Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 286 (2012) 57–71

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Typed vs. Untyped Realizability

Ulrich Berger[1] Tie Hou[2]

Computer Science Department
Swansea University
Swansea, UK

Abstract

We study the domain-theoretic semantics of a Church-style typed λ-calculus with constructors, pattern matching and recursion, and show that it is closely related to the semantics of its untyped counterpart. The motivation for this study comes from program extraction from proofs via realizability where one has the choice of extracting typed or untyped terms from proofs. Our result shows that under a certain regularity condition, the choice is irrelevant. The regularity condition is that in every use of a fixed point type $\mathrm{fix}\,\alpha.\rho$, α occurs only positively in ρ.

Keywords: Scott domain, typed and untyped λ-calculus, program extraction, logical relation, realizability.

1 Introduction

This paper is part of a research project aiming at a semantical foundation for program extraction from proofs [7]. It contributes to a soundness proof for a language of realizers of proofs involving inductive and coinductive definitions. The natural language of realizers for inductive and coinductive definitions is a typed lambda calculus with types modeling initial algebras and final coalgebras, and terms modeling structural recursion and corecursion. In this paper we study a more general calculus that allows fixed points of arbitrary type operators and definitions of functions by general recursion. The advantage of this generality is that our results will apply to all conceivable extensions of our theory of realizers of inductive and coinductive definitions.

We study the domain-theoretic semantics of a Church-style typed λ-calculus with constructors, pattern matching and recursion, and compare it with its untyped counterpart. We work with polymorphic types that allow fixed points of arbitrary type operators. A type ρ is interpreted as (the image of) a finitary projection $\langle\rho\rangle$,

[1] Email: u.berger@swansea.ac.uk

[2] Email: cshou@swansea.ac.uk

doi:10.1016/j.entcs.2012.08.005

following [2]. The main result (Theorem 4.15) relates the semantics of a typed term M with its untyped variant M^-: if M has type ρ, where ρ is a regular type, that is, fixed points are only taken of positive operators, then

$$\langle\rho\rangle[\![M^-]\!] = [\![M]\!].$$

The proof uses logical relations. We do not know whether the result also holds if ρ is not regular.

A similar problem was studied by Reynolds [13,14] who established a coherence between the typed and untyped meanings of expressions based on cpo models of a version of PCF. The main differences to our work are as follows: Reynolds considers simple types over the base types of natural numbers and booleans while we allow arbitrary recursive types. On the other hand, he includes subtyping which we do not. Regarding the typed semantics, Reynolds interprets typing derivations in a typed model while we interpret terms with a typed abstraction in an untyped model.

The motivation for this study comes from program extraction from proofs via realizability (see e.g. [6,4,7,8] for applications in constructive analysis) where one has the choice of extracting typed or untyped terms from proofs. Our result shows that if the extracted type is regular, the choice is irrelevant. In fact, regularity is a harmless restriction because in the intended realizability interpretation the types of realizing terms will always be regular. In [5] the soundness of a realizability interpretation based on a fragment of the untyped version of our calculus was proven, and the calculus was shown to be computationally adequate with respect to a domain-theoretic semantics (the same semantics we are considering here). In [9] it was shown that the extracted programs admit a Curry-style typing. In the present paper we provide the missing semantical link to Curry-style typing.

The application to realizability is also our motivation for working with Scott domains (instead of arbitrary cpos, as Reynolds does): the adequacy proof in [5] uses the fact that all semantic objects can be approximated by compact ones, hence we have to ensure that types are interpreted in a cartesian closed category of algebraic domains. This is achieved by interpreting types as finitary projections. Apart from that, the results of this paper could also be obtained using arbitrary cpos and embedding-retraction pairs.

The problem of relating typed and untyped realizability was also studied by Longley [10]. He used a condition called (constructive) logical full abstraction to connect realizability over typed and untyped structures by means of partial combinatory algebra.

Our paper is organized as follows. Section 2 introduces the syntax of types and typed terms, and typing rules for typed terms. Next, in Section 3, the semantics of types and typed terms are described in the setting of Scott domains and its correctness is proved. In Section 4, the relation between the domain-theoretic semantics of typed terms and the semantics of their untyped counterparts are studied. The proof uses logical relations, which are related to Tait's computability method and Girard's method of reducibility candidates. Finally, Section 5 concludes the paper and outlines some future work.

2 Types and terms

In this and the next section we study the syntax and semantics of types and typed terms. Untyped terms will be introduced in Section 4.

Definition 2.1 [Types] The set of types is defined by the following grammar:

$$\mathbf{Type} \ni \rho, \sigma, \tau ::= \alpha \mid \rho \to \sigma \mid \mathbf{1} \mid \rho \times \sigma \mid \rho + \sigma \mid \mathrm{fix}\, \alpha.\rho.$$

where α ranges over a set **TVar** of type variables. A fixed-point construction, $\mathrm{fix}\, \alpha.\rho$, binds all free occurrences of α in ρ.

We work with a Church-style typed lambda-calculus with constructors, pattern matching and recursion which we call *Language of Realizers* (**LoR**) because its terms are intended to be used as extracted programs from proofs obtained by a realizability interpretation.

We consider only the constructors Nil (nullary), Pair (binary), and Left, Right, In (unary). The intention behind the first four constructors should be obvious. The constructor In is used to model type fixed points up to isomorphism. Many definitions and results could be extended to an arbitrary set of constructors.

Definition 2.2 [Terms] The set of (Church-style typed) terms is defined by

$$\mathbf{LoR} \ni M, N, R_i ::= x \mid \lambda x : \rho.M \mid MN \mid \mathrm{rec}\, x : \rho.M \mid C(M_1, \ldots, M_n) \mid \\ \mathrm{case}\, M \,\mathrm{of}\, \{C_i(\boldsymbol{x_i}) \to R_i\}_{i \in \{1,\ldots,n\}}.$$

where x ranges over a set of variables **Var**, C is a constructor of arity n, and in $\mathrm{case}\, M \,\mathrm{of}\, \{C_i(\boldsymbol{x_i}) \to R_i\}_{i \in \{1,\ldots,n\}}$ all constructors C_i are distinct and each $\boldsymbol{x}_i$ is a vector of distinct variables whose length coincide with the arity of C_i. Lambda abstraction, $\lambda x : \rho.M$, and recursion, $\mathrm{rec}\, x : \rho.M$, bind all free occurrences of x in M, and a pattern matching clause, $C_i(\boldsymbol{x_i}) \to R_i$, binds all free occurrences of $\boldsymbol{x_i}$ in R_i.

We introduce typing rules for **LoR**-terms. A *type context* is a set of pairs $\Gamma := x_1 : \rho_1, \ldots, x_n : \rho_n$ (for notational convenience we omit the curly braces) where ρ_i are types and x_i are distinct variables. The set of variables $\{x_1, \ldots, x_n\}$ (which may be empty) is denoted by $\mathrm{dom}(\Gamma)$.

The relation $\Gamma \vdash M : \rho$ (M is a **LoR** term of type ρ in context Γ) is inductively defined as follows. Note that in the definition of terms (Definition 2.2), a case expression can have in general many clauses, but our typing rules only allow two or one clauses.

$$\Gamma \vdash \mathrm{Nil} : 1 \qquad\qquad \Gamma, x : \rho \vdash x : \rho$$

$$\frac{\Gamma, x : \rho \vdash M : \sigma}{\Gamma \vdash \lambda x : \rho . M : \rho \to \sigma} \qquad\qquad \frac{\Gamma, x : \tau \vdash M : \tau}{\Gamma \vdash \mathrm{rec}\ x : \tau . M : \tau}$$

$$\frac{\Gamma \vdash M : \rho \to \sigma \qquad \Gamma \vdash N : \rho}{\Gamma \vdash M\,N : \sigma} \qquad\qquad \frac{\Gamma \vdash M : \rho \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash \mathrm{Pair}(M, N) : \rho \times \sigma}$$

$$\frac{\Gamma \vdash M : \rho}{\Gamma \vdash \mathrm{Left}(M) : \rho + \sigma} \qquad\qquad \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \mathrm{Right}(M) : \rho + \sigma}$$

$$\frac{\Gamma \vdash M : \rho + \sigma \qquad \Gamma, x_1 : \rho \vdash L : \tau \qquad \Gamma, x_2 : \sigma \vdash R : \tau}{\Gamma \vdash \mathrm{case}\, M\ \text{of}\ \{\mathrm{Left}(x_1) \to L; \mathrm{Right}(x_2) \to R\} : \tau}$$

$$\frac{\Gamma \vdash M : \rho \times \sigma \qquad \Gamma, x : \rho, y : \sigma \vdash N : \tau}{\Gamma \vdash \mathrm{case}\, M\ \text{of}\ \{\mathrm{Pair}(x, y) \to N\} : \tau}$$

$$\frac{\Gamma \vdash M : \rho[\mathrm{fix}\, \alpha . \rho / \alpha]}{\Gamma \vdash \mathrm{In}(M) : \mathrm{fix}\, \alpha . \rho}$$

$$\frac{\Gamma \vdash M : \mathrm{fix}\, \alpha . \rho \qquad \Gamma, x : \rho[\mathrm{fix}\, \alpha . \rho / \alpha] \vdash N : \sigma}{\Gamma \vdash \mathrm{case}\, M\ \text{of}\ \{\mathrm{In}(x) \to N\} : \sigma}$$

3 Domain-theoretic semantics

We assume familiarity with the basic theory of Scott domains and the method of defining domains by recursive domain equations [15,1,3]. We omit the proofs of the most basic results since they are rather elementary, or can be found in the above cited literature. The reason for working with Scott domains is that all the semantic constructions we need are readily available, e.g. cartesian closure, solutions to recursive domain equation, recursive definition of functions, interpretation of types, including recursive types, as finitary projections. All these constructions are very elementary and do not require a heavy category-theoretical machinery.

By a *Scott-domain*, or *domain* for short, we mean a bounded complete ω-algebraic dcpo with least element. We will denote the least element of a domain by $\perp$. By $\mathbf{1}$ we denote the sole-element domain $\{\mathrm{Nil}\}$, and by $(D_1 + \ldots + D_n)_\perp$, $D \times E$, $[D \to E]$ the separated sum, cartesian product, and continuous function space of domains [3].

[3] These domain operations should not be confused with the syntactic constructors for types which for simplicity we denoted by the same symbols.

Due to ω-algebraicity, every element x of a domain D is the directed countable supremum of compact elements, where $y \in D$ is called *compact* if for every directed $A \subseteq D$, A has a supremum $\sqcup A$ and $y \sqsubseteq \sqcup A$ s.t. $y \sqsubseteq z$ for some $z \in A$. By D_c we denote the set of compact elements of D.

Definition 3.1 [Subdomain] $E \subseteq D$ is a *subdomain* of D if

(i) $\bot_D \in E$.

(ii) If $A \subseteq E$ and $\sqcup_D A$ exists in D, then $\sqcup_D A \in E$ and $\sqcup_D A = \sqcup_E A$.

(iii) If x is compact in E, then x is compact in D.

(iv) $\forall y \in D_c \forall x \in E(y \sqsubseteq x \to \exists y' \in E_c(y \sqsubseteq y' \sqsubseteq x))$.

Lemma 3.2 *Let $E \subseteq D$ be a subdomain of D. Then E is a domain.*

Proof. By verifying clauses (directed complete, algebraic, bounded complete) of the definition of domain. □

Following [2] we interpret types as finitary projections in D. Since the range of a finitary projection is a subdomain of D the semantics of types can be viewed as a domain. This approach provides an easy solution to the problem of defining the semantics of a fixed point type: one can simply take the least fixed point of a suitable continuous function on the domain $[D \to D]$.

Definition 3.3 [Finitary projection] $f : D \to D$ is a *projection* if

- f is continuous,
- $f \sqsubseteq \mathrm{id}$, i.e. $\forall x \in D. f(x) \sqsubseteq x$,
- $f \circ f = f$, i.e. $\forall x \in D. f(f(x)) = f(x)$.

A projection f is *finitary* if the range of f, denoted by $f(D)$, is a subdomain of D.

For $f \colon D \to D$ we set $\mathrm{Fix}(f) := \{x \in D \mid f(x) = x\}$. Obviously, if $f \circ f = f$, then $f(D) = \mathrm{Fix}(f)$.

In the following two lemmas we assume that $p : D \to D$ is a projection, and we set $p(D)_c := D_c \cap p(D)$. We omit their proofs since they are easy.

Lemma 3.4 *E is a subdomain of D if and only if there exists a finitary projection $p : D \to D$ such that $E = p(D)$.*

Lemma 3.5 *The following are equivalent:*

(a) p is finitary

(b) $\forall x \in D(A_x := \{a \in p(D)_c \mid a \sqsubseteq x\})$ is directed and $p(x) = \sqcup A_x$.

(c) $\exists A \subseteq D_c. \forall x \in D(p(x) = \sqcup\{a \in A \mid a \sqsubseteq x\})$.

Lemma 3.6 *If for all n, p_n is a finitary projection and $p_n \sqsubseteq p_{n+1}$, then $\sqcup_n p_n$ is a finitary projection.*

Proof. Let $p := \sqcup_n p_n$. By Lemma 3.5 (c), it suffices to show that $p(x) = \sqcup\{a \in p(D)_c \mid a \sqsubseteq x\}$. We first show that p is idempotent: $p(p(x)) = (\sqcup_n p_n)((\sqcup_m p_m)(x)) = \sqcup_n \sqcup_m (p_n(p_m(x))) \overset{(*)}{=} \sqcup_n(p_n(p_n(x))) = \sqcup_n(p_n(x)) = (\sqcup_n p_n)(x) = p(x)$. Equation $(*)$ easily follows from the fact that the double sequence $p_n(p_m(x))$ is increasing in m and in n. Hence, we get $p(D) = \{x \mid p(x) = x\}$, and therefore, $\sqcup\{a \in p(D)_c \mid a \sqsubseteq x\} = \sqcup\{a \in D_c \mid p(a) = a \wedge a \sqsubseteq x\} = \sqcup\{a \in D_c \mid \sqcup_n(p_n(a)) = a \wedge a \sqsubseteq x\} = \sqcup\{a \in D_c \mid \exists n.p_n(a) = a \wedge a \sqsubseteq x\}$ (by compactness) $= \sqcup_n(\sqcup\{a \in D_c \mid p_n(a) = a \wedge a \sqsubseteq x\}) = \sqcup_n(p_n(x)) = p(x)$. □

Now we define by a recursive domain equation a particular domain D which we will use to interpret types and terms.

Definition 3.7 We define the Scott domain D by the recursive domain equation:

$$\mathrm{D} \simeq (\mathbf{1} + \mathrm{D} + \mathrm{D} + \mathrm{D} + \mathrm{D} \times \mathrm{D} + [\mathrm{D} \to \mathrm{D}])_\perp$$

Using the constructors of **LoR** as names for the injections into the sum, each element in D has exactly one of the following forms: $\perp$, Nil, Left(a), Right(a), In(a), Pair(a, b), Fun(f), where a and b range over D, and f ranges over continuous function from D to D.

It will be convenient to use the continuous functions

$$\mathrm{case}_{C_1,\dots,C_n} : \mathrm{D} \to [\mathrm{D}^{\mathrm{arity}(C_1)} \to \mathrm{D}] \to \dots \to [\mathrm{D}^{\mathrm{arity}(C_n)} \to \mathrm{D}] \to \mathrm{D}$$

$$\mathrm{case}_{C_1,\dots,C_n}\ a\ f_1 \dots f_n := \begin{cases} f_i(\boldsymbol{b_i}) & \text{if } a = C_i(\boldsymbol{b_i}), \\ \perp & \text{otherwise.} \end{cases}$$

We also use an informal lambda-notation $\lambda a.f(a)$ and composition $f \circ g$ to define continuous functions on D. We do not prove the continuity in each case since this follows from well-known fact about the category of Scott domains and continuous functions. We also let LFP : $[\mathrm{D} \to \mathrm{D}] \to \mathrm{D}$ be the continuous least fixed point operator, which can be defined by $\mathrm{LFP}(f) = \bigsqcup_n f^n(\perp)$.

The following definition gives an unexpected interpretation of a type ρ as a finitary projection $\langle\rho\rangle$, but from this one can derive a more familiar definition as a set, namely the image (or, equivalently, set of fixed points) of $\langle\rho\rangle$. Also, $\langle\rho\rangle$ will be used later in Theorem 4.15, and act as a function (not only a type), providing a link between the two semantics.

Definition 3.8 [Semantics of types] For every type ρ we define
$\langle\rho\rangle : [[\mathrm{D} \to \mathrm{D}]^{\mathbf{TVar}} \to [\mathrm{D} \to \mathrm{D}]]$ [4]

[4] $[\mathrm{D} \to \mathrm{D}]^{\mathbf{TVar}}$ is the set of type environments, i.e., functions from **TVar** to $[\mathrm{D} \to \mathrm{D}]$.

$$\begin{aligned}
\langle \mathbf{1}\rangle\zeta(a) &= \mathrm{case}_{\mathrm{Nil}}\ a\ \mathrm{Nil} \qquad (= \begin{cases} \mathrm{Nil} \text{ if } a = \mathrm{Nil} \\ \bot \quad \text{otherwise} \end{cases}) \\
\langle \alpha\rangle\zeta(a) &= \zeta(\alpha)(a) \\
\langle \rho+\sigma\rangle\zeta(a) &= \mathrm{case}_{\mathrm{Left,Right}}\ a\ (\mathrm{Left}\circ\langle\rho\rangle\zeta)\ (\mathrm{Right}\circ\langle\sigma\rangle\zeta) \\
\langle \rho\times\sigma\rangle\zeta(a) &= \mathrm{case}_{\mathrm{Pair}}\ a\ (\lambda b_1 b_2.\mathrm{Pair}(\langle\rho\rangle\zeta(b_1),\langle\sigma\rangle\zeta(b_2))) \\
\langle \rho\to\sigma\rangle\zeta(a) &= \mathrm{case}_{\mathrm{Fun}}\ a\ (\lambda f.\mathrm{Fun}(\langle\sigma\rangle\zeta\circ f\circ\langle\rho\rangle\zeta)) \\
\langle \mathrm{fix}\,\alpha.\rho\rangle\zeta &= \mathrm{LFP}(\lambda p.\lambda a.\mathrm{case}_{\mathrm{In}}\ a\ (\lambda b.\mathrm{In}(\langle\rho\rangle\zeta[\alpha:=p](b))))
\end{aligned}$$

We set $[\![\rho]\!]\zeta := (\langle\rho\rangle\zeta)(\mathrm{D})$.

We call $\zeta : [\mathrm{D}\to\mathrm{D}]^{\mathbf{TVar}}$ a finitary projection if $\zeta(\alpha)$ is a finitary projection for all $\alpha \in \mathbf{TVar}$.

Lemma 3.9 *If ζ is a finitary projection, then $\langle\rho\rangle\zeta$ is a finitary projection.*

Proof. By induction on ρ using Lemma 3.4 and 3.5. We only look at the interesting case which is fixed point types. Let

$$f := \lambda p.\lambda a.\mathrm{case}_{\mathrm{In}}\ a\ (\lambda b.\mathrm{In}(\langle\rho\rangle\zeta[\alpha:=p](b))),$$

hence $\langle \mathrm{fix}\,\alpha.\rho\rangle\zeta = \mathrm{LFP}(f) = \sqcup_{n\in N} f^n(\bot)$. Therefore, by Lemma 3.6, it suffices to show that $\forall n. f^n(\bot)$ is a finitary projection. We do a side induction on n.

$n = 0$: To show $f^0(\bot)$ is a finitary projection. Since $f^0(\bot) = \bot$ and $\bot$ is a finitary projection, $f^0(\bot)$ is a finitary projection.

$n+1$: Assume $p := f^n(\bot)$ is a finitary projection. To show $f(p)$ is a finitary projection. We first show that $f(p)$ is a projection. We have

$$f(p) = \lambda a.\mathrm{case}_{\mathrm{In}}\ a\ (\lambda b.\mathrm{In}(\langle\rho\rangle\zeta[\alpha:=p](b))).$$

By definition of $\mathrm{case}_{\mathrm{In}}$, $f(p)$ is a continuous function, since, by I.H., $\langle\rho\rangle\zeta[\alpha:=p]$ is a continuous function/finitary projection. To show $f(p)\sqsubseteq \mathrm{id}$. We have

$$f(p)(x) = \begin{cases} \mathrm{In}(\langle\rho\rangle\zeta[\alpha:=p](b)) \text{ if } x = \mathrm{In}(b) \\ \bot \qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

where $\mathrm{In}(\langle\rho\rangle\zeta[\alpha:=p](b)) \sqsubseteq \mathrm{In}(b)$, since by I.H. $\langle\rho\rangle\zeta[\alpha:=p]$ is a projection. Therefore, $f(p)(x)\sqsubseteq x$. To show $f(p)(f(p)(x)) = f(p)(x)$. If $x = \mathrm{In}(b)$, we have $f(p)(f(p)(x)) = \mathrm{In}(\langle\rho\rangle\zeta[\alpha:=p](\langle\rho\rangle\zeta[\alpha:=p](b))) = \mathrm{In}(\langle\rho\rangle\zeta[\alpha:=p](b)) = f(p)(x)$, since by I.H. $\langle\rho\rangle\zeta[\alpha:=p]$ is idempotent. Otherwise, we get $f(p)(f(p)(x)) = \bot = f(p)(x)$. Therefore, $f(p)$ is a projection.

Now we show that $f(p)(\mathrm{D})$ is a subdomain of D. Let

$$A := \{\langle\rho\rangle\zeta[\alpha:=p](b) \mid b\in\mathrm{D}\} = \langle\rho\rangle\zeta[\alpha:=p](\mathrm{D}).$$

By I.H. $\langle\rho\rangle\zeta[\alpha:=p]$ is a finitary projection. Thus, A is a subdomain of D. By the definition of f, $f(p)(\mathrm{D}) = \mathrm{In}(A) \cup \{\bot\}$. It is easy to prove that $\mathrm{In}(A) \cup \{\bot\}$ is a subdomain of D. □

Now we are ready to define the semantics of **LoR**-terms. The leading idea in the definition of the value of a typed lambda-abstraction $\lambda x : \rho.M$ is that the domain of the resulting function is (the semantics of) ρ. Therefore, the incoming argument a is first projected down to ρ.

Definition 3.10 [Semantics of terms] For all environments $\zeta : [\mathrm{D} \to \mathrm{D}]^{\mathbf{TVar}}$, $\eta : \mathrm{D}^{\mathbf{Var}}$, and every **LoR** term M we define the value $[\![M]\!]^{\zeta}\eta \in \mathrm{D}$.

$$\begin{aligned}
[\![x]\!]^{\zeta}\eta &= \eta(x)\\
[\![C(M_1,\dots,M_n)]\!]^{\zeta}\eta &= C([\![M_1]\!]^{\zeta}\eta,\dots,[\![M_n]\!]^{\zeta}\eta)\\
[\![MN]\!]^{\zeta}\eta &= \mathrm{case}_{\mathrm{Fun}}\ ([\![M]\!]^{\zeta}\eta)\ (\lambda f. f([\![N]\!]^{\zeta}\eta))\\
[\![\lambda x : \rho.M]\!]^{\zeta}\eta &= \mathrm{Fun}(\lambda a.[\![M]\!]^{\zeta}\eta[x := \langle\rho\rangle\zeta(a)])\\
[\![\mathrm{rec}\, x : \tau.M]\!]^{\zeta}\eta &= \mathrm{LFP}(\lambda a.[\![M]\!]^{\zeta}\eta[x := \langle\tau\rangle\zeta(a)])\\
[\![\mathrm{case}\, M\, \mathrm{of}\, \{C_i(\boldsymbol{x_i}) \to R_i\}_i]\!]^{\zeta}\eta &= \mathrm{case}_{C_1,\dots,C_n}\ ([\![M]\!]^{\zeta}\eta)\ (\lambda \boldsymbol{a}.[\![R_i]\!]^{\zeta}\eta[\boldsymbol{x_i} := \boldsymbol{a}])_i
\end{aligned}$$

One can prove the following soundness theorem, stating that if from a context Γ we can derive **LoR** term M with type ρ, and for every variable $x \in \mathrm{dom}(\Gamma)$, $\eta(x)$ is an element of $[\![\Gamma(x)]\!]\zeta$ (we will write $\eta \in [\![\Gamma]\!]\zeta$ for this), then the value of term M is an element of the value of type ρ.

Theorem 3.11 (Soundness For LoR terms) *Let ζ be a finitary projection. If $\Gamma \vdash M : \rho$ and $\eta \in [\![\Gamma]\!]\zeta$, then $[\![M]\!]^{\zeta}\eta \in [\![\rho]\!]\zeta$.*

Proof. By induction on the definition of the relation $\Gamma \vdash M : \rho$. □

4 Relating typed and untyped terms

We now relate the semantics of typed terms with the semantics of untyped terms which are defined exactly as typed terms except that the type annotations for abstraction and recursion are omitted:

Definition 4.1 [Untyped terms]

$$\begin{aligned}
\mathbf{LoR}^{-} \ni M, N, R_i ::= x \mid \lambda x.M \mid MN \mid \mathrm{rec}\, x.M \mid C(M_1,\dots,M_n) \mid\\
\mathrm{case}\, M\, \mathrm{of}\, \{C_i(\boldsymbol{x_i}) \to R_i\}_{i\in\{1,\dots,n\}}
\end{aligned}$$

The same provisions made in Definition 2.2 for typed terms apply here.

The semantics of untyped terms is straightforward. It can be defined exactly as in the typed case except that the type environment $\zeta : [\mathrm{D} \to \mathrm{D}]^{\mathbf{TVar}}$ and finitary projections involved in typed abstraction and recursion are omitted.

Definition 4.2 [Semantics of untyped terms] For every environment $\eta : \mathbf{Var} \to \mathrm{D}$ and every $\mathbf{LoR}^{-}$ term M we define the value $[\![M]\!]\eta \in \mathrm{D}$.

$$
\begin{aligned}
[\![x]\!]\eta &= \eta(x)\\
[\![C(M_1,\ldots,M_n)]\!]\eta &= C([\![M_1]\!]\eta,\ldots,[\![M_n]\!]\eta)\\
[\![MN]\!]\eta &= \mathrm{case}_{\mathrm{Fun}}\ ([\![M]\!]\eta)\ (\lambda f.f([\![N]\!]\eta))\\
[\![\lambda x.M]\!]\eta &= \mathrm{Fun}(\lambda a.[\![M]\!]\eta[x:=a])\\
[\![\mathrm{rec}\,x.M]\!]\eta &= \mathrm{LFP}(\lambda a.[\![M]\!]\eta[x:=a])\\
[\![\mathrm{case}\,M\ \mathrm{of}\ \{C_i(x_i)\to R_i\}_i]\!]\eta &= \mathrm{case}_{C_1,\ldots,C_n}\ ([\![M]\!]\eta)\ (\lambda \boldsymbol{a}.[\![R_i]\!]\eta[\boldsymbol{x_i}:=\boldsymbol{a}])_i
\end{aligned}
$$

Our main result, the Coincidence Theorem 4.15, only applies to terms that are typed w.r.t. to a restricted notion of types where fixed point types $\mathrm{fix}\,\alpha.\rho$ are allowed only if ρ is positive in α.

Definition 4.3 [ρ positive/negative in α]
α is positive in α.
$\mathbf{1}$ is positive and negative in α.
$\rho\to\sigma$ is positive in α if ρ is negative in α and σ is positive in α.
$\rho\to\sigma$ is negative in α if ρ is positive in α and σ is negative in α.
$\rho+\sigma$ and $\rho\times\sigma$ are positive in α if ρ and σ are positive in α.
$\rho+\sigma$ and $\rho\times\sigma$ are negative in α if ρ and σ are negative in α.
$\mathrm{fix}\,\beta.\rho$ is positive in α if $\alpha=\beta$ or ρ is positive in α.
$\mathrm{fix}\,\beta.\rho$ is negative in α if $\alpha=\beta$ or ρ is negative in α.

Definition 4.4 [Regular types] We define *regular* types ρ as follows.
$\mathbf{1}$ is regular.
α is regular.
$\rho+\sigma$, $\rho\times\sigma$, $\rho\to\sigma$ are regular if ρ and σ are regular.
$\mathrm{fix}\,\alpha.\rho$ is regular if ρ is regular and ρ is positive in α.

Example 4.5 $\mathrm{fix}\,\alpha.(\alpha\to\alpha)$ is not regular, since $\alpha\to\alpha$ is not positive in α.

In the following all types are assumed to be regular.

To prove our main result we define a logical relation $\sim_\rho^R\subseteq \mathrm{D}\times\mathrm{D}$ which can intuitively be understood as a notion of equivalence of elements of a *regular* type ρ. We use the informal (second-order) lambda abstraction $\Lambda r\subseteq \mathrm{D}^2$. to define functions on the set $\mathcal{P}(\mathrm{D}^2)$ of binary relations on D.

Definition 4.6 and Lemma 4.8 below should be considered simultaneously, since in the clause of $\mathrm{fix}\,\alpha.\rho$, the clause is well-defined only if $\sim_\rho^{R[\alpha:=r]}$ is monotone in r.

Definition 4.6 [Logical Relation] In the following definition it assumed that $R\in\mathcal{P}(\mathrm{D}^2)^{\mathbf{TVar}}$.

$$\begin{aligned}
\sim^R_{\mathbf{1}} &:= \{(\bot,\bot),(\mathrm{Nil},\mathrm{Nil})\}\\
\sim^R_{\alpha} &:= R(\alpha)\\
\sim^R_{\rho_1\times\rho_2} &:= \{(\bot,\bot)\}\cup\{(\mathrm{Pair}(a_1,a_2),\mathrm{Pair}(b_1,b_2)) \mid a_1 \sim^R_{\rho_1} b_1, a_2 \sim^R_{\rho_2} b_2\}\\
\sim^R_{\rho_1+\rho_2} &:= \{(\bot,\bot)\}\cup\{(\mathrm{Left}(a_1),\mathrm{Left}(b_1)) \mid a_1 \sim^R_{\rho_1} b_1\}\\
&\qquad\qquad\quad \cup\{(\mathrm{Right}(a_2),\mathrm{Right}(b_2)) \mid a_2 \sim^R_{\rho_2} b_2\}\\
\sim^R_{\rho\to\sigma} &:= \{(\bot,\bot)\}\cup\{(\mathrm{Fun}(f),\mathrm{Fun}(g)) \mid\\
&\qquad\qquad\quad \forall a,b\in \mathrm{D}(a\sim^R_\rho b \Rightarrow f(a)\sim^R_\sigma g(b))\\
\sim^R_{\mathrm{fix}\,\alpha.\rho} &:= \mathrm{LFP}(\Lambda r\subseteq \mathrm{D}^2.\{(\bot,\bot)\}\cup\{(\mathrm{In}(a),\mathrm{In}(b)) \mid a \sim^{R[\alpha:=r]}_\rho b\})
\end{aligned}$$

Remark 4.7 Logical relations [12] have been used successfully to prove properties of typed systems. Famous examples are the strong normalization proofs by Tait and Girard using logical relations called computability predicates or reducibility candidates. The crucial feature of a logical relation is that it is a family of relations indexed by types and defined by induction on types such that all type constructors are interpreted by their logical interpretations, e.g. $\to$ is interpreted as logical implication.

Lemma 4.8

(1) If ρ is positive in α, then $\Lambda r\subseteq \mathrm{D}^2.\ \sim^{R[\alpha:=r]}_\rho$ is monotone.

(2) If ρ is negative in α, then $\Lambda r\subseteq \mathrm{D}^2.\ \sim^{R[\alpha:=r]}_\rho$ is anti-monotone.

Proof. By induction on ρ. □

The notion of *admissibility* has been used in [1] and generalized in [11], where it is used to prove properties of least fixed points.

Definition 4.9 [Admissible relation] A relation $r\subseteq \mathrm{D}^2$ is called *admissible* if it satisfies

(i) $(\bot,\bot)\in r$.

(ii) If $(d_n,d'_n)\in r$ and $(d_n,d'_n)\sqsubseteq(d_{n+1},d'_{n+1})$ for all n, then $\sqcup_{n\in N}(d_n,d'_n)\in r$.

Note that a finite relation $r\subseteq \mathrm{D}^2$ with $(\bot,\bot)\in r$ is always admissible.

Let $\mathrm{Ad} := \{r\subseteq \mathrm{D}^2 \mid r \text{ is admissible}\}$.

Lemma 4.10 Ad *is a complete lattice. Moreover, if $\mathcal{A}$ is a directed set of admissible sets, then $\bigcup\mathcal{A}$ is admissible.*

Proof. Easy. □

We call $R\in(\mathrm{D}^2)^{\mathbf{TVar}}$ admissible if $R(\alpha)$ is admissible for all $\alpha\in\mathbf{TVar}$.

Lemma 4.11 *If R is admissible, then $\sim^R_\rho$ is admissible.*

Proof. By induction on ρ. We only look at the most interesting case, which is $\mathrm{fix}\,\alpha.\rho$. We have $\sim^R_{\mathrm{fix}\,\alpha.\rho} = \mathrm{LFP}(\Phi)$ where

$$\Phi : \mathcal{P}(\mathrm{D}^2) \to \mathcal{P}(\mathrm{D}^2)$$
$$\Phi(r) \ = \ \{(\bot,\bot)\} \cup \{(\mathrm{In}(a),\mathrm{In}(b)) \mid a \sim_{\rho}^{R[\alpha:=r]} b\}.$$

Clearly $(\bot,\bot) \in \Phi(r)$ for every $r \subseteq \mathrm{D}^2$. Hence, we have $\mathrm{LFP}(\Phi) = r_{\beta_0}$ for some ordinal β_0, where for every ordinal β, r_β is defined by $r_0 = \{\}$, $r_{\beta+1} = \Phi(r_\beta)$, $r_\lambda = \bigcup\{r_\beta \mid \beta < \lambda\}$ (λ a limit ordinal).

It is easy to see that if $r \in \mathrm{Ad}$, then $\Phi(r) \in \mathrm{Ad}$, i.e. $\Phi : \mathrm{Ad} \to \mathrm{Ad}$. Hence by induction on ordinals and Lemma 4.10 it follows that all r_β where $\beta > 0$ are admissible. □

Definition 4.12 [Compatibility] Let $r \subseteq \mathrm{D}^2$ and $p \in [\mathrm{D} \to \mathrm{D}]$. We call r and p *compatible*, in symbols $r \approx p$, if

(i) $\forall a,b \in \mathrm{D}\,(r(a,b) \to p(a) = p(b))$.

(ii) $\forall a \in \mathrm{D}\ r(p(a),p(a))$.

(iii) $\forall a,b \in \mathrm{D}\,(r(a,b) \to r(p(a),b))$.

We call $R \in \mathcal{P}(\mathrm{D}^2)^{\mathbf{TVar}}$ and $\zeta \in [\mathrm{D} \to \mathrm{D}]^{\mathbf{TVar}}$ compatible, in symbols $R \approx \zeta$ if $R(\alpha) \approx \zeta(\alpha)$ holds for all $\alpha \in \mathbf{TVar}$.

To obtain an example of compatibility one may take any idempotent $p \in [\mathrm{D} \to \mathrm{D}]$ and define $r \subseteq \mathrm{D}^2$ by $r := \{(a,b) \in \mathrm{D}^2 \mid p(a) = p(b)\}$. Then, clearly, $r \approx p$.

Lemma 4.13 *If R is admissible, ζ is a finitary projection, and $R \approx \zeta$, then $\sim_\rho^R \approx \langle\rho\rangle\zeta$.*

Proof. We write $r \approx_{i,ii} p$ for the notion of compatibility obtained by deleting property (iii) in Definition 4.12. Similarly, $r \approx_{iii} p$ means compatibility where properties (i) and (ii) are deleted. The notions $R \approx_{i,ii} \zeta$ and $R \approx_{iii} \zeta$ are defined mutatis mutandis as in Definition 4.12.

We show that if R is admissible and $\zeta(\alpha)$ is a finitary projection, then:

(1) If $R \approx_{i,ii} \zeta$, then $\sim_\rho^R \approx_{i,ii} \langle\rho\rangle\zeta$.

(2) If $R \approx_{iii} \zeta$, then $\sim_\rho^R \approx_{iii} \langle\rho\rangle\zeta$.

Both statements are proved by induction on ρ. We only look at the interesting cases which are function and fixed point types.

(1) **Case** $\rho \to \sigma$: (i) Assume $a \sim_{\rho\to\sigma}^R b$. We have to show $\langle\rho \to \sigma\rangle\zeta(a) = \langle\rho \to \sigma\rangle\zeta(b)$. If $a = b = \bot$, then this is trivial. If $a = \mathrm{Fun}(f)$ and $b = \mathrm{Fun}(g)$. By the definition of $\sim_{\rho\to\sigma}^R$ (Definition 4.6), we have

$$\forall c,d \in \mathrm{D}(c \sim_\rho^R d \Rightarrow f(c) \sim_\sigma^R g(d)) \qquad (*)$$

By Definition 3.8, we have $\langle\rho \to \sigma\rangle\zeta(f) = \mathrm{Fun}(\langle\sigma\rangle\zeta \circ f \circ \langle\rho\rangle\zeta)$ and $\langle\rho \to \sigma\rangle\zeta(g) = \mathrm{Fun}(\langle\sigma\rangle\zeta \circ g \circ \langle\rho\rangle\zeta)$. Let $c \in \mathrm{D}$. We need to show $\langle\sigma\rangle\zeta(f(\langle\rho\rangle\zeta(c))) = \langle\sigma\rangle\zeta(g(\langle\rho\rangle\zeta(c)))$. By induction hypothesis (ii) for ρ, we have $\langle\rho\rangle\zeta(c) \sim_\rho^R \langle\rho\rangle\zeta(c)$. Then by $(*)$ we have $f(\langle\rho\rangle\zeta(c)) \sim_\sigma^R g(\langle\rho\rangle\zeta(c))$. And then by induction hypothesis (i) for σ, we have $\langle\sigma\rangle\zeta(f(\langle\rho\rangle\zeta(c))) = \langle\sigma\rangle\zeta(g(\langle\rho\rangle\zeta(c)))$.

(ii) Let $a \in \mathrm{D}$. We have to show $\langle\rho \to \sigma\rangle\zeta(a) \sim^R_{\rho\to\sigma} \langle\rho \to \sigma\rangle\zeta(a)$. If $a = \bot$. Then $\langle\rho \to \sigma\rangle\zeta(a) = \bot$. Thus $\bot \sim^R_{\rho\to\sigma} \bot$. If $a = \mathrm{Fun}(f)$, then, by Definition 3.8, we have $\langle\rho \to \sigma\rangle\zeta(a) = \mathrm{Fun}(\langle\sigma\rangle\zeta \circ f \circ \langle\rho\rangle\zeta)$. We need to show $\mathrm{Fun}(\langle\sigma\rangle\zeta \circ f \circ \langle\rho\rangle\zeta) \sim^R_{\rho\to\sigma}$ $\mathrm{Fun}(\langle\sigma\rangle\zeta \circ f \circ \langle\rho\rangle\zeta)$. By the definition of $\sim^R_{\rho\to\sigma}$ (Definition 4.6), it is to show

$$\forall c, d \in \mathrm{D}(c \sim^R_\rho d \Rightarrow \langle\sigma\rangle\zeta(f(\langle\rho\rangle\zeta(c))) \sim^R_\sigma \langle\sigma\rangle\zeta(f(\langle\rho\rangle\zeta(d)))).$$

Assume $c \sim^R_\rho d$. By induction hypothesis (i) for ρ, we have $\langle\rho\rangle\zeta(c) = \langle\rho\rangle\zeta(d)$. Then $f(\langle\rho\rangle\zeta(c)) = f(\langle\rho\rangle\zeta(d))$. And then by induction hypothesis (ii) for σ, we have $\langle\sigma\rangle\zeta(f(\langle\rho\rangle\zeta(c))) \sim^R_\sigma \langle\sigma\rangle\zeta(f(\langle\rho\rangle\zeta(d)))$.

Case $\mathrm{fix}\,\alpha.\rho$: By Definition 3.8, we have $\langle\mathrm{fix}\,\alpha.\rho\rangle\zeta = \bigsqcup_n p_n$ where $p_0 = \lambda a.\bot$, $p_{n+1} = \lambda a.\mathrm{case}_{\mathrm{In}}\ a\ (\lambda b.\mathrm{In}(\langle\rho\rangle\zeta[\alpha := p_n](b)))$. We set $r := \sim^R_{\mathrm{fix}\,\alpha.\rho}$ and show $r \approx_{i,ii} p_n$ by a side induction on n. This will be sufficient, since, as one easily sees, because r is admissible (by Lemma 4.11), the conditions (i) and (ii) of compatibility are closed under taking directed suprema of the right argument. Hence from $r \approx_{i,ii} p_n$ for all n it follows $r \approx_{i,ii} p$.

$n = 0$: (i) is trivial since p_0 is constant. (ii) holds since $p_0(a) = \bot$ and $(\bot, \bot) \in r$ since r is admissible, by Lemma 4.11.

$n + 1$: (i) Assume $r(a, b)$. We have to show $p_{n+1}(a) = p_{n+1}(b)$. If $a = b = \bot$, the equation trivially holds. Now assume $a = \mathrm{In}(c)$ and $b = \mathrm{In}(d)$. Then $p_{n+1}(a) = \mathrm{In}(\langle\rho\rangle\zeta[\alpha := p_n](c))$ and $p_{n+1}(b) = \mathrm{In}(\langle\rho\rangle\zeta[\alpha := p_n](d))$, and we have $c \sim^{R[\alpha:=r]}_\rho d$ by the definition of r. By the side induction hypothesis, $r \approx_{i,ii} p_n$. Hence, by the main induction hypothesis, $\sim^{R[\alpha:=r]}_\rho \approx_{i,ii} \langle\rho\rangle\zeta[\alpha := p_n]$, since r is admissible and p_n is a finitary projection (see proof of Lemma 3.9). It follows $\langle\rho\rangle\zeta[\alpha := p_n](c) = \langle\rho\rangle\zeta[\alpha := p_n](d)$ and therefore $p_{n+1}(a) = p_{n+1}(b)$.

(ii) Let $a \in D$. We have to show $r(p_{n+1}(a), p_{n+1}(a))$. If $a = \bot$, then $p_{n+1}(a) = \bot$ and $r(\bot, \bot)$ holds by admissibility of r. If $a = \mathrm{In}(c)$, then $p_{n+1}(a) = \mathrm{In}(\langle\rho\rangle\zeta[\alpha := p_n](c))$. By the side induction hypothesis, $r \approx_{i,ii} p_n$. Hence, by the main induction hypothesis, $\sim^{R[\alpha:=r]}_\rho \approx_{i,ii} \langle\rho\rangle\zeta[\alpha := p_n]$, since r is admissible and p_n is a finitary projection (see proof of Lemma 3.9). It follows $\langle\rho\rangle\zeta[\alpha := p_n](c) \sim^{R[\alpha:=r]}_\rho \langle\rho\rangle\zeta[\alpha := p_n](c)$ and therefore $p_{n+1}(a) \sim^R_{\mathrm{fix}\,\alpha.\rho} p_{n+1}(a)$, i.e. $r(p_{n+1}(a), p_{n+1}(a))$.

(2) **Case** $\rho \to \sigma$: Assume $a \sim^R_{\rho\to\sigma} b$. We have to show $\langle\rho \to \sigma\rangle\zeta(a) \sim^R_{\rho\to\sigma} b$. If $a = b = \bot$. Then $\langle\rho \to \sigma\rangle\zeta(a) = \bot = b$. Thus $\bot \sim^R_{\rho\to\sigma} \bot$. If $a = \mathrm{Fun}(f)$ and $b = \mathrm{Fun}(g)$, then by the definition of $\sim^R_{\rho\to\sigma}$, we have

$$\forall c, d \in \mathrm{D}(c \sim^R_\rho d \Rightarrow f(c) \sim^R_\sigma g(d)) \qquad (*)$$

By Definition 3.8, we have $\langle\rho \to \sigma\rangle\zeta(a) = \mathrm{Fun}(\langle\sigma\rangle\zeta \circ f \circ \langle\rho\rangle\zeta)$. We need to show $\mathrm{Fun}(\langle\sigma\rangle\zeta \circ f \circ \langle\rho\rangle\zeta) \sim^R_{\rho\to\sigma} \mathrm{Fun}(g)$. By the definition of $\sim^R_{\rho\to\sigma}$, it is to show $\forall c, d \in \mathrm{D}(c \sim^R_\rho d \Rightarrow \langle\sigma\rangle\zeta(f(\langle\rho\rangle\zeta(c))) \sim^R_\sigma g(d))$. Assume $c \sim^R_\rho d$. By induction hypothesis for ρ, we have $\langle\rho\rangle\zeta(c) \sim^R_\rho d$. Then by $(*)$ we have $f(\langle\rho\rangle\zeta(c)) \sim^R_\sigma g(d)$, and by induction hypothesis for σ, we have $\langle\sigma\rangle\zeta(f(\langle\rho\rangle\zeta(c))) \sim^R_\sigma g(d)$.

Case $\mathrm{fix}\,\alpha.\rho$: Set $r := \sim^R_{\mathrm{fix}\,\alpha.\rho}$ and $p := \langle\mathrm{fix}\,\alpha.\rho\rangle\zeta$. We have to show that $r(a, b)$ implies $r(p(a), b)$, i.e. $r \subseteq s$ where $s := \{(a, b) \mid r(p(a), b)\}$. We verify that $r \cap s \approx_{iii}$

p holds. Indeed, if $(r \cap s)(a,b)$, then $r(p(a),b)$ since $s(a,b)$ holds, and hence, since by Lemma 3.9 p is idempotent, $r(p(p((a)),b)$ since $s(p(a),b)$ holds, i.e. $(r \cap s)(p(a),b)$. Since r is the least fixed point of the operator $\Phi := \Lambda r.\{(\bot,\bot)\} \cup \{(\mathrm{In}(a),\mathrm{In}(b)) \mid a \sim_\rho^{R[\alpha:=r]} b\}$ we can attempt to prove the inclusion $r \subseteq s$ by induction. In fact we use the *strong induction principle* (see, for example, [8]) according to which it suffices to show $\Phi(r \cap s) \subseteq s$ (instead of $\Phi(s) \subseteq s$). Clearly $(\bot,\bot) \in s$. Hence we assume $a \sim_\rho^{R[\alpha:=r\cap s]} b$ and have to show $r(p(\mathrm{In}(a)),\mathrm{In}(b))$. Since $p(\mathrm{In}(a)) = \mathrm{In}(\langle\rho\rangle\zeta[\alpha := p](a))$ and $r = \Phi(r)$, we have to show $\Phi(r)(\mathrm{In}(\langle\rho\rangle\zeta[\alpha := p](a)),\mathrm{In}(b))$, i.e., $\langle\rho\rangle\zeta[\alpha := p](a) \sim_\rho^{R[\alpha:=r]} b$. By induction hypothesis, and because $r \cap s \approx_{iii} p$, $\langle\rho\rangle\zeta[\alpha := p](a) \sim_\rho^{R[\alpha:=r\cap s]} b$ and hence $\langle\rho\rangle\zeta[\alpha := p](a) \sim_\rho^{R[\alpha:=r]} b$, by monotonicity (Lemma 4.8). □

Let $\eta \sim_\Gamma^R \eta'$ denote the following: for all $x \in \mathrm{dom}(\Gamma)$, $\eta(x) \sim_{\Gamma(x)}^R \eta'(x)$. Let $\Gamma \vdash^{\mathrm{reg}} M : \rho$ mean that $\Gamma \vdash M : \rho$ has been derived using regular types only.

Let M^- be the untyped term obtained from the Church-style term M by deleting the type information in lambda abstractions. The following lemma is the core of the proof of the Coincidence Theorem.

Lemma 4.14 *Assume R is admissible, ζ is a finitary projection, and $R \approx \zeta$. If $\Gamma \vdash^{\mathrm{reg}} M : \rho$ and $\eta \sim_\Gamma^R \eta'$, then $[\![M]\!]^\zeta\eta \sim_\rho^R [\![M^-]\!]\eta'$.*

Proof. By induction on the definition of the relation $\Gamma \vdash^{\mathrm{reg}} M : \rho$.

The interesting cases are lambda abstraction and recursion.

(a) $\dfrac{\Gamma, x : \rho \vdash^{\mathrm{reg}} M : \sigma}{\Gamma \vdash^{\mathrm{reg}} \lambda x : \rho.M : \rho \to \sigma}$.

We have to show $[\![\lambda x : \rho.M]\!]^\zeta\eta \sim_{\rho\to\sigma}^R [\![\lambda x.M^-]\!]\eta'$. By Definition 3.10 and 4.2, we have

$$[\![\lambda x : \rho.M]\!]^\zeta\eta = \mathrm{Fun}(f) \quad \text{where } f(a) = [\![M]\!]^\zeta\eta[x := \langle\rho\rangle\zeta(a)],$$

$$[\![\lambda x.M^-]\!]\eta' = \mathrm{Fun}(g) \qquad \text{where } g(b) = [\![M^-]\!]\eta'[x := b].$$

It is to show $\mathrm{Fun}(f) \sim_{\rho\to\sigma}^R \mathrm{Fun}(g)$. By definition of our logical relation (Definition 4.6), it is to show

$$\forall a,b \in \mathrm{D}(a \sim_\rho^R b \Rightarrow f(a) \sim_\sigma^R g(b)) \tag{$*$}$$

By induction hypothesis for σ we have

$$\forall a,b \in \mathrm{D}(a \sim_\rho^R b \Rightarrow [\![M]\!]^\zeta\eta[x := a] \sim_\sigma^R [\![M^-]\!]\eta'[x := b]) \tag{IH}$$

To prove $(*)$, assume $a \sim_\rho^R b$. By Lemma 4.13 (iii) we have $\langle\rho\rangle\zeta(a) \sim_\rho^R b$. Hence $[\![M]\!]^\zeta\eta[x := \langle\rho\rangle\zeta(a)] \sim_\sigma^R [\![M^-]\!]\eta'[x := b])$.

(b) $\dfrac{\Gamma, x : \tau \vdash^{\mathrm{reg}} M : \tau}{\Gamma \vdash^{\mathrm{reg}} \mathrm{rec}\ x : \tau.M : \tau}$.

We have to show $[\![\mathrm{rec}\, x : \tau.M]\!]^\zeta\eta \sim_\tau^R [\![\mathrm{rec}\, x.M^-]\!]\eta'$. By Definition 3.10 and 4.2, we have

$$[\![\mathrm{rec}\, x : \tau.M]\!]^{\zeta}\eta = \mathrm{LFP}(f) \quad \text{where } f(a) = [\![M]\!]^{\zeta}\eta[x := \langle\tau\rangle\zeta(a)],$$

$$[\![\mathrm{rec}\, x.M^{-}]\!]\eta' = \mathrm{LFP}(g) \qquad \text{where } g(b) = [\![M^{-}]\!]\eta'[x := b].$$

Now we have to show $\mathrm{LFP}(f) \sim^{R}_{\tau} \mathrm{LFP}(g)$. By definition it is to show $\sqcup_n f^n(\bot) \sim^{R}_{\tau} \sqcup_n g^n(\bot)$. Since, by Lemma 4.11, $\sim^{R}_{\tau}$ is admissible, it suffices to show that $\forall n. f^n(\bot) \sim^{R}_{\tau} g^n(\bot)$. We do a side induction on n.

$n = 0$: To show $f^0(\bot) \sim^{R}_{\tau} g^0(\bot)$, i.e. $\bot \sim^{R}_{\tau} \bot$. This holds by Definition 4.6.

$n + 1$: Assume as side induction hypothesis, $f^n(\bot) \sim^{R}_{\tau} g^n(\bot)$, to show $f^{(n+1)}(\bot) \sim^{R}_{\tau} g^{(n+1)}(\bot)$. We have

$$f^{(n+1)}(\bot) = f(f^n(\bot)) = [\![M]\!]^{\zeta}\eta[x := \langle\tau\rangle\zeta(f^n(\bot))],$$
$$g^{(n+1)}(\bot) = g(g^n(\bot)) = [\![M^{-}]\!]\eta'[x := g^n(\bot)].$$

It is to show $[\![M]\!]^{\zeta}\eta[x := \langle\tau\rangle\zeta(f^n(\bot))] \sim^{R}_{\tau} [\![M^{-}]\!]\eta'[x := g^n(\bot)]$. By side induction hypothesis, we have $\langle\tau\rangle\zeta(f^n(\bot)) \sim^{R}_{\tau} g^n(\bot)$ by Lemma 4.13 (iii). By main induction hypothesis we have $[\![M]\!]^{\zeta}\eta[x := \langle\tau\rangle\zeta(f^n(\bot))] \sim^{R}_{\tau} [\![M^{-}]\!]\eta'[x := g^n(\bot)]$.

□

The above lemma (Lemma 4.14) yields as an immediate consequence our main result that if from a context Γ we can derive a **LoR** term M with regular type ρ, and for every variable $x \in \mathrm{dom}(\Gamma)$, $\eta(x)$ is an element of $[\![\Gamma(x)]\!]\zeta$, then the value of M and its corresponding untyped term M^{-} coincide up to the finitary projection $\langle\rho\rangle\zeta$.

Theorem 4.15 (Coincidence) *If $\Gamma \vdash^{\mathrm{reg}} M : \rho$ and $\eta \in [\![\Gamma]\!]\zeta$ where ζ is a finitary projection, then $[\![M]\!]^{\zeta}\eta = \langle\rho\rangle\zeta([\![M^{-}]\!]\eta)$.*

Proof. Given a finitary projection ζ, we define $R(\alpha) := \{(a, b) \in \mathrm{D}^2 \mid \zeta(\alpha)(a) = p(b)\}$. Then $R \approx \zeta$, as explained in the example following Definition 4.12. By Lemma 4.14, we then have $[\![M]\!]^{\zeta}\eta \sim^{R}_{\rho} [\![M^{-}]\!]\eta$. By Lemma 4.13 (i), we get $\langle\rho\rangle\zeta([\![M]\!]^{\zeta}\eta) = \langle\rho\rangle\zeta([\![M^{-}]\!]\eta)$. Then, by Soundness Theorem 3.11 and the definition of $\langle\rho\rangle\zeta(\mathrm{D})$, we have $[\![M]\!]^{\zeta}\eta = \langle\rho\rangle\zeta([\![M]\!]^{\zeta}\eta)$. Thus, $[\![M]\!]^{\zeta}\eta = \langle\rho\rangle\zeta([\![M^{-}]\!]\eta)$. □

5 Conclusion

We have studied a domain-theoretic semantics for Church-style system **LoR** of typed lambda terms and proved that, when restricted to regular types, it is closely related to its untyped counterpart. The proof uses hybrid logical relations. The reason for studying this domain-theoretic semantics is that it allows for very simply and elegant proofs of computational adequacy, and hence the correctness of program extraction.

Our results could be easily extended to also include full second-order polymorphism $\forall\alpha.\rho$, $\exists\alpha.\rho$ as in [2], but for our application, simple parametric and recursive types are sufficient.

As future work we intend to investigate whether the requirement of regularity is indeed necessary for our result to hold. Furthermore, we plan to compare the

Church-style system with a corresponding Curry-style system.

Acknowledgements

We are grateful to the anonymous referees for providing valuable suggestions and drawing our attention to related work.

References

[1] Abramsky, S., and Jung, A., *Domain theory.* In Handbook of Logic in Computer Science (1994), S. Abramsky, D. Gabbay, and T. S. E. Maibaum, Eds., Oxford University Press, 1-168.

[2] Amadio, R. M., Bruce, K. B., and Longo, G., *The finitary projection model for second order lambda calculus and solutions to higher order domain equations.* In First Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press (1986) 122-130.

[3] Amadio, R. M., and Curien, P.-L., "Domains and Lambda-Calculi", vol. **46** of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.

[4] Berger, U., *From coinductive proofs to exact real arithmetic.* LNCS **5771** (2009) 132-146.

[5] Berger, U., *Realisability for Induction and Coinduction with Applications to Constructive Analysis.* Jour. Universal Comput. Sci. **16(18)** (2010), 2535-2555.

[6] Berger, U., and Hou, T., *Coinduction for Exact Real Number Computation.* Theory Comput. Sys. **43** (2008) 394-409.

[7] Berger, U., and Seisenberger, M., *Proofs, programs, processes.* In (Ferreira, F., Löwe, B., Mayordomo, E., Gomes, L. M., eds.) Programs, Proofs, Processes, CiE 2010, Ponta Delgada, Azores, Portugal LNCS **6158** (2010) 39-48.

[8] Berger, U., *From coinductive proofs to exact real arithmetic: theory and applications.* Logical Methods in Comput. Sci. **7(1)** (2011) 1-24.

[9] Berger, U., and Seisenberger, M., *Program extraction via typed realisability for induction and coinduction.* In (Schindler, R., ed.) Ways of Proof Theory. Ontos Series in Mathematical Logic, Ontos Verlag, Frankfurt. (2011) 157-182.

[10] Longley, J., *Matching typed and untyped realizability.* Electronic Notes in Theoretical Computer Science **35** (2000) 109-132.

[11] Pitts, A. M., *Relational properties of recursively defined domains.* In Proceedings of the Eighth Annual IEEE Symp. on Logic in Computer Science, LICS **1993** (1993), M. Vardi, Ed., IEEE Computer Society Press, 86-97.

[12] Plotkin, G.D., "Lambda-definability and logical relations". Memorandum SAIRM-4, University of Edinburgh, October 1973.

[13] Reynolds, J.C., "The meaning of types - from intrinsic to extrinsic semantics". BRICS Report Series RS-00-32, University of Aarhus, December 2000.

[14] Reynolds, J.C., *What do types mean? - from intrinsic to extrinsic semantics*, Essays on Programming Methodology, Springer (2003) 309 - 327.

[15] Stoltenberg-Hansen, V., Lindström, I., and Griffor, E.R., "Mathematical Theory of Domains", vol. **22** of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1994.