

# Second-Order Pre-Logical Relations and Representation Independence

Hans Leiß

Centrum für Informations- und Sprachverarbeitung  
Universität München, Oettingenstr. 67, D-80538 München  
leiss@cis.uni-muenchen.de

**Abstract.** We extend the notion of pre-logical relation between models of simply typed lambda-calculus, recently introduced by F. Honsell and D. Sannella, to models of second-order lambda calculus. With pre-logical relations, we obtain characterizations of the lambda-definable elements of and the observational equivalence between second-order models. These are simpler than those using logical relations on extended models.

We also characterize representation independence for abstract data types and abstract data type constructors by the existence of a pre-logical relation between the representations, thereby varying and generalizing results of J.C. Mitchell to languages with higher-order constants.

## 1 Introduction

“Logical” relations between terms or elements of models of the simply typed  $\lambda$ -calculus allow one to prove, by induction on the structure of types, syntactical properties like normalization and Church-Rosser theorems ([Tai67], [Sta85]) and semantical properties like  $\lambda$ -definability of elements ([Pl80], [Sta85]) or observational equivalence of models ([Mit91]).

In first-order, i.e. simply typed,  $\lambda$ -calculus, a logical relation  $\{R_\tau \mid \tau \in \text{Type}\} = \mathcal{R}$  is built by induction on types using

$$(f, g) \in R_{(\rho \rightarrow \sigma)} \iff \forall a, b : \rho \ ((a, b) \in R_\rho \rightarrow (f \cdot a, g \cdot b) \in R_\sigma), \quad (1)$$

starting from relations  $R_\tau$  on base types  $\tau$ . In second-order  $\lambda$ -calculus, the domain of (semantic) types can no longer be constructed inductively; yet, central results of the first-order case can be transferred, such as the characterization of  $\lambda$ -definable elements or that of observational equivalence (cf. J. Mitchell [MM85],[Mit86]).

Even in the first-order case, logical relations have some disturbing properties, viz. they are not closed under relation composition and do not admit a characterization of observational equivalence for languages with higher-order constants. To overcome these difficulties, several authors (cf. [HS99], [PPST00], [PR00]) recently have proposed to use a more general class of relations, called *pre-* or *lax logical relations*. They arise by restricting direction  $\Leftarrow$  in (1) to functions that are

$\lambda$ -definable (using related parameters). F. Honsell and D. Sannella [HS99] gave several characterizations and applications of pre-logical relations that demonstrate the stability and usefulness of the notion.

We here follow this route and generalize pre-logical relations to the second-order  $\lambda$ -calculus by similarly weakening the conditions for  $R_{\forall\alpha\tau}$ . As in the first-order case, the  $\lambda$ -definable elements of a model can then be characterized as the intersection of all pre-logical predicates on the model. It follows that observational equivalence of second-order models can be expressed as a suitable pre-logical relation between models.

But our main interest lies on the equivalence of different implementations or representations of an abstract datatype  $\alpha$  and the operations  $c : \sigma(\alpha)$  coming with it. Two representations  $(\tau_i, t_i : \sigma[\tau_i/\alpha])$  of  $(\alpha, c : \sigma)$  in a model  $\mathcal{A}$  are equivalent if appropriately defined expansions  $\mathcal{A}(\tau_i, t_i)$  of  $\mathcal{A}$  are observationally equivalent with respect to the language not containing  $\alpha$  and  $c$ .

For languages without higher-order constants, J. C. Mitchell [Mit86, Mit91] has characterized this equivalence by the existence of a suitable logical relation between expansions of the models. However, programming languages like Standard ML [MTHM97] not only have abstract data types in the presence of higher-order constants, but they also allow to abstract from data type *constructors* (and, in some versions, higher-order functors). We handle abstract type constructors as indeterminates on the kind level and their implementations as definable expansions of a second-order model. Representation independence for abstract type constructors can then be characterized as the existence of a second-order pre-logical relation between the representations. This also holds for languages with higher-order constants.

In section 2 we consider declarations of abstract data types as expansions of first-order models by definable types and elements, and compare a characterization of equivalent representations based on pre-logical relations with a criterion by Mitchell based on logical relations. Section 3 gives syntax and semantics for second-order  $\lambda$ -calculus. In section 4 we define second-order pre-logical relations, present some basic properties in 4.1, and use them in 4.2 to characterize definable elements and observational equivalence for second order models. Section 4.3 treats abstract type constructors and polymorphic constants and characterizes the equivalence of two representations by definable type constructors.

## 2 Pre-Logical Relations for Simply Typed $\lambda$ -Calculus

Let  $T$  be the simple types  $\sigma, \tau ::= \beta \mid (\sigma \rightarrow \tau)$  over base types  $\beta$ . We write  $\Gamma \triangleright t : \tau$  for a term typed in the context  $\Gamma : \text{Var} \rightarrow T$  using the standard typing rules.  $\lambda_C^{\rightarrow}$  stands for the set of typed terms over a set  $C$  of typed constants,  $c : \sigma$ .

**Definition 2.1.** An (extensional) model  $\mathcal{A} = (A, \Phi^{\rightarrow}, \Psi^{\rightarrow}, C^{\mathcal{A}})$  of the simply typed  $\lambda$ -calculus  $\lambda_C^{\rightarrow}$  consists of a family  $A = \langle A_{\sigma} \rangle_{\sigma \in T}$  of sets, an interpretation  $C^{\mathcal{A}}(c) \in A_{\sigma}$  of the constants  $c : \sigma \in C$ , and families  $\langle \Phi_{\sigma, \tau}^{\rightarrow} \rangle_{\sigma, \tau \in T}$  and  $\langle \Psi_{\sigma, \tau}^{\rightarrow} \rangle_{\sigma, \tau \in T}$

of mappings

$$\Phi_{\sigma,\tau}^{\rightarrow} : A_{(\sigma \rightarrow \tau)} \rightarrow [A_{\sigma} \rightarrow A_{\tau}] \quad \text{and} \quad \Psi_{\sigma,\tau}^{\rightarrow} : [A_{\sigma} \rightarrow A_{\tau}] \rightarrow A_{(\sigma \rightarrow \tau)},$$

where  $[A_{\sigma} \rightarrow A_{\tau}] \subseteq \{h \mid h : A_{\sigma} \rightarrow A_{\tau}\}$  contains all definable functions, such that  $\Phi_{\sigma,\tau}^{\rightarrow} \circ \Psi_{\sigma,\tau}^{\rightarrow} = Id_{(A_{\sigma \rightarrow \tau})}$  (and  $\Psi_{\sigma,\tau}^{\rightarrow} \circ \Phi_{\sigma,\tau}^{\rightarrow} = Id_{A_{(\sigma \rightarrow \tau)}}$ ) for all  $\sigma, \tau \in T$ .

An environment  $\eta$  over  $\mathcal{A}$  satisfies the context  $\Gamma$ , in symbols:  $\eta : \Gamma \rightarrow \mathcal{A}$ , if  $\eta(x) \in A_{\sigma}$  for all  $x : \sigma$  in  $\Gamma$ . Terms are interpreted in  $\mathcal{A}$  under  $\eta : \Gamma \rightarrow \mathcal{A}$  via

$$\begin{aligned} \llbracket \Gamma \triangleright (r \cdot s) \rrbracket \eta &:= \Phi_{\sigma,\tau}(\llbracket \Gamma \triangleright r : (\sigma \rightarrow \tau) \rrbracket \eta)(\llbracket \Gamma \triangleright s : \sigma \rrbracket \eta), \\ \llbracket \Gamma \triangleright \lambda x t : (\sigma \rightarrow \tau) \rrbracket \eta &:= \Psi_{\sigma,\tau}(\lambda a \in A_{\sigma}. \llbracket \Gamma, x : \sigma \triangleright t : \tau \rrbracket \eta[a/x]). \end{aligned}$$

We write  $\llbracket t \rrbracket \eta$  rather than  $\llbracket \Gamma \triangleright t : \tau \rrbracket \eta$ , if  $\Gamma$  and  $\tau$  are clear from the context.

**Definition 2.2.** A predicate  $\mathcal{R} \subseteq \mathcal{A}$  on  $\mathcal{A}$  is a family  $\mathcal{R} = \{R_{\sigma} \mid \sigma \in T\}$  with  $R_{\sigma} \subseteq A_{\sigma}$  for each  $\sigma \in T$ . An environment  $\eta : \Gamma \rightarrow \mathcal{A}$  over  $\mathcal{A}$  respects  $\mathcal{R}$ , in symbols:  $\eta : \Gamma \rightarrow \mathcal{R}$ , if  $\eta(x) \in R_{\sigma}$  for each  $x : \sigma \in \Gamma$ .

A predicate  $\mathcal{R} \subseteq \mathcal{A}$  is called pre-logical, if  $\llbracket \Gamma \triangleright t : \tau \rrbracket \eta \in R_{\tau}$  for each term  $\Gamma \triangleright t : \tau$  and environment  $\eta : \Gamma \rightarrow \mathcal{R}$ . A pre-logical relation between  $\mathcal{A}$  and  $\mathcal{B}$  is a pre-logical predicate on  $\mathcal{A} \times \mathcal{B}$ .

The definition is easily adapted to extensions of  $T$  by cartesian products ( $\sigma \times \tau$ ) and disjoint unions ( $\sigma + \tau$ ), if the extension of terms is done by adding constants. The property used here as a definition is called the ‘‘Basic Lemma’’ for pre-logical relations by Honsell and Sannella. The Basic Lemma for logical relations says that every logical relation is a pre-logical relation.

Let  $Def^{\mathcal{A}} = \{Def_{\tau}^{\mathcal{A}} \mid \tau \in T\}$  be the predicate of definable elements of  $\mathcal{A}$ , where

$$Def_{\tau}^{\mathcal{A}} := \{\llbracket t \rrbracket^{\mathcal{A}} \mid \triangleright t : \tau, t \in \lambda_C^{\rightarrow}\}.$$

Normally,  $Def^{\mathcal{A}}$  is not a logical predicate since  $[Def_{\sigma} \rightarrow Def_{\tau}] \not\subseteq Def_{\sigma \rightarrow \tau}$ . But:

**Theorem 2.3** ([HS99], Proposition 5.7, Example 3.10) Let  $\mathcal{A}$  be a model of  $\lambda_C^{\rightarrow}$ . (i) For each predicate  $\mathcal{C} \subseteq \mathcal{A}$ , there is a least pre-logical predicate  $\mathcal{R} \subseteq \mathcal{A}$  with  $\mathcal{C} \subseteq \mathcal{R}$ . (ii)  $Def^{\mathcal{A}}$  is the least pre-logical predicate on  $\mathcal{A}$ .

We use the following version of observational equivalence (it differs slightly from indistinguishability by closed equations of observable types as used in [HS99]):

**Definition 2.4.** Let  $T^+$  be the simple types generated from a superset of the base types of  $T$ , and  $C^+$  an extension of  $C$  by new constants with types in  $T^+$ . Let  $\mathcal{A}^+, \mathcal{B}^+$  be two models of  $\lambda_{C^+}^{\rightarrow}$  with  $A_{\tau}^+ = B_{\tau}^+$  for all  $\tau \in T$ .  $\mathcal{A}^+$  and  $\mathcal{B}^+$  are observationally equivalent with respect to  $T$ , in symbols:  $\mathcal{A}^+ \equiv_T \mathcal{B}^+$ , if  $\llbracket t \rrbracket^{\mathcal{A}^+} = \llbracket t \rrbracket^{\mathcal{B}^+}$  for all  $\triangleright t : \tau$  of  $\lambda_{C^+}^{\rightarrow}$  where  $\tau \in T$ .

**Theorem 2.5** (cf. [HS99], Theorem 8.4)  $\mathcal{A}^+ \equiv_T \mathcal{B}^+$  iff there is a pre-logical relation  $\mathcal{R}^+ \subseteq \mathcal{A}^+ \times \mathcal{B}^+$  such that  $R_{\tau}^+ \cap (Def_{\tau}^{\mathcal{A}^+} \times Def_{\tau}^{\mathcal{B}^+}) \subseteq Id_{\tau}$  for all  $\tau \in T$ .

For  $\Rightarrow$  use 2.3 and  $\mathcal{R}^+ = Def^{\mathcal{A}^+ \times \mathcal{B}^+}$ . With logical relations we have  $\Rightarrow$  only if  $C$  does not contain higher-order constants. (cf. Exercise 8.5.6 in [Mit96].)

## 2.1 Abstract data types and representation independence

The declaration of an abstract data type,

$$\mathbf{(abstype\ } (\alpha, x : \sigma) \mathbf{ with\ } x : \sigma \triangleright e_1 = e_2 : \rho \mathbf{ is\ } (\tau, t : \sigma[\tau/\alpha]) \mathbf{ in\ } s), \quad (2)$$

introduces a “new” type  $\alpha$  and a “new” object  $x : \sigma(\alpha)$  with “defining” property  $e_1 = e_2 : \rho(\alpha)$ , which are interpreted in scope  $s$  by the type  $\tau$  and the term  $t$ , where  $\alpha$  does not occur in the type of  $s$ . Roughly, evaluation of (2) in a model  $\mathcal{A}$  of  $\lambda_{\vec{C}}$  is done by first expanding  $\mathcal{A}$  by interpreting  $\alpha$  by  $A_\tau$  and  $x$  by the value of  $t : \sigma[\tau/\alpha]$  and then evaluating  $s$  in the expanded model,  $\mathcal{A}(\tau, t)$  (see below).

The data type  $(\alpha, x : \sigma)$  is *abstract* in  $s$  if the value of  $s$  is independent from the representation  $(\tau, t)$  used; i.e. if for any two representations  $(\tau_i, t_i : \sigma[\tau_i/\alpha])$ ,  $i = 1, 2$ , that “satisfy” the equation  $x : \sigma \triangleright e_1 = e_2 : \rho$ , we have

$$\llbracket s \rrbracket^{\mathcal{A}(\tau_1, t_1)} = \llbracket s \rrbracket^{\mathcal{A}(\tau_2, t_2)}. \quad (3)$$

To “satisfy” the equation  $e_1 = e_2$  in an expansion  $\mathcal{A}(\tau_i, t_i)$ , the equality on the new type  $\alpha$  normally is not true equality on  $A_{\tau_i}$ , but a suitable partial equivalence relation on  $A_{\tau_i}$ :

$$\begin{aligned} & \llbracket (\mathbf{abstype\ } (\alpha, x_0 : \sigma_0) \mathbf{ with\ } x_0 : \sigma_0 \triangleright e_1 = e_2 : \rho \mathbf{ is\ } (\tau_0, t : \sigma_0[\tau_0/\alpha]) \mathbf{ in\ } s) \rrbracket^{\mathcal{A}} \eta \\ := & \begin{cases} (\text{let } \mathcal{A}^+ = \mathcal{A}(\tau_0, \llbracket t \rrbracket^{\mathcal{A}} \eta) \text{ in } \llbracket s \rrbracket^{\mathcal{A}^+} \eta), & \text{if there is a pre-logical} \\ & \text{partial equivalence relation } \mathcal{E} \subseteq \mathcal{A}^+ \times \mathcal{A}^+ \text{ such that} \\ & (\llbracket e_1 \rrbracket \eta^{\mathcal{A}^+}, \llbracket e_2 \rrbracket \eta^{\mathcal{A}^+}) \in E_\rho \text{ and } E_\tau = Id_{A_\tau} \text{ for } \tau \in T, \\ \text{error,} & \text{else.} \end{cases} \end{aligned}$$

Note that in the first case, there is a least such  $\mathcal{E}$ , and  $\mathcal{A}$  is canonically embedded in  $\mathcal{A}^+/\mathcal{E}$ . Since  $\mathcal{E}$  is pre-logical, the value of  $x_0$  in  $\mathcal{A}^+$  is self-related by  $E_{\sigma_0}$ , and the additional operations provided by  $t$  actually belong to  $\mathcal{A}^+/\mathcal{E}$ , which satisfies  $e_1 = e_2$ . But since the type of  $s$  does not involve  $\alpha$ , the value of  $s$  in  $\mathcal{A}^+$  and  $\mathcal{A}^+/\mathcal{E}$  is the same, so there is no need to actually construct  $\mathcal{A}^+/\mathcal{E}$ .

**Definition 2.6.** Let  $T^+$  be the types constructed from the base types of  $T$  and a new base type  $\alpha$ . Let  $\sigma_0 \in T^+$  and  $C^+ = C, x_0 : \sigma_0$ . For  $\tau_0 \in T$ ,  $a \in A_{\sigma_0[\tau_0/\alpha]}$  let  $\mathcal{A}(\tau_0, a) = \mathcal{A}^+ := (A^+, \Phi^+, \Psi^+, (C^+)^{\mathcal{A}^+})$  be given by:

$$\begin{aligned} A_\sigma^+ & := A_{\sigma[\tau_0/\alpha]}, & (\Phi^+)_{\sigma, \tau}^{\rightarrow} & := \Phi_{\sigma[\tau_0/\alpha], \tau[\tau_0/\alpha]}^{\rightarrow}, \\ (C^+)^{\mathcal{A}^+}(c) & = \begin{cases} C^{\mathcal{A}}(c), & c : \tau \in C, \\ a, & c : \tau \equiv x_0 : \sigma_0, \end{cases} & (\Psi^+)_{\sigma, \tau}^{\rightarrow} & := \Psi_{\sigma[\tau_0/\alpha], \tau[\tau_0/\alpha]}^{\rightarrow}. \end{aligned}$$

We call the model  $\mathcal{A}^+$  of  $\lambda_{\vec{C}^+}$  the expansion of  $\mathcal{A}$  defined by  $(\tau_0, a)$ .

From now on we focus on the equivalence of representations and ignore the restriction to implementations satisfying a specification. Two expansions  $\mathcal{A}(\tau_1, t_1)$  and  $\mathcal{A}(\tau_2, t_2)$  of  $\mathcal{A}$  are equivalent representations of an abstype  $(\alpha, x : \sigma)$  if (3) holds for all terms  $s$  whose type does not contain  $\alpha$ , i.e. if  $\mathcal{A}(\tau_1, t_1) \equiv_T \mathcal{A}(\tau_2, t_2)$ .

*Remark 2.7.* For a careful categorical treatment of abstract data types with equational specifications, see [PR00]. The simpler form of abstype declarations (**abstype**  $(\alpha, x : \sigma)$  **is**  $(\tau, t)$  **in**  $s$ ) does occur in programming languages like SML and has been investigated in [MP85,Mit91]. It can be viewed as a case of SML's restriction  $Str \rightarrow Sig$  of the *structure*  $(\alpha = \tau, x = t)$  to the *signature*  $(\alpha, x : \sigma)$ .

**Example 2.8** Consider a type of multisets with some higher-order operations. (Assume that  $'a = ''b$  is a fixed type of  $T$ . Actually,  $bag : T \Rightarrow T$  is a type constructor, since  $'a$  and  $''b$  are type variables. See section 4.2 for this.)

```
signature BAG = sig
  type 'a bag
  val empty : 'a bag
  val member : ''a -> 'a bag -> int
  val insert : ''a * int -> 'a bag -> 'a bag
  val map : ('a -> ''b) -> 'a bag -> ''b bag
  val union : ('a -> ''b bag) -> 'a bag -> ''b bag
end
```

An  $A$ -bag  $B$  represents the multiset  $\{(a, n) \mid \text{member } a \ B = n > 0, a \in A\}$ . Each implementation of `member` gives a partial equivalence relation  $\mathcal{E}$  such that  $B_1 \mathcal{E}_{\text{BAG}} B_2$  iff  $B_1$  and  $B_2$  represent the same multiset. One implementation `Bag1` of bags is by lists of elements paired with their multiplicity:

```
structure Bag1 :> BAG = struct
  type 'a bag = ('a * int) list
  fun member a [] = 0
    | member a ((b,n)::B) = if a=b then n else (member a B)
  ...
end
```

Another implementation `Bag2` is by lists of elements:

```
structure Bag2 :> BAG = struct
  type 'a bag = 'a list
  fun member a [] = 0
    | member a (b::B) = if a=b then 1+(member a B) else (member a B)
  ...
end
```

The implementations `Bag1` and `Bag2` are observationally equivalent if they assign the same value to the closed terms  $(\text{member } s \ t) : \text{int}$ .

Observational equivalence can be characterized by pre-logical relations. To cover nested abstype-declarations, we characterize the observational equivalence of two expansions of models that are related by a pre-logical relation  $\mathcal{R}$  instead of  $Id$ .

**Theorem 2.9** Let  $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$  be a pre-logical relation. Let  $\mathcal{A}^+$  and  $\mathcal{B}^+$  be definable expansions of  $\mathcal{A}$  and  $\mathcal{B}$  to models of  $\lambda_{C^+}^{\rightarrow}$ . The following are equivalent:

- (i) for each closed  $\lambda_{C^+}^{\rightarrow}$ -term  $\triangleright t : \tau$  with  $\tau \in T$  we have  $\llbracket t \rrbracket^{\mathcal{A}^+} R_{\tau} \llbracket t \rrbracket^{\mathcal{B}^+}$ .
- (ii) there is a pre-logical relation  $\mathcal{R}^+$  with  $\mathcal{A}^+ \mathcal{R}^+ \mathcal{B}^+$  and  $R_{\tau}^+ = R_{\tau}$  for all  $\tau \in T$ .

*Proof.* For simplicity of notation we consider the unary case. i.e. pre-logical predicates  $\mathcal{R} \subseteq \mathcal{A}$  and  $\mathcal{R}^+ \subseteq \mathcal{A}^+$ . (ii)  $\Rightarrow$  (i) is obvious. (i)  $\Rightarrow$  (ii): for  $\tau \in T^+$  let

$$Def_\tau^+ = \{\llbracket t \rrbracket^{\mathcal{A}^+} \mid t \in \lambda_{C^+}^{\rightarrow}, \vdash t : \tau\} \quad \text{and} \quad C_\tau := \begin{cases} R_\tau, & \tau \in T, \\ Def_\tau^+, & \text{else.} \end{cases}$$

By assumption we have  $Def_\tau^+ \subseteq R_\tau$  for  $\tau \in T$ . Each logical predicate  $\mathcal{R}^+$  on  $\mathcal{A}^+$  as in (ii) satisfies the following conditions in positively occurring unknowns  $X_\tau$ :

$$X_\tau \supseteq C_\tau \cup \bigcup \{X_{(\rho \rightarrow \tau)} \cdot X_\rho \mid \rho \in T^+\}, \quad \tau \in T^+. \quad (4)$$

Let  $\mathcal{R}^+ = \{R_\tau^+ \mid \tau \in T^+\} \subseteq \mathcal{A}^+$  be the least solution of (4). It is a pre-logical predicate: if  $\eta : \Gamma \rightarrow \mathcal{R}^+$  and  $\Gamma \triangleright t : \tau$  is a  $\lambda_{C^+}^{\rightarrow}$ -term, with  $\Gamma = \{x : \sigma\}$  say, then

$$\llbracket \Gamma \triangleright t : \tau \rrbracket \eta = \llbracket \triangleright \lambda x : \sigma. t : \sigma \rightarrow \tau \rrbracket \cdot \eta(x) \in Def_{\sigma \rightarrow \tau}^+ \cdot R_\sigma^+ \subseteq R_{\sigma \rightarrow \tau}^+ \cdot R_\sigma^+ \subseteq R_\tau^+.$$

It is not hard to see that for each  $\tau \in T^+$

$$R_\tau^+ = \bigcup \{Def_{\rho_1 \rightarrow \dots \rightarrow \rho_n \rightarrow \tau}^+ \cdot R_{\rho_1} \cdots R_{\rho_n} \mid n \in \mathbb{N}, \rho_1, \dots, \rho_n \in T\}.$$

For  $\rho, \tau \in T$  we have  $Def_{\rho \rightarrow \tau}^+ \subseteq R_{\rho \rightarrow \tau}$  by (i), which gives  $Def_{\rho \rightarrow \tau}^+ \cdot R_\rho \subseteq R_\tau$  and so  $R_\tau^+ \subseteq R_\tau$ . Therefore we also have  $R_\tau^+ = R_\tau$  for  $\tau \in T$ .  $\square$

In the corresponding theorem for logical relations (cf. [Mit91], Cor.1), direction (i)  $\Rightarrow$  (ii) is restricted to languages where  $C$  has no higher-order constants. The logical relation  $\mathcal{R}^+$  in (ii) can then be inductively generated from the predicates  $\{R_\tau^+ \mid \tau \in T^+ \text{ a base type}\}$  of the least solution of (4). This works even when all constants  $c : \tau \in C^+$  are first-order *in*  $\alpha$ , like `map` in 2.8. The observational equivalence of two representations can be shown by (ii)  $\Rightarrow$  (i). When used with *logical* relations, premise (ii) can be reformulated to a ‘local’ criterion for equivalence of representations, which fails for *pre-logical* relations:

**Lemma 2.10.** (cf. [Mit86], Lemma 4) *Let  $\mathcal{A}$  and  $\mathcal{B}$  be logically related by  $\mathcal{R}$ , and let  $\mathcal{A}^+$  and  $\mathcal{B}^+$  be definable expansions of  $\mathcal{A}$  and  $\mathcal{B}$  to models of  $\lambda_{C^+}^{\rightarrow}$ . Then (ii) and (i) are equivalent:*

- (i) *There is a relation  $R_\alpha^+ \subseteq A_\alpha^+ \times B_\alpha^+$ , such that  $c^{\mathcal{A}^+} R_\tau^+ c^{\mathcal{B}^+}$  for  $c : \tau \in C^+ \setminus C$ , where  $R_\tau^+ := R_\tau$  for  $\tau \in T$  and  $R_{(\rho \rightarrow \sigma)}^+ = [R_\rho^+ \rightarrow R_\sigma^+]$  for  $(\rho \rightarrow \sigma) \notin T$ .*
- (ii) *There is a logical relation  $\mathcal{R}^+ \subseteq \mathcal{A}^+ \times \mathcal{B}^+$  with  $R_\tau^+ = R_\tau$  for all  $\tau \in T$ .*

Unfortunately, if  $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$  is pre-logical, then condition (i) does *not* imply the existence of a pre-logical relation  $\mathcal{R}^+ \subseteq \mathcal{A}^+ \times \mathcal{B}^+$  that agrees with  $\mathcal{R}$  on types of  $T$ , because the necessary condition (cf. Theorem 2.9, (i))  $Def_{(\sigma \rightarrow \tau)}^+ \subseteq R_{(\sigma \rightarrow \tau)}$  for  $(\sigma \rightarrow \tau) \in T$  may fail: Take  $f \in [R_\sigma \rightarrow R_\tau] \setminus R_{(\sigma \rightarrow \tau)}$  and let  $f = c_2 \circ c_1$  with constants  $c_1 : \sigma \rightarrow \rho, c_2 : \rho \rightarrow \tau$  for some  $\rho \notin T$ .

Hence, although pre-logical relations give a complete *characterization* of the equivalence of representations when higher-order constants are present, they seem harder to use in *establishing* the equivalence of representations: instead of checking the existence of  $R_\alpha^+$  with property (i) of 2.10, one has to check the global property of  $\lambda_{C^+}^{\rightarrow}$ -observational equivalence with respect to all types of  $T$ .

### 3 Second Order Lambda Calculus, $\lambda_C^{\rightarrow, \forall}$

We now look at definability and observational equivalence for the second order  $\lambda$ -calculus  $\lambda_C^{\rightarrow, \forall}$ . The characterizations by pre-logical relations are similar to the first-order case, but the construction of pre-logical relations is more elaborate since these can no longer be defined by induction on type expressions.

#### 3.1 Syntax and semantics of $\lambda_C^{\rightarrow, \forall}$

In second order lambda calculus, in addition to terms denoting objects one has constructors denoting types or functions on types.

**Definition 3.1.** *The kinds  $\kappa$  of  $\lambda_C^{\rightarrow, \forall}$  are given by  $\kappa := T \mid (\kappa \Rightarrow \kappa)$ . The classification  $C$  of constants is split in two components,  $C = (C_{Kind}, C_{Type})$ . By  $C_{Kind}$ , a set of assumptions of the form  $c : \kappa$ , each constructor constant  $c$  is assigned a unique kind  $\kappa$ . In particular,  $C_{Kind}$  contains the type constructors  $\rightarrow : T \Rightarrow (T \Rightarrow T)$  and  $\forall : (T \Rightarrow T) \Rightarrow T$ .*

*A constructor  $\Delta \triangleright \mu : \kappa$  of kind  $\kappa$  is a sequent derivable with the following rules of  $\lambda_{C_{Kind}}^{\rightarrow}$ , where  $\Delta$  is a context of kind assumptions for constructor variables:*

$$(Mon) \frac{\Delta \triangleright \mu : \kappa}{\Delta, v : \kappa' \triangleright \mu : \kappa}, \quad v \notin \text{dom}(\Delta)$$

$$(Const) \Delta \triangleright c : \kappa, \quad \text{for } c : \kappa \in C_{Kind}$$

$$(Var) \Delta, v : \kappa \triangleright v : \kappa$$

$$(\Rightarrow E) \frac{\Delta \triangleright \mu : (\kappa_1 \Rightarrow \kappa_2), \quad \Delta \triangleright \nu : \kappa_1}{\Delta \triangleright (\mu \cdot \nu) : \kappa_2}$$

$$(\Rightarrow I) \frac{\Delta, v : \kappa_1 \triangleright \mu : \kappa_2}{\Delta \triangleright \lambda v. \mu : (\kappa_1 \Rightarrow \kappa_2)}.$$

*For  $\lambda_{C_{Kind}}^{\rightarrow}$  we assume a reduction relation  $\sim$  subsuming  $\alpha, \beta, \eta$ -reduction and satisfying the subject reduction, i.e. if  $\Delta \triangleright \mu : \kappa$  and  $\mu \sim \nu$ , then  $\Delta \triangleright \nu : \kappa$ .*

$$(\Rightarrow_1) \frac{\Delta \triangleright \mu : \kappa \quad \mu \sim \nu}{\Delta \triangleright \mu = \nu : \kappa}$$

$$(\Rightarrow_2) \frac{\Delta \triangleright \mu : \kappa \quad \mu \sim \nu}{\Delta \triangleright \nu = \mu : \kappa}$$

*A type  $\Delta \triangleright \mu : T$  is constructor of kind  $T$ . We use  $\tau, \sigma$  for types and write  $(\sigma \rightarrow \tau)$  and  $\forall v \tau$  etc. By  $C_{Type}$ , a set of assumptions of the form  $c : \mu$ , each individual constant  $c$  is assigned a unique closed constructor  $\mu$  of kind  $T$ .*

*A typed term  $\Delta; \Gamma \triangleright t : \tau$  is a sequent derivable with the following rules, where  $\Delta$  is a context of kind assumptions for constructor variables and  $\Gamma$  is a context of type assumptions for individual variables:*

$$(mon) \frac{\Delta; \Gamma \triangleright t : \tau \quad \Delta \triangleright \sigma : T}{\Delta; \Gamma, x : \sigma \triangleright t : \tau}, \quad x \notin \text{dom}(\Gamma)$$

$$(const) \frac{\triangleright \tau : T}{\Delta; \Gamma \triangleright c : \tau}, \quad \text{for } c : \tau \in C_{Type}$$

$$(var) \frac{\Delta \triangleright \tau : T}{\Delta; \Gamma, x : \tau \triangleright x : \tau}, \quad x \notin \text{dom}(\Gamma)$$

$$\begin{aligned}
(\rightarrow I) & \frac{\Delta \triangleright \sigma : T \quad \Delta; \Gamma, x : \sigma \triangleright t : \tau}{\Delta; \Gamma \triangleright (\lambda x : \sigma. t) : (\sigma \rightarrow \tau)} \\
(\rightarrow E) & \frac{\Delta \triangleright \tau : T \quad \Delta; \Gamma \triangleright t : (\sigma \rightarrow \tau) \quad \Delta; \Gamma \triangleright s : \sigma}{\Delta; \Gamma \triangleright (t \cdot s) : \tau} \\
(\forall I) & \frac{\Delta \triangleright \mu : (T \Rightarrow T) \quad \Delta, \alpha : T; \Gamma \triangleright t : (\mu \cdot \alpha)}{\Delta; \Gamma \triangleright \lambda \alpha t : \forall \mu}, \quad \text{if } \alpha \text{ is not in } \Gamma \\
(\forall E) & \frac{\Delta \triangleright \mu : (T \Rightarrow T) \quad \Delta \triangleright \tau : T \quad \Delta; \Gamma \triangleright t : \forall \mu}{\Delta; \Gamma \triangleright (t \cdot \tau) : (\mu \cdot \tau)} \\
(\text{Typ} =) & \frac{\Delta; \Gamma \triangleright t : \tau \quad \Delta \triangleright \tau = \sigma : T}{\Delta; \Gamma \triangleright t : \sigma}
\end{aligned}$$

By induction on the derivation, using the subject reduction property of  $\lambda_{C_{Kind}}^{\rightarrow, \forall}$  in the case of (Typ =), we obtain:

**Proposition 3.2.** *If  $\Delta; \Gamma \triangleright t : \sigma$  is a term, then  $\Delta \triangleright \sigma : T$  is a type.*

**Definition 3.3.** *A model frame  $\mathcal{M} = (\mathcal{U}, \mathcal{D}, \Phi, \Psi)$  for  $\lambda_C^{\rightarrow, \forall}$  consists of*

1. *an extensional model  $\mathcal{U} = (U, \Phi^{\rightarrow}, \Psi^{\rightarrow}, C_{Kind}^{\mathcal{U}}, \llbracket \cdot \rrbracket^{\mathcal{U}})$  of  $\lambda_{C_{Kind}}^{\rightarrow}$  to interpret the constructors,*
2. *a family of individual domains for the types,  $\mathcal{D} = (\langle D_A \rangle_{A \in U_T}, C_{Type}^{\mathcal{D}})$ , with individuals  $C_{Type}^{\mathcal{D}}(c) \in D_{\llbracket \tau \rrbracket^{\mathcal{U}}}$  for  $c : \tau \in C_{Type}$ ,*
3. *families  $\Phi = (\Phi^{\rightarrow}, \Phi^{\forall})$ ,  $\Psi = (\Psi^{\rightarrow}, \Psi^{\forall})$  with  $\Phi^{\rightarrow} = \langle \Phi_{A,B}^{\rightarrow} \rangle_{A,B \in U_T}$ ,  $\Phi^{\forall} = \langle \Phi_f^{\forall} \rangle_{f \in U_{(T \Rightarrow T)}}$ ,  $\Psi^{\rightarrow} = \langle \Psi_{A,B}^{\rightarrow} \rangle_{A,B \in U_T}$ ,  $\Psi^{\forall} = \langle \Psi_f^{\forall} \rangle_{f \in U_{(T \Rightarrow T)}}$  of mappings*

$$\begin{aligned}
\Phi_{A,B}^{\rightarrow} : D_{(A \rightarrow B)} & \longrightarrow [D_A \rightarrow D_B] & \Psi_{A,B}^{\rightarrow} : [D_A \rightarrow D_B] & \longrightarrow D_{(A \rightarrow B)} \\
\Phi_f^{\forall} : D_{\forall f} & \longrightarrow [\Pi A \in U_T. D_{f.A}] & \Psi_f^{\forall} : [\Pi A \in U_T. D_{f.A}] & \longrightarrow D_{\forall f},
\end{aligned}$$

for subsets  $[D_A \rightarrow D_B] \subseteq \{h \mid h : D_A \rightarrow D_B\}$  and  $[\Pi A \in U_T. D_{f.A}] \subseteq \Pi A \in U_T. D_{f.A}$ , such that for all  $A, B \in U_T$  and  $f \in U_{(T \Rightarrow T)}$

$$\Phi_{A,B}^{\rightarrow} \circ \Psi_{A,B}^{\rightarrow} = Id_{[D_A \rightarrow D_B]} \quad \text{and} \quad \Phi_f^{\forall} \circ \Psi_f^{\forall} = Id_{[\Pi A \in U_T. D_{f.A}]}.$$

A (environment) model of  $\lambda_C^{\rightarrow, \forall}$  consist of a model frame  $\mathcal{M} = (\mathcal{U}, \mathcal{D}, \Phi, \Psi)$  and an evaluation  $\llbracket \cdot \rrbracket^{\mathcal{D}}$  of terms, such that every typed term gets a value using

$$\begin{aligned}
\llbracket \Delta; \Gamma \triangleright x : \tau \rrbracket \eta &= \eta(x), & \llbracket \Delta; \Gamma \triangleright c : \tau \rrbracket \eta &= C_{Type}^{\mathcal{D}}(c), \quad \text{for } c : \tau \in C_{Type} \\
\llbracket \Delta; \Gamma \triangleright (t \cdot s) : \tau \rrbracket \eta &= \Phi_{\llbracket \Delta \triangleright \sigma \rrbracket \eta, \llbracket \Delta \triangleright \tau \rrbracket \eta}^{\rightarrow}(\llbracket \Delta; \Gamma \triangleright t : (\sigma \rightarrow \tau) \rrbracket \eta)(\llbracket \Delta; \Gamma \triangleright s : \sigma \rrbracket \eta) \\
\llbracket \Delta; \Gamma \triangleright \lambda x : \sigma. t : (\sigma \rightarrow \tau) \rrbracket \eta &= \\
& \Psi_{\llbracket \Delta \triangleright \sigma \rrbracket \eta, \llbracket \Delta \triangleright \tau \rrbracket \eta}^{\rightarrow}(\lambda a \in D_{\llbracket \Delta \triangleright \sigma : T \rrbracket \eta}. \llbracket \Delta; \Gamma \triangleright t : \tau \rrbracket \eta[a/x]) \\
\llbracket \Delta; \Gamma \triangleright (t \cdot \tau) : (\mu \cdot \tau) \rrbracket \eta &= \Psi_{\llbracket \Delta \triangleright \mu : (T \Rightarrow T) \rrbracket \eta}^{\forall}(\llbracket \Delta; \Gamma \triangleright t : \forall \mu \rrbracket \eta)(\llbracket \Delta \triangleright \tau : T \rrbracket \eta) \\
\llbracket \Delta; \Gamma \triangleright \lambda \alpha t : \forall \mu \rrbracket \eta &= \Psi_{\llbracket \Delta \triangleright \mu : (T \Rightarrow T) \rrbracket \eta}^{\forall}(\lambda A \in U_T. \llbracket \Delta, \alpha : T; \Gamma \triangleright t : (\mu \cdot \alpha) \rrbracket \eta[A/\alpha])
\end{aligned}$$

In particular,  $\llbracket \Delta; \Gamma \triangleright t : \tau \rrbracket \eta \in D_{\llbracket \Delta \triangleright \tau : T \rrbracket \eta}$ . If it is clear which context and type is meant, we write  $\llbracket t \rrbracket \eta$  instead of  $\llbracket \Delta; \Gamma \triangleright t : \tau \rrbracket \eta$ . We write  $c^{\mathcal{D}}$  for  $C_{Type}^{\mathcal{D}}(c)$ .

We refer to Bruce e.a. [BMM90] for an overview of various models of  $\lambda_C^{\rightarrow, \forall}$ .

## 4 Pre-Logical Relations for Second Order $\lambda$ -Calculus

**Definition 4.1.** Let  $\mathcal{A} = (\mathcal{U}, \mathcal{D}, \Phi, \Psi, \llbracket \cdot \rrbracket^{\mathcal{D}})$  be an environment model of  $\lambda_C^{\rightarrow, \forall}$ . A predicate  $\mathcal{R} = (\mathcal{R}_{Kind}, \mathcal{R}_{Type})$  on  $\mathcal{A}$  consists of a predicate  $\mathcal{R}_{Kind} = \{R_\kappa \mid \kappa \in Kind\}$  on  $\mathcal{U}$  where  $R_\kappa \subseteq U_\kappa$  for each  $\kappa \in Kind$  and a predicate  $\mathcal{R}_{Type} = \{R_A \mid A \in R_T\}$  on  $\mathcal{D}$  where  $R_A \subseteq D_A$  for each  $A \in R_T$ .

An environment  $\eta : \Delta; \Gamma \rightarrow \mathcal{A}$  respects the predicate  $\mathcal{R}$ , in symbols:  $\eta : \Delta; \Gamma \rightarrow \mathcal{R}$ , if  $\eta(v) \in R_\kappa$  for each  $v : \kappa \in \Delta$  and  $\eta(x) \in R_{\llbracket \Delta \triangleright \tau : T \rrbracket \eta}$  for each  $x : \tau \in \Gamma$ .

The predicate  $\mathcal{R} \subseteq \mathcal{A}$  is algebraic, if  $\mathcal{R}_{Kind}$  is algebraic (Def. 3.2 of [HS99]) and

- a)  $c^{\mathcal{D}} \in R_A$  for each  $c : \tau \in C_{Type}$  where  $A = \llbracket \triangleright \tau : T \rrbracket^{\mathcal{U}}$ ,
- b)  $R_{(A \rightarrow B)} \subseteq \{h \in D_{(A \rightarrow B)} \mid h \cdot R_A \subseteq R_B\}$  for all  $A, B \in R_T$ ,
- c)  $R_{\forall f} \subseteq \{h \in D_{\forall f} \mid \forall A \in R_T \ h \cdot A \in R_{f \cdot A}\}$  for all  $f \in R_{(T \rightarrow T)}$ .

An algebraic predicate  $\mathcal{R}$  is logical, if in b) and c) we also have  $\supseteq$ .

**Definition 4.2.** A predicate  $\mathcal{R} = (\mathcal{R}_{Kind}, \mathcal{R}_{Type})$  on  $\mathcal{A}$  is pre-logical, if

- (i)  $\mathcal{R}_{Kind}$  is a pre-logical predicate on  $\mathcal{U}$ , and
- (ii)  $\mathcal{R}_{Type}$  is a ‘pre-logical predicate on  $\mathcal{D}$ ’, i.e.  $\llbracket \Delta; \Gamma \triangleright t : \tau \rrbracket \eta \in R_{\llbracket \Delta \triangleright \tau : T \rrbracket \eta}$  for each term  $\Delta; \Gamma \triangleright t : \tau$  and every environment  $\eta : \Delta; \Gamma \rightarrow \mathcal{R}$ .

An algebraic (logical, pre-logical) relation  $\mathcal{R}$  between  $\mathcal{A}_1, \dots, \mathcal{A}_n$  is an algebraic (logical, pre-logical) predicate  $\mathcal{R} \subseteq \mathcal{A}_1 \times \dots \times \mathcal{A}_n$ .

### 4.1 Basic properties of second-order pre-logical relations

The Basic Lemma for second order logical relations just says they are pre-logical:

**Theorem 4.3** ([MM85], Theorem 2) Let  $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$  be a logical relation between environment models  $\mathcal{A}$  and  $\mathcal{B}$  of  $\lambda_C^{\rightarrow, \forall}$ . For each term  $\Delta; \Gamma \triangleright t : \tau$  and every environment  $\eta : \Delta; \Gamma \rightarrow \mathcal{R}$ ,  $\llbracket \Delta \triangleright \tau : T \rrbracket \eta \in R_T$  and  $\llbracket \Delta; \Gamma \triangleright t : \tau \rrbracket \eta \in R_{\llbracket \Delta \triangleright \tau : T \rrbracket \eta}$ .

As in the first-order case one can view definition 4.2 as a Basic Lemma for relations defined using the following properties:

**Lemma 4.4.** A predicate  $\mathcal{R} \subseteq \mathcal{A}$  is pre-logical iff:

1.  $\mathcal{R}_{Kind}$  is pre-logical and  $\mathcal{R}$  is algebraic,
2. for all terms  $\Delta; \Gamma, x : \sigma \triangleright t : \tau$  and environments  $\eta : \Delta; \Gamma \rightarrow \mathcal{R}$  such that

$$\forall a \in R_{\llbracket \sigma \rrbracket \eta} \llbracket \Delta; \Gamma, x : \sigma \triangleright t : \tau \rrbracket \eta[a/x] \in R_{\llbracket \tau \rrbracket \eta},$$

we have  $\llbracket \Delta; \Gamma \triangleright \lambda x : \sigma \ t : (\sigma \rightarrow \tau) \rrbracket \eta \in R_{\llbracket \sigma \rightarrow \tau \rrbracket \eta}$ ,

3. for all terms  $\Delta, \alpha : T; \Gamma \triangleright t : \tau$  and environments  $\eta : \Delta; \Gamma \rightarrow \mathcal{R}$  such that

$$\forall A \in R_T \llbracket \Delta, \alpha : T; \Gamma \triangleright t : \tau \rrbracket \eta[A/\alpha] \in R_{\llbracket \Delta, \alpha : T \triangleright \tau : T \rrbracket \eta[A/\alpha]},$$

we have  $\llbracket \Delta; \Gamma \triangleright \lambda \alpha t : \forall \alpha \tau \rrbracket \eta \in R_{\llbracket \forall \alpha \tau \rrbracket \eta}$ .

*Proof.*  $\Rightarrow$ : Let  $\mathcal{R}$  be pre-logical. Then  $\mathcal{R}_{Kind}$  is pre-logical and algebraic. To show that  $\mathcal{R}_{Type}$  is algebraic, i.e. that a), b), c) of definition 4.1 hold, we focus on c): If  $f \in R_{(T \Rightarrow T)}$ ,  $h \in R_{\forall f}$  and  $A \in R_T$ , and say  $\Delta = \{v : (T \Rightarrow T), \alpha : T\}$ ,  $\Gamma = \{x : \forall v\}$ , then since  $\Delta; \Gamma \triangleright x \cdot \alpha : v \cdot \alpha$  one has

$$h \cdot A = \llbracket \Delta; \Gamma \triangleright x \cdot \alpha : v \cdot \alpha \rrbracket \eta \in R_{\llbracket \Delta \triangleright v \cdot \alpha : T \rrbracket \eta} = R_{fA}$$

for all environments  $\eta$  with  $\eta(v) = f$ ,  $\eta(\alpha) = A$ , and  $\eta(x) = h$ . That  $\mathcal{R}$  satisfies 2. and 3. on definable functions is seen for 2. via  $\Delta; \Gamma \triangleright \lambda x : \sigma t : (\sigma \rightarrow \tau)$  and for 3. via  $\Delta; \Gamma \triangleright \lambda \alpha t : \forall \alpha \tau$ .

$\Leftarrow$ : Condition (i) of definition 4.2 is clear, and for (ii) the assumptions are just what is needed to prove the claim  $\llbracket \Delta; \Gamma \triangleright t : \tau \rrbracket \eta \in R_{\llbracket \tau \rrbracket \eta}$  by induction on the derivation of  $\Delta; \Gamma \triangleright t : \tau$ .  $\square$

The class of first-order binary pre-logical relations is closed under composition. For second-order relations  $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$  and  $\mathcal{S} \subseteq \mathcal{B} \times \mathcal{C}$ , define  $\mathcal{R} \circ \mathcal{S}$  by taking

$$\begin{aligned} (\mathcal{R} \circ \mathcal{S})_{Kind} &= \{(R \circ S)_\kappa \mid \kappa \in Kind\}, \quad \text{where } (R \circ S)_\kappa := R_\kappa \circ S_\kappa, \\ (\mathcal{R} \circ \mathcal{S})_{Type} &= \{(R \circ S)_{(A,C)} \mid (A,C) \in (R \circ S)_T\}, \quad \text{where} \\ (R \circ S)_{(A,C)} &= \bigcup \{R_{(A,B)} \circ S_{(B,C)} \mid B \in U_T^B, (A,B) \in R_T, (B,C) \in S_T\}. \end{aligned}$$

It is then routine to check the following

**Proposition 4.5.** *Let  $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$  and  $\mathcal{S} \subseteq \mathcal{B} \times \mathcal{C}$  be pre-logical relations between models of  $\lambda_C^{\rightarrow, \forall}$ . Let  $\eta = (\eta^A, \eta^C) : \Delta; \Gamma \rightarrow \mathcal{R} \circ \mathcal{S}$  and suppose there is some  $\eta^B : \Delta; \Gamma \rightarrow \mathcal{B}$  such that  $(\eta^A, \eta^B) : \Delta; \Gamma \rightarrow \mathcal{R}$  and  $(\eta^B, \eta^C) : \Delta; \Gamma \rightarrow \mathcal{S}$ . Then*

$$\llbracket \Delta; \Gamma \triangleright t : \tau \rrbracket \eta \in (R \circ S)_{\llbracket \Delta \triangleright \tau : T \rrbracket \eta}$$

for each term  $\Delta; \Gamma \triangleright t : \tau$ .

This does not quite mean that  $\mathcal{R} \circ \mathcal{S}$  is a pre-logical relation: of course, for each  $\eta : \Delta; \Gamma \rightarrow \mathcal{R} \circ \mathcal{S}$  there is some  $\eta_{Kind}^B : \Delta \rightarrow \mathcal{B}$  such that  $(\eta^A, \eta_{Kind}^B) : \Delta \rightarrow \mathcal{R}_{Kind}$  and  $(\eta_{Kind}^B, \eta^C) : \Delta \rightarrow \mathcal{S}_{Kind}$ . Yet, there may be no extension  $\eta^B = \eta_{Kind}^B \cup \eta_{Type}^B$  as needed in the proposition: if  $\eta(x : \sigma) = (a, c) \in (R \circ S)_{(A,C)}$  for  $(A, C) = \llbracket \Delta \triangleright \sigma : T \rrbracket \eta$ , there is *some*  $B$  such that  $(A, B) \in R_T$ ,  $(B, C) \in S_T$ , and  $(a, b) \in R_{(A,B)}$ ,  $(b, c) \in S_{(B,C)}$ , but  $B$  may depend on  $(a, c)$ , not just  $(A, C)$ , and may be different from  $\llbracket \sigma \rrbracket \eta_{Kind}^B$ .

If for each  $(A, C) \in (R \circ S)_T$ , the relations  $R_{(A,B)} \circ S_{(B,C)}$  for all  $B$  with  $(A, B) \in R_T$  and  $(B, C) \in S_T$  coincide, we can extend  $\eta_{Kind}^B$  by a suitable  $\eta_{Type}^B$ . So we still have the following special case, which may be sufficient for applications of second-order pre-logical relations to step-wise data refinement (cf. [HLSST00]):

**Corollary 4.6.** *If  $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$  and  $\mathcal{S} \subseteq \mathcal{B} \times \mathcal{A}$  are pre-logical and  $R_T$  (or the inverse of  $S_T$ ) is functional, then  $\mathcal{R} \circ \mathcal{S}$  is pre-logical.*

For the same reason, the projection of a pre-logical  $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$  to the first component is a pre-logical predicate on  $\mathcal{A}$  if  $\mathcal{R}$  is functional, but not in general.

**Proposition 4.7.** *Let  $\{\mathcal{R}_i \mid i \in I\}$  be a family of pre-logical predicates on  $\mathcal{A}$ . Then  $\bigcap \{\mathcal{R}_i \mid i \in I\}$  is a pre-logical predicate.*

*Remark 4.8.* By Proposition 7.1 of [HS99], every first-order pre-logical relation  $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$  is the composition of three logical relations,  $embed_A \subseteq \mathcal{A} \times \mathcal{A}[X]$ ,  $\mathcal{R}[X] \subseteq \mathcal{A}[X] \times \mathcal{B}[X]$ , and  $embed_B^{-1} \subseteq \mathcal{B}[X] \times \mathcal{B}$ , where  $X$  is a set of indeterminates. Since the embedding relations are functional, we expect that this result extends to the second-order case, using Corollary 4.6.

## 4.2 Definability and observational equivalence

**Definition 4.9.** *Let  $\mathcal{A} = (\mathcal{U}, \mathcal{D}, \Phi, \Psi, \llbracket \cdot \rrbracket)$  and  $A \in U_T$ . Element  $a \in D_A$  is definable of type  $A$ , if there is a closed term  $; \triangleright t : \tau$  of  $\lambda_C^{\rightarrow, \forall}$  with  $A = \llbracket \triangleright \tau : T \rrbracket^A$  and  $a = \llbracket ; \triangleright t : \tau \rrbracket^A$ . We denote the set of definable elements of type  $A$  by*

$$Def_A^A := \{ \llbracket ; \triangleright t : \tau \rrbracket^A \mid ; \triangleright t : \tau \text{ a closed term, } A = \llbracket \triangleright \tau : T \rrbracket^A \}.$$

Mitchell and Meyer ([MM85], Theorem 4) have characterized the set of definable elements of a model  $\mathcal{A}$  of  $\lambda_C^{\rightarrow, \forall}$  as the intersection of all logical relations on an extension  $\mathcal{A}^*$  of  $\mathcal{A}$  by infinitely many unknowns of each type. Using pre-logical relations, we get a simpler characterization:

**Theorem 4.10** *An element  $a$  of  $\mathcal{A}$  is definable of type  $A \in U_T$  iff for each pre-logical predicate  $\mathcal{R} \subseteq \mathcal{A}$  we have  $A \in R_T$  and  $a \in R_A$ .*

*Proof.*  $\Rightarrow$ : There is a term  $; \triangleright t : \tau$  with  $A = \llbracket \triangleright \tau : T \rrbracket$  and  $a = \llbracket ; \triangleright t : \tau \rrbracket \in D_A$ . For each pre-logical predicate  $\mathcal{R}$  on  $\mathcal{A}$ ,  $\llbracket ; \triangleright t : \tau \rrbracket \in R_{\llbracket \triangleright \tau : T \rrbracket}$  by definition, and  $A = \llbracket \triangleright \tau : T \rrbracket \in R_T$  follows from  $\triangleright \tau : T$ , by 2.3.

$\Leftarrow$ : We show that  $Def = (Def_{Kind}, Def_{Type})$  is a pre-logical predicate on  $\mathcal{A}$ . Then  $A \in Def_T$ , so  $A = \llbracket \triangleright \sigma : T \rrbracket$  for some type  $\triangleright \sigma : T$ , and for  $a \in Def_A$  there is a term  $; \triangleright t : \tau$  with  $a = \llbracket ; \triangleright t : \tau \rrbracket$  and  $\llbracket \triangleright \tau : T \rrbracket = A$ . So  $a$  is definable of type  $A$ . By theorem 2.3,  $Def_{Kind} = \{Def_\kappa \mid \kappa \in Kind\}$  where  $Def_\kappa = \{ \llbracket \mu \rrbracket \mid \triangleright \mu : \kappa \}$ , is pre-logical, since  $\mathcal{U}$  is a model of  $\lambda_C^{\rightarrow, Kind}$ .

For  $Def_{Type}$ , suppose  $\Delta; \Gamma \triangleright t : \tau$  and  $\eta : \Delta; \Gamma \rightarrow Def$ . For each  $v_i : \kappa_i \in \Delta$  and  $x_j : \tau_j \in \Gamma$  there is  $\triangleright \mu_i : \kappa$  and  $; \triangleright t_j : \tau_j$  such that  $\eta(v_i) = \llbracket \mu_i \rrbracket \in Def_\kappa$  and  $\eta(x_j) = \llbracket ; \triangleright t_j : \tau_j \rrbracket \in Def_{\llbracket \triangleright \tau_j : T \rrbracket}$ . With the substitution lemma (cf. [BMM90]),

$$\begin{aligned} \llbracket \Delta; \Gamma \triangleright t : \tau \rrbracket \eta &= \llbracket \Delta; \Gamma \triangleright t : \tau \rrbracket \llbracket \llbracket \triangleright \mu_1 : \kappa_1 \rrbracket / v_1, \dots, \llbracket ; \triangleright t_1 : \tau_1 \rrbracket / x_1, \dots \rrbracket \\ &= \llbracket ; \triangleright (t : \tau) [\mu_1 / v_1, \dots, t_1 / x_1, \dots] \rrbracket \\ &\in Def_{\llbracket \triangleright \tau [\mu_1 / v_1, \dots, t_1 / x_1, \dots] : T \rrbracket} = Def_{\llbracket \Delta \triangleright \tau : T \rrbracket \eta}. \end{aligned}$$

Hence  $Def$  is the least pre-logical predicate on  $\mathcal{A}$ .  $\square$

**Definition 4.11.** Let  $OBS$  be a set of closed types and  $\mathcal{A}, \mathcal{B}$  be models of  $\lambda_C^{\vec{\cdot}, \forall}$  such that  $D_{[\tau]}^{\mathcal{A}} = D_{[\tau]}^{\mathcal{B}}$  for each  $\tau \in OBS$ .  $\mathcal{A}$  and  $\mathcal{B}$  are observationally equivalent, in symbols:  $\mathcal{A} \equiv_{OBS} \mathcal{B}$ , if  $\llbracket ; \triangleright t : \tau \rrbracket^{\mathcal{A}} = \llbracket ; \triangleright t : \tau \rrbracket^{\mathcal{B}}$  for all closed terms  $; \triangleright t : \tau$  where  $\tau \in OBS$ .

**Theorem 4.12** Suppose  $\vdash \sigma = \tau$  whenever  $\llbracket \triangleright \sigma : T \rrbracket^{A \times B} = \llbracket \triangleright \tau : T \rrbracket^{A \times B}$ . Then  $\mathcal{A} \equiv_{OBS} \mathcal{B}$  iff there is a pre-logical relation  $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{B}$  with

$$R_{(A,B)} \cap (Def_A^{\mathcal{A}} \times Def_B^{\mathcal{B}}) \subseteq Id_{A,B} \subseteq D_A^{\mathcal{A}} \times D_B^{\mathcal{B}}$$

for each observable type  $\triangleright \tau : T \in OBS$ , where  $(A, B) = \llbracket \triangleright \tau : T \rrbracket^{A \times B}$ .

*Proof.*  $\Leftarrow$  Let  $; \triangleright t : \tau$  be a term. Then  $\triangleright \tau : T$  and since  $\mathcal{R}$  is pre-logical, we have  $(A, B) \in R_T$  for  $(A, B) = \llbracket \triangleright \tau : T \rrbracket^{A \times B}$ , and  $(a, b) \in R_{(A,B)}$  for  $(a, b) = \llbracket \triangleright t : \tau \rrbracket^{A \times B}$ . If  $\tau : T \in OBS$  then  $a = b$  by the assumption.

$\Rightarrow$ : Take  $\mathcal{R} := Def^{A \times B}$ , which is pre-logical by the previous proof. Suppose  $\triangleright \tau : T \in OBS$  and  $(A, B) = \llbracket \triangleright \tau : T \rrbracket^{A \times B}$ . For  $(a, b) \in R_{(A,B)}$ , there is a term  $; \triangleright s : \sigma$  such that

$$(a, b) = \llbracket ; \triangleright s : \sigma \rrbracket^{A \times B} \quad \text{and} \quad (A, B) = \llbracket \triangleright \sigma : T \rrbracket^{A \times B}.$$

By assumption,  $\vdash \sigma = \tau$ , hence  $; \triangleright s : \tau$  by (*Typ =*), and  $(a, b) = \llbracket ; \triangleright s : \tau \rrbracket^{A \times B}$ . Since  $\mathcal{A} \equiv_{OBS} \mathcal{B}$  and  $\tau \in OBS$ , we get  $a = b$ .  $\square$

### 4.3 Abstract type constructors and representation independence

To simplify the setting, we had assumed that the declaration in example 2.8 introduces an abstract *type*,  $\alpha$  *bag*, and constants  $x : \sigma$  of simple type. But in fact we would like to model a declaration like

$$(\mathbf{abstype} \ (bag : T \Rightarrow T, x : \sigma) \ \mathbf{is} \ (\lambda \alpha : T. \tau, t : \sigma[\lambda \alpha : T. \tau / bag]) \ \mathbf{in} \ s),$$

introducing a *type constructor* and a constant of *polymorphic* type  $\sigma$ . In  $\lambda_C^{\vec{\cdot}, \forall}$  we can do this as follows: we extend  $\mathcal{A} = (\mathcal{U}, \Phi, \Psi, C^{\mathcal{A}}, \llbracket \cdot \rrbracket \cdot)$  to a model  $\mathcal{A}^+$ , by first adjoining (cf. definition 6.1) an indeterminate, *bag*, to the kind structure  $\mathcal{U}$ :

$$\mathcal{U}^+ = \mathcal{U}[bag : T \Rightarrow T] = \{U_{\kappa}^+ \mid \kappa \in Kind\}, \text{ where } U_{\kappa}^+ := U_{(T \Rightarrow T) \Rightarrow \kappa} \cdot bag.$$

This provides us with new types  $(A \text{ bag})$ ,  $((A \text{ bag}) \text{ bag}) \in U_T^+$ , for each  $A \in U_T$ , and we can introduce new constants of polymorphic type, like

$$\mathbf{empty} : \forall \alpha. (\alpha \text{ bag}) \quad \text{or} \quad \mathbf{member} : \forall \alpha. (\alpha \rightarrow ((\alpha \text{ bag}) \rightarrow \mathit{int})).$$

Next, when assigning domains to the new types, we interpret *bag* by a definable constructor, for example the list constructor  $* : T \Rightarrow T$ , and use the domains of the resulting types of  $U_T$  as domains of the new types, i.e.

$$D_{A \text{ bag}}^+ := D_{A*} = (D_A)^*,$$

and as interpretation of the new constants the elements of these domains that are the values of the defining terms, for example

$$\llbracket \text{empty} : \forall \alpha. (\alpha \text{ bag}) \rrbracket^+ := \square \in D_{\llbracket \forall \alpha. \alpha^* \rrbracket} = D_{\llbracket \forall \alpha (\alpha \text{ bag}) \rrbracket}^+$$

Notice that a predicate  $\mathcal{R}$  on  $\mathcal{A}^+$  will have different predicates  $R_{A \text{ bag}}, R_{A^*} \subseteq D_{A \text{ bag}} = D_{A^*}$ , since the types  $A \text{ bag}$  and  $A^*$  are not the same.

**Definition 4.13.** Let  $v_0 : \kappa_0 \triangleright \sigma_0 : T$  and  $C^+ := C_{Kind}, v_0 : \kappa_0 ; C_{Type}, x_0 : \sigma_0$  be an extension of  $C$  by new ‘constants’  $v_0, x_0$ . Let  $\mathcal{A} = (\mathcal{U}, \mathcal{D}, \Phi, \Psi, \llbracket \cdot \rrbracket)$  be an environment model of  $\lambda_C^{\rightarrow, \forall}$ ,  $\triangleright \mu_0 : \kappa_0$  a closed constructor of  $\lambda_{C_{Kind}}^{\rightarrow}$  with value  $k_0 = \llbracket \triangleright \mu_0 : \kappa_0 \rrbracket$ , and  $A = \llbracket \sigma_0[\mu_0/v_0] \rrbracket \in U_T$  and  $a \in D_A$ .

Define the expansion  $\mathcal{A}(\mu_0, a) := \mathcal{A}^+ = (\mathcal{U}^+, \mathcal{D}^+, \Phi^+, \Psi^+, \llbracket \cdot \rrbracket^+)$  as follows:  $\mathcal{U}^+ = \mathcal{U}[v_0 : \kappa_0]$  is the extension of  $\mathcal{U}$  by an indeterminate  $v_0 : \kappa_0$  (cf. definition 6.1). Each  $k \in U_{\kappa}^+$  can be seen as an element of  $U_{(\kappa_0 \Rightarrow \kappa)}$  and hence determines an element  $k(k_0) \in U_{\kappa}$ ; therefore,  $\mathcal{D}^+, \Phi^+, \Psi^+, (\llbracket \cdot \rrbracket^+)$  are given by

$$\begin{array}{ll} D_A^+ & := D_{A(k_0)}, \\ x_0^{\mathcal{D}^+} & := a, & c^{\mathcal{D}^+} & := c^{\mathcal{D}} \text{ for } c : \sigma \in C_{Type}, \\ (\Phi^+)_{A,B}^{\rightarrow} & := \Phi_{A(k_0), B(k_0)}^{\rightarrow}, & (\Psi^+)_{A,B}^{\rightarrow} & := \Psi_{A(k_0), B(k_0)}^{\rightarrow}, \\ (\Phi^+)_{f}^{\forall} & := \Phi_{f(k_0)}^{\forall}, & (\Psi^+)_{f}^{\forall} & := \Psi_{f(k_0)}^{\forall}, \end{array}$$

and  $\llbracket \Delta ; \Gamma \triangleright t : \tau \rrbracket^{\mathcal{D}^+} \eta := \llbracket \Delta ; \Gamma \triangleright t : \tau \rrbracket^{\mathcal{D}} \eta_{k_0}$ , where  $\eta_{k_0}(v) := \eta(v)(k_0) \in U_{\kappa}$  for  $v : \kappa \in \Delta$  and  $\eta_{k_0}(x) := \eta(x) \in D_{A(k_0)}$  for  $x : \sigma \in \Gamma$  with  $A = \llbracket \Delta \triangleright \sigma : T \rrbracket \in U_T^+$ .

We can now give a generalization of the representation independence theorem 2.9 to  $\lambda_C^{\rightarrow, \forall}$  that covers abstract type constructors. (This is also the reason why we did not use  $\lambda_C^{\rightarrow, \forall, \exists}$ : adding  $\exists : (T \Rightarrow T) \Rightarrow T$  to  $\lambda_C^{\rightarrow, \forall}$  to handle abstract types (cf. [MP85]) would not be sufficient; we’d need existential quantifiers of other kinds as well.) For simplicity of notation, we only state the unary version.

**Theorem 4.14** Let  $\mathcal{R}$  be a pre-logical predicate on a model  $\mathcal{A}$  of  $\lambda_C^{\rightarrow, \forall}$ , such that  $\mathcal{R}_{Kind}$  is logical and its elements are definable using parameters of  $R_T$ . For each definable expansion  $\mathcal{A}^+ = \mathcal{A}(\mu_0, a)$  of  $\mathcal{A}$  to a model of  $\lambda_{C^+}^{\rightarrow, \forall}$ , the following are equivalent:

- (i)  $Def_A^+ \subseteq R_A$  for each  $A \in R_T$ .
- (ii) There is a pre-logical predicate  $\mathcal{R}^+ \subseteq \mathcal{A}^+$  such that  $R_T \subseteq R_T^+$  and  $R_A^+ = R_A$  for all  $A \in R_T$ .

*Proof.* (ii)  $\Rightarrow$  (i): for  $A \in R_T \subseteq R_T^+$  we have  $Def_A^+ \subseteq R_A^+$  since  $\mathcal{R}^+$  is pre-logical, so  $Def_A^+ \subseteq R_A$  by  $R_A^+ = R_A$ .

(i)  $\Rightarrow$  (ii): By construction, elements of  $U_{\kappa}^+$  are equivalence classes  $[v_0 : \kappa_0 \triangleright \mu : \kappa]$  of constructors  $\mu \in \lambda_{C_{Kind}, \mathcal{U}}^{\rightarrow}$  with parameters for elements of  $\mathcal{U}$ ; each one can be written as  $[k \cdot v_0]$  with a unique  $k \in U_{(\kappa_0 \Rightarrow \kappa)}$ . We define  $\mathcal{R}_{Kind}^+ \subseteq U^+$  by

$$\mathcal{R}_{Kind}^+ := \{R_{\kappa}^+ \mid \kappa \in Kind\}, \quad \text{where } R_{\kappa}^+ := R_{(\kappa_0 \Rightarrow \kappa)} \cdot v_0 \text{ for } \kappa \in Kind.$$

For  $\mathcal{R}_{Type}^+ = \{R_A^+ \mid A \in R_T^+\}$ , let  $R_A^+$  be the set of elements of  $D_A^+$  that can be defined in  $\lambda_{C^+}^{\rightarrow, \forall}$  with *type* parameters from  $R_T$  and individual parameters from  $\mathcal{R}_{Type}$ , i.e. for  $A \in R_T^+$  (and  $\eta = (\eta_{Kind}; \eta_{Type})$ ) we put

$$\begin{aligned} R_A^+ := \{ \llbracket \Delta; \Gamma \triangleright t : \tau \rrbracket \eta \mid & \Delta; \Gamma \triangleright t : \tau \text{ is a term of } \lambda_{C^+}^{\rightarrow, \forall}, \\ & \eta_{Kind} : \Delta \rightarrow R_T, A = \llbracket \Delta \triangleright \tau : T \rrbracket \eta_{Kind}, \\ & \text{for all } x : \sigma \in \Gamma \text{ is } \sigma \text{ a type of } \lambda_C^{\rightarrow, \forall}, \\ & \eta_{Type} : \Gamma \rightarrow \mathcal{R}_{Type} \}. \end{aligned} \quad (5)$$

**Claim 1**  $\mathcal{R}^+ := (\mathcal{R}_{Kind}^+, \mathcal{R}_{Type}^+)$  is a pre-logical predicate on  $\mathcal{A}^+$ .

*Proof:* (Sketch) Since  $\mathcal{R}_{Kind}$  is logical, not just pre-logical, by lemma 6.2,  $\mathcal{R}_{Kind}^+ \subseteq \mathcal{U}^+$  is a logical, hence pre-logical predicate. For  $\mathcal{R}_{Type}$  we use lemma 4.4. To see that  $\mathcal{R}^+$  is algebraic, use the fact that types  $A \in R_T^+$  can be represented as  $\mu \cdot v_0$  for some  $\mu \in R_{(\kappa_0 \Rightarrow T)}$ , and by the assumption on  $\mathcal{R}$ ,  $\mu$  is definable by a term of  $\lambda_{C_{Kind}^+}^{\rightarrow}$  with parameters from  $R_T$ .

To show 2. and 3. of Lemma 4.4, suppose  $f$  is a function  $\lambda$ -definable with parameters from  $\mathcal{R}^+$ . Use the substitution lemma to replace in  $f$ 's defining term all parameters from  $\mathcal{R}^+$  by their defining  $\lambda_{C^+}^{\rightarrow, \forall}$ -terms with parameters from  $\mathcal{R}_{Type}$ .

**Claim 2** For each  $A \in R_T$  we have  $A \in R_T^+$  and  $R_A^+ = R_A$ .

*Proof:* Let  $A \in R_T$ . Then  $A = \llbracket \alpha : T; \triangleright (\lambda v \alpha) \cdot v_0 : T \rrbracket [A/\alpha] \in R_{(\kappa \Rightarrow T)} \cdot v_0 = R_T^+$ . To show  $R_A \subseteq R_A^+$ , let  $\eta$  be an environment with  $\eta(\alpha : T) = A$  and  $\eta(x : \alpha) = a \in R_A$ . Then  $a = \llbracket \alpha : T; x : \alpha \triangleright x : \alpha \rrbracket \eta \in R_A^+$  by definition of  $R_A^+$ .

To show  $R_A^+ \subseteq R_A$ , let  $a = \llbracket \Delta; \Gamma \triangleright t : \tau \rrbracket \eta \in R_A^+$  where  $\Delta; \Gamma \triangleright t : \tau$  and  $\eta$  have the properties given in (5). Then  $\Delta$  has the form  $\alpha_1 : T, \dots, \alpha_n : T$ , and with  $\Gamma = x_1 : \sigma_1, \dots, x_m : \sigma_m$  the abstraction

$$\bar{t} : \bar{\tau} := \lambda \alpha_1 \dots \lambda \alpha_n \lambda x_1 : \sigma_1 \dots \lambda x_m : \sigma_m \cdot t : \forall \alpha_1 \dots \forall \alpha_n (\sigma_1 \rightarrow \dots \rightarrow \sigma_m \rightarrow \tau)$$

is a closed term of  $\lambda_{C^+}^{\rightarrow, \forall}$  with type  $\bar{A} := \llbracket \triangleright \bar{\tau} : T \rrbracket \in R_T$ . By (i),

$$\llbracket ; \triangleright \bar{t} : \bar{\tau} \rrbracket \in Def_{\bar{A}}^+ \subseteq R_{\bar{A}}.$$

Let  $A_i = \eta(\alpha_i)$ ,  $b_j = \eta(x_j)$  and  $B_j = \llbracket \Delta \triangleright \sigma_j : T \rrbracket \eta$ . By assumption on  $\eta$  and since  $\mathcal{R}$  is pre-logical we have  $A_i, B_j \in R_T$  and  $b_j \in R_{B_j}$ . This gives

$$\begin{aligned} a &= \llbracket ; \triangleright \bar{t} : \bar{\tau} \rrbracket \cdot A_1 \cdots A_n \cdot b_1 \cdots b_m \in R_{\bar{A}} \cdot A_1 \cdots A_n \cdot b_1 \cdots b_m \\ &\subseteq R_{B_1 \rightarrow \dots \rightarrow B_m \rightarrow A} \cdot R_{B_1} \cdots R_{B_m} \subseteq R_A. \end{aligned}$$

□

*Remark 4.15.* For  $\lambda_C^{\rightarrow, \forall, \exists}$ , Mitchell ([Mit86], Theorem 7) states (without proof) a criterion for the equivalence of two representations of an abstract datatype where  $\mathcal{R}^+$  in (ii) is a logical predicate. I am unable to construct such a logical predicate from his version of (i). It seems unlikely that we can modify the pre-logical predicate from the proof of 4.14 to obtain a logical predicate satisfying (ii) (when  $\exists$  is dropped), in particular since  $C$  may have higher-order constants.

## 5 Directions for future work

First, we conjecture that the characterizations of pre-logical relations by logical relations given in [HS99] for first-order models can be transferred to the class of second-order models considered here, but details remain to be checked (cf. remark 4.8). Next, a categorical generalization of second-order pre-logical relations, extending the work on lax logical relations [PPST00], would be useful for applications to categorical models and probably to imperative languages. Third, representation independence theorems for languages with  $\exists$ -types or dependent types and for a general form of abstraction like

**(abstype context with specification is representation in scope)**

or SML's restriction construct `structure :> signature` would be very useful, especially for SML with higher-order functors (cf. [Ler95], section 4). Finally, observational equivalence as the logical relation at  $\exists$ -types (cf. [Mit86]) may have a flexible variant with pre-logical relations, in particular in connection with specification refinement as studied in [Han99].

**Acknowledgement** Thanks to Furio Honsell for directing me to [HS99], and to the referees for helpful comments and criticism.

## 6 Appendix: Adjunction of indeterminates

**Definition 6.1.** Let  $\mathcal{A} = (A, \Phi, \Psi, C^A)$  be an extensional model of  $\lambda_{\vec{C}}$ ,  $\tau \in T$ . Extend  $\lambda_{\vec{C}}$  to  $\lambda_{\vec{C}, \mathcal{A}}$ -terms by adding a constant  $\underline{a}$  for each  $a \in A_\sigma$ . Consider the equivalence of  $\lambda_{\vec{C}, \mathcal{A}}$ -terms of type  $\sigma$  in the free variable  $x : \tau$  given by

$$s(x) =_{\mathcal{A}} t(x) : \iff \forall a \in A_\tau \llbracket s \rrbracket[a/x] = \llbracket t \rrbracket[a/x] \in A_\sigma.$$

Each equivalence class  $[s : \sigma]$  of  $=_{\mathcal{A}}$  can be represented in the form  $[\underline{f} \cdot x]$  where  $f \in A_{\tau \rightarrow \sigma}$ . Since  $\mathcal{A}$  is extensional,  $f = \llbracket \lambda x s \rrbracket$  is unique and  $a \mapsto \underline{a}$  is injective. Let  $\mathcal{A}[x : \tau] := (A', \Phi', \Psi', C')$ , the extension of  $\mathcal{A}$  by the indeterminate  $x : \tau$ , be

$$\begin{aligned} A'_\sigma &= A[x : \tau]_\sigma := \{[s] \mid s \in \lambda_{\vec{C}, \mathcal{A}}, x : \tau \triangleright s : \sigma\} \\ \Phi'_{\rho, \sigma}([\underline{f} \cdot x])([\underline{g} \cdot x]) &:= \underline{\Phi_{\tau \rightarrow \rho, \tau \rightarrow \sigma}(S_{\tau, \rho, \sigma} \cdot f)(g) \cdot x} \\ \Psi'_{\rho, \sigma}(\lambda[\underline{g} \cdot x].[\underline{h}(g) \cdot x]) &:= \underline{\Psi_{\tau \rightarrow \rho, \tau \rightarrow \sigma}(h) \cdot x} \\ C'(c) &:= \underline{C^A(c)} \quad \text{for } c : \sigma \in C, \end{aligned}$$

where  $S_{\tau, \rho, \sigma} := \lambda f \lambda g \lambda x (fx(gx)) : (\tau \rightarrow \rho \rightarrow \sigma) \rightarrow (\tau \rightarrow \rho) \rightarrow (\tau \rightarrow \sigma)$ . We write  $f \cdot x$  instead of  $[\underline{f} \cdot x]$  and, correspondingly,  $A[x : \tau]_\sigma = A_{\tau \rightarrow \sigma} \cdot x$ .

**Lemma 6.2.** Let  $\mathcal{A}$  be an extensional model of  $\lambda_{\vec{C}}$  and  $\mathcal{R} \subseteq \mathcal{A}$  a logical predicate. There is a logical predicate  $\mathcal{S}$  on  $\mathcal{A}[x : \tau]$  with  $[x] \in \mathcal{S}_\tau$  and  $R_\sigma \subseteq \mathcal{S}_\sigma$  for all types  $\sigma$ . If  $R_\tau \neq \emptyset$ , then  $\mathcal{S}_\sigma \cap A_\sigma = R_\sigma$ .

*Proof.* Putting  $\mathcal{S}_\sigma := R_{\tau \rightarrow \sigma} \cdot x$  for all types  $\sigma$ , the claim is easily verified.

Note that  $C$  may contain higher-order constants. This is used for  $\forall$  when applying 6.2 in the proof of theorem 4.14. Unfortunately, the lemma apparently is *wrong* with pre-logical instead of logical relations.

## References

- [BMM90] Kim B. Bruce, Albert R. Meyer, and John C. Mitchell, *The semantics of second-order lambda calculus*, Information and Computation **85** (1990), 76–134.
- [Han99] Jo Erskine Hannay, *Specification refinement with system F*, In Proc. CSL'99, LNCS 1683, Springer Verlag, 1999, pp. 530–545.
- [HLST00] Furio Honsell, John Longley, Donald Sannella, and Andrzej Tarlecki, *Constructive data refinement in typed lambda calculus*, 3rd Intl. Conf. on Foundations of Software Science and Computation Structures. European Joint Conferences on Theory and Practice of Software (ETAPS'2000), LNCS 1784, Springer Verlag, 2000, pp. 149–164.
- [HS99] Furio Honsell and Donald Sannella, *Pre-logical relations*, Proc. Computer Science Logic, CSL'99, LNCS 1683, Springer Verlag, 1999, pp. 546–561.
- [Ler95] Xavier Leroy, *Applicative functors and fully transparent higher-order modules*, Proc. of the 22nd Annual ACM Symposium on Principles of Programming Languages, ACM, 1995, pp. 142–153.
- [Mit86] John C. Mitchell, *Representation independence and data abstraction*, Proceedings of the 13th ACM Symposium on Principles of Programming Languages, January 1986, pp. 263–276.
- [Mit91] John C. Mitchell, *On the equivalence of data representations*, Artificial Intelligence and Mathematical Theory of Computation: Papers in Honour of John C. McCarthy (V. Lifschitz, ed.), Academic Press, 1991, pp. 305–330.
- [Mit96] John C. Mitchell, *Foundations for programming languages*, The MIT Press, Cambridge, Mass., 1996.
- [MM85] John C. Mitchell and Albert Meyer, *Second-order logical relations*, Proc. Logics of Programs, LNCS 193, Springer Verlag, 1985, pp. 225–236.
- [MP85] John C. Mitchell and Gordon D. Plotkin, *Abstract types have existential type*, 12-th ACM Symposium on Principles of Programming Languages, 1985, pp. 37–51.
- [MTHM97] Robin Milner, Mads Tofte, Robert Harper, and David MacQueen, *The definition of Standard ML (revised)*, The MIT Press, Cambridge, MA, 1997.
- [Plo80] Gordon D. Plotkin, *Lambda definability in the full type hierarchy*, To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press, 1980, pp. 363–373.
- [PPST00] Gordon Plotkin, John Power, Don Sannella, and Robert Tennent, *Lax logical relations*, ICALP 2000, Springer LNCS 1853, 2000, pp. 85–102.
- [PR00] John Power and Edmund Robinson, *Logical relations and data abstraction*, Proc. Computer Science Logic, CSL 2000, LNCS 1862, Springer Verlag, 2000, pp. 497–511.
- [Sta85] R. Statman, *Logical relations and the typed lambda calculus*, Information and Control **65** (1985), 85–97.
- [Tai67] W.W. Tait, *Intensional interpretation of functionals of finite type*, Journal of Symbolic Logic **32** (1967), 198–212.