, 16(6):1811–1841, 1994.

Figure 5: The Predicate Assignment System

Let $A \equiv [\ell_i{:}B_i\ ^{(i\,\in\,I)}]$:

$$(\textsf{Env}\ \emptyset):\quad \frac{\Delta \vdash_{\mathrm{T}} \diamond}{\Delta;\emptyset \vdash_{\mathrm{P}} \diamond}$$

$$(\textsf{Env}\ x):\quad \frac{\Delta \vdash_{\mathrm{PT}} B:\sigma \quad \Delta;\Gamma \vdash_{\mathrm{P}} \diamond}{\Delta;\Gamma, x{:}B{:}\sigma \vdash_{\mathrm{P}} \diamond}\ (x \notin \mathit{dom}(\Gamma))$$

$$(\textsf{Val}\ x):\quad \frac{\Delta;\Gamma', x{:}B{:}\sigma, \Gamma'' \vdash_{\mathrm{P}} \diamond}{\Delta;\Gamma', x{:}B{:}\sigma, \Gamma'' \vdash_{\mathrm{P}} x{:}B:\psi}\ (\sigma \leq \psi)$$

$$(<:):\quad \frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}B:\psi \quad \widehat{\Delta} \vdash_{\mathrm{T}} B <: C \quad \Delta \vdash_{\mathrm{PT}} C:\psi}{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}C:\psi}$$

$$(\textsf{Val Fun}):\quad \frac{\Delta;\Gamma, x{:}A{:}\sigma \vdash_{\mathrm{P}} a{:}B:\phi}{\Delta;\Gamma \vdash_{\mathrm{P}} \lambda x^A.a{:}A{\to}B:\sigma{\to}\phi}$$

$$(\textsf{Val Appl}):\quad \frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A{\to}B:\sigma{\to}\phi \quad \Delta;\Gamma \vdash_{\mathrm{P}} b{:}A:\sigma}{\Delta;\Gamma \vdash_{\mathrm{P}} a(b){:}B:\phi}$$

$$(\textsf{Val Object}):\quad \frac{\Delta;\Gamma, x_j{:}A{:}\sigma \vdash_{\mathrm{P}} b_j{:}B_j:\phi \quad \widehat{\Delta;\Gamma}, x_i{:}A \vdash_{\mathrm{O}} b_i:B_i \quad (\forall i \in I\backslash j)}{\Delta;\Gamma \vdash_{\mathrm{P}} [\ell_i = \varsigma(x_i^A)b_i\ ^{(i\,\in\,I)}]{:}A:\langle \ell_j{:}\sigma{\to}\phi\rangle}\ (j \in I)$$

$$(\textsf{Val Update}_1):\quad \frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A:\sigma \quad \Delta;\Gamma, y{:}A{:}\tau \vdash_{\mathrm{P}} b{:}B_j:\phi}{\Delta;\Gamma \vdash_{\mathrm{P}} (a.\ell_j \Leftarrow \varsigma(y^A)b){:}A:\langle \ell_j{:}\tau{\to}\phi\rangle}\ (\sigma \neq \omega, j \in I)$$

$$(\textsf{Val Update}_2):\quad \frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A:\langle \ell_k{:}\phi\rangle \quad \widehat{\Delta;\Gamma}, y{:}A \vdash_{\mathrm{O}} b:B_j}{\Delta;\Gamma \vdash_{\mathrm{P}} (a.\ell_j \Leftarrow \varsigma(y^A)b){:}A:\langle \ell_k{:}\phi\rangle}\ (j,k \in I, k \neq j)$$

$$(\textsf{Val Select}):\quad \frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}A:\langle \ell_j{:}\sigma{\to}\phi\rangle \quad \Delta;\Gamma \vdash_{\mathrm{P}} a{:}A:\sigma}{\Delta;\Gamma \vdash_{\mathrm{P}} a.\ell_j{:}B_j:\phi}$$

$$(\wedge I):\quad \frac{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}B:\sigma_i \quad (\forall i \in I)}{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}B:\bigwedge_{i\in I}\sigma_i}$$

$$(\omega):\quad \frac{\Delta;\Gamma \vdash_{\mathrm{P}} \diamond \quad \widehat{\Delta;\Gamma} \vdash_{\mathrm{O}} a:B}{\Delta;\Gamma \vdash_{\mathrm{P}} a{:}B:\omega}$$

Figure 6: The Equational Theory of Objects

(Eq Refl) :
$$\frac{E \vdash_{\mathrm{O}} a : A}{E \vdash_{\mathrm{T}} a \leftrightarrow a : A}$$

(Eq Symm) :
$$\frac{E \vdash_{\mathrm{T}} a \leftrightarrow b : A}{E \vdash_{\mathrm{T}} b \leftrightarrow a : A}$$

(Eq Trans) :
$$\frac{E \vdash_{\mathrm{T}} a \leftrightarrow b : A \quad E \vdash_{\mathrm{T}} b \leftrightarrow d : A}{E \vdash_{\mathrm{T}} a \leftrightarrow d : A}$$

(Eval Beta) :
$$\frac{E \vdash_{\mathrm{O}} \lambda x^A b : A \to B \quad E \vdash_{\mathrm{O}} a : A}{E \vdash_{\mathrm{T}} (\lambda x^A b)(a) \leftrightarrow b\{x \leftarrow a\} : B}$$

(Eval Select) :
$$\frac{E \vdash_{\mathrm{O}} a : A}{E \vdash_{\mathrm{T}} a.\ell_j \leftrightarrow b_j\{x_j \leftarrow a\} : B_j} \; (j \in I)$$

(Eval Update) where $I \cap J = \emptyset,\ A \equiv [\ell_i{:}B_i\ ^{i \in I}],\ A' \equiv [\ell_i{:}B_i\ ^{i \in I \cup J}],\ a \equiv [\ell_i = \varsigma(x_i^{A'})b_i\ ^{i \in I}]$:
$$\frac{E \vdash_{\mathrm{O}} a : A \qquad E, x{:}A \vdash_{\mathrm{O}} b : B_j}{E \vdash_{\mathrm{T}} a.\ell_j \Leftarrow \varsigma(x^A)b \leftrightarrow [\ell_j = \varsigma(x^{A'})b, \ell_i = \varsigma(x^{A'})b_i\ ^{(i \in I \cup J \setminus \{j\})}] : A} \; (j \in J)$$

(Eq Subsumption) :
$$\frac{E \vdash_{\mathrm{T}} a \leftrightarrow a' : A \quad E \vdash_{\mathrm{T}} A <: B}{E \vdash_{\mathrm{T}} a \leftrightarrow a' : B}$$

(Eq Top) :
$$\frac{E \vdash_{\mathrm{O}} a : A \quad E \vdash_{\mathrm{O}} b : B}{E \vdash_{\mathrm{T}} a \leftrightarrow b : \mathsf{Top}}$$

(Eq Sub Object) where $I \cap J = \emptyset,\ A \equiv [\ell_i{:}B_i\ ^{i \in I}],\ A' \equiv [\ell_k{:}B_k\ ^{k \in I \cup J}]$:
$$\frac{E, x_i{:}A \vdash_{\mathrm{O}} b_i : B_i \quad (\forall i \in I) \quad E, x_j{:}A' \vdash_{\mathrm{O}} b_j : B_j \quad (\forall j \in J)}{E \vdash_{\mathrm{T}} [\ell_i = \varsigma(x_i^A)b_i\ ^{i \in I}] \leftrightarrow [\ell_k = \varsigma(x_k^{A'})b_k\ ^{k \in I \cup J}] : A}$$

(Abs Cong) :
$$\frac{E, x{:}A \vdash_{\mathrm{T}} b \leftrightarrow b' : B}{E \vdash_{\mathrm{T}} \lambda x^A.b \leftrightarrow \lambda x^A.b' : A \to B}$$

(App Cong) :
$$\frac{E \vdash_{\mathrm{T}} a \leftrightarrow a' : A \to B \quad E \vdash_{\mathrm{T}} b \leftrightarrow b' : A}{E \vdash_{\mathrm{T}} a(b) \leftrightarrow a'(b') : B}$$

(Object Cong) $A \equiv [\ell_i{:}B_i\ ^{i \in I}]$:
$$\frac{E, x_i{:}A \vdash_{\mathrm{T}} b_i \leftrightarrow b_i' : B_i \quad (\forall i \in I)}{E \vdash_{\mathrm{T}} [\ell_i = \varsigma(x_i^A)b_i\ ^{i \in I}] \leftrightarrow [\ell_i = \varsigma(x_i^A)b_i'\ ^{i \in I}] : A}$$

(Sel Cong) $A \equiv [\ell_i{:}B_i\ ^{i \in I}], i \in I$:
$$\frac{E \vdash_{\mathrm{T}} a \leftrightarrow a' : A}{E \vdash_{\mathrm{T}} a.\ell_i \leftrightarrow a'.\ell_i : B_i}$$

(Update Cong) $A \equiv [\ell_i{:}B_i\ ^{i \in I}], j \in I$:
$$\frac{E \vdash_{\mathrm{T}} a \leftrightarrow a' : A \quad E, x{:}A \vdash_{\mathrm{T}} b \leftrightarrow b' : B_j}{E \vdash_{\mathrm{T}} a.\ell_j \Leftarrow \varsigma(x^A)b \leftrightarrow a'.\ell_j \Leftarrow \varsigma(x^A)b' : A}$$

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399