

Prelogical Relations¹

Furio Honsell

Dipartimento di Matematica e Informatica, Università di Udine, Italy
honsell@dimi.uniud.it www.dimi.uniud.it/~honsell/

and

Donald Sannella

Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland
dts@dcs.ed.ac.uk www.dcs.ed.ac.uk/~dts/

We study a weakening of the notion of logical relations, called *prelogical relations*, that has many of the features that make logical relations so useful as well as further algebraic properties including composability. The basic idea is simply to require the reverse implication in the definition of logical relations to hold only for pairs of functions that are expressible by the same lambda term. Prelogical relations are the minimal weakening of logical relations that gives composability for extensional structures and simultaneously the most liberal definition that gives the Basic Lemma. Prelogical predicates (i.e., unary prelogical relations) coincide with sets that are invariant under Kripke logical relations with varying arity as introduced by Jung and Tiuryn, and prelogical relations are the closure under projection and intersection of logical relations. These conceptually independent characterizations of prelogical relations suggest that the concept is rather intrinsic and robust. The use of prelogical relations gives an improved version of Mitchell's representation independence theorem which characterizes observational equivalence for all signatures rather than just for first-order signatures. Prelogical relations can be used in place of logical relations to give an account of data refinement where the fact that prelogical relations compose explains why stepwise refinement is sound.

Key Words: logical relations, typed lambda calculus, semantics, representation independence, data refinement

¹This is an extended and revised version of [9].

1. INTRODUCTION

Logical relations are structure-preserving relations between models of typed lambda calculus.

DEFINITION 1.1. A *logical relation* \mathcal{R} over \mathcal{A} and \mathcal{B} is a family of relations $\{R^\sigma \subseteq \llbracket \sigma \rrbracket^{\mathcal{A}} \times \llbracket \sigma \rrbracket^{\mathcal{B}}\}_{\sigma \in \text{Types}(\Sigma)}$ such that:

- $R^{\sigma \rightarrow \tau}(f, g)$ iff $\forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}}. \forall b \in \llbracket \sigma \rrbracket^{\mathcal{B}}. R^\sigma(a, b) \Rightarrow R^\tau(\text{App}_{\mathcal{A}} f a, \text{App}_{\mathcal{B}} g b)$.
- $R^\sigma(\llbracket c \rrbracket^{\mathcal{A}}, \llbracket c \rrbracket^{\mathcal{B}})$ for every term constant $c : \sigma$ in Σ .

Logical relations are used extensively in the study of typed lambda calculus and have applications outside lambda calculus, for example to abstract interpretation [1] and data refinement [34]. A good reference for logical relations is [19]. An important but more difficult reference is [33].

The Basic Lemma is the key to many of the applications of logical relations. It says that any logical relation over \mathcal{A} and \mathcal{B} relates the interpretation of each lambda term in \mathcal{A} to its interpretation in \mathcal{B} .

LEMMA 1.2 (Basic Lemma). *Let \mathcal{R} be a logical relation over \mathcal{A} and \mathcal{B} . Then for all Γ -environments $\eta_{\mathcal{A}}, \eta_{\mathcal{B}}$ such that $R^\Gamma(\eta_{\mathcal{A}}, \eta_{\mathcal{B}})$ and every term $\Gamma \triangleright M : \sigma$, $R^\sigma(\llbracket \Gamma \triangleright M : \sigma \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}}, \llbracket \Gamma \triangleright M : \sigma \rrbracket_{\eta_{\mathcal{B}}}^{\mathcal{B}})$.*

Here, $R^\Gamma(\eta_{\mathcal{A}}, \eta_{\mathcal{B}})$ refers to the obvious extension of \mathcal{R} to environments, see Sect. 3 below.

As structure-preserving relations, logical relations resemble familiar algebraic concepts such as homomorphisms and congruence relations but they lack some of the convenient properties of such concepts. In particular, the composition of two logical relations is not in general a logical relation. This calls into question their application to data refinement at least, where one would expect composition to provide an account of stepwise refinement.

We propose a weakening of the notion of logical relations called *prelogical relations* (Sect. 3) that has many of the features that make logical relations so useful — in particular, the Basic Lemma still holds for prelogical relations (Lemma 4.1) — but having further algebraic properties including composability (Prop. 5.6). The basic idea is simply to require the reverse implication in the definition of logical relations to hold only for pairs of functions that are expressible by the same lambda term. Prelogical relations turn out to be the minimal weakening of logical relations that gives composability for extensional structures (Corollary 7.2) and simultaneously the most liberal definition that gives the Basic Lemma. Prelogical predicates (the unary case of prelogical relations) coincide with sets that are invariant under Kripke logical relations with varying arity as introduced by Jung and Tiuryn [10] (Prop. 6.2). Moreover, prelogical relations are the closure under projection and intersection of logical relations (Theorem 7.4). In view of these many conceptually independent characterizations, prelogical relations appear to be a rather intrinsic and robust concept. The use of prelogical relations in place of logical relations gives an improved version of Mitchell’s representation independence theorem (Corollaries 8.5 and 8.6 to Theorem 8.4) which characterizes observational equivalence for all signatures rather than just for first-order signatures. Prelogical relations can be used in place of logical relations in Reynolds’ and Tennent’s account of data

refinement in [34] and then the fact that prelogical relations compose explains why stepwise refinement is sound.

Many applications of logical relations follow a standard pattern where the result comes directly from the Basic Lemma once an appropriate logical relation has been defined. Some results in the literature follow similar lines in the sense that a type-indexed family of relations is defined by induction on types and a proof like that of the Basic Lemma is part of the construction, but the family of relations defined is not logical. Examples can be found in Plotkin's and Jung and Tiuryn's lambda-definability results using I-relations [23] and Kripke logical relations with varying arity [10] respectively, and Gandy's proof of strong normalization using hereditarily strict monotonic functionals [6]. In each of these cases, the family of relations involved turns out to be a prelogical relation (Example 3.12, Sect. 6 and Example 3.13) which allows the common pattern to be extended to these cases as well. Since prelogical relations are more general than logical relations and variants like I-relations, they provide a framework within which these different classes can be compared. Here we begin by studying and comparing their closure properties (Prop. 5.7) with special attention to closure under composition.

See Sect. 11 for a discussion of related work.

2. SYNTAX AND SEMANTICS

We begin with λ^\rightarrow , the simply-typed lambda calculus having \rightarrow as the only type constructor. Other type constructors will be considered in Sect. 10. We follow the terminology in [19] for the most part, with slightly different notation.

DEFINITION 2.1. The set of *types* over a set B of *base types* (or *type constants*) is given by the grammar $\sigma ::= b \mid \sigma \rightarrow \sigma$ where b ranges over B . A *signature* Σ consists of a set B of type constants and a collection C of typed *term constants* $c : \sigma$. We write $Types(\Sigma)$ for the set of types over B .

Let $\Sigma = \langle B, C \rangle$ be a signature. We assume familiarity with the usual notions of *context* $\Gamma = x_1:\sigma_1, \dots, x_n:\sigma_n$ and Σ -*term* M of type σ over a context Γ , written $\Gamma \triangleright M : \sigma$, with the meta-variable t reserved for lambda-free Σ -terms. When we need to make Σ explicit we write $\Gamma \triangleright_\Sigma M : \sigma$. If Γ is empty then we write simply $M : \sigma$. Capture-avoiding substitution $[N/x]M$ is as usual.

DEFINITION 2.2. A Σ -*applicative structure* \mathcal{A} consists of:

- a *carrier set* $\llbracket \sigma \rrbracket^{\mathcal{A}}$ for each $\sigma \in Types(\Sigma)$;
- a function $App_{\mathcal{A}}^{\sigma, \tau} : \llbracket \sigma \rightarrow \tau \rrbracket^{\mathcal{A}} \rightarrow \llbracket \sigma \rrbracket^{\mathcal{A}} \rightarrow \llbracket \tau \rrbracket^{\mathcal{A}}$ for each $\sigma, \tau \in Types(\Sigma)$;
- an element $\llbracket c \rrbracket^{\mathcal{A}} \in \llbracket \sigma \rrbracket^{\mathcal{A}}$ for each term constant $c : \sigma$ in Σ .

We drop the subscripts and superscripts when they are determined by the context, and we sometimes abbreviate $App_{\mathcal{A}}^{\sigma, \tau} f x$ as $f x$. Two elements $f, g \in \llbracket \sigma \rightarrow \tau \rrbracket^{\mathcal{A}}$ are said to be *extensionally equal* if $App_{\mathcal{A}}^{\sigma, \tau} f x = App_{\mathcal{A}}^{\sigma, \tau} g x$ for every $x \in \llbracket \sigma \rrbracket^{\mathcal{A}}$. A Σ -applicative structure is *extensional* when extensional equality coincides with identity.

A Σ -*combinatory algebra* is a Σ -applicative structure \mathcal{A} that has elements $K_{\mathcal{A}}^{\sigma, \tau} \in \llbracket \sigma \rightarrow (\tau \rightarrow \sigma) \rrbracket^{\mathcal{A}}$ and $S_{\mathcal{A}}^{\rho, \sigma, \tau} \in \llbracket (\rho \rightarrow \sigma \rightarrow \tau) \rightarrow (\rho \rightarrow \sigma) \rightarrow \rho \rightarrow \tau \rrbracket^{\mathcal{A}}$ for each $\rho, \sigma, \tau \in Types(\Sigma)$ satisfying the equations $K_{\mathcal{A}}^{\sigma, \tau} xy = x$ and $S_{\mathcal{A}}^{\rho, \sigma, \tau} xyz = (xz)(yz)$.

An extensional combinatory algebra is called a *Henkin model*. An applicative structure \mathcal{A} is a *full type hierarchy* when $\llbracket \sigma \rightarrow \tau \rrbracket^{\mathcal{A}} = \llbracket \sigma \rrbracket^{\mathcal{A}} \rightarrow \llbracket \tau \rrbracket^{\mathcal{A}}$ (the full set-theoretic function space) for every $\sigma, \tau \in \text{Types}(\Sigma)$ with $\text{App}_{\mathcal{A}}^{\sigma, \tau} f x = f(x)$ and then it is obviously a Henkin model.

In a combinatory algebra, we can extend the definition of lambda-free Σ -terms by allowing them to contain S and K ; we call these *combinatory Σ -terms*.

A Γ -*environment* $\eta_{\mathcal{A}}$ assigns elements of an applicative structure \mathcal{A} to variables, with $\eta_{\mathcal{A}}(x) \in \llbracket \sigma \rrbracket^{\mathcal{A}}$ for $x : \sigma$ in Γ . A lambda-free Σ -term $\Gamma \triangleright t : \sigma$ is interpreted in a Σ -applicative structure \mathcal{A} under a Γ -environment $\eta_{\mathcal{A}}$ in the obvious way, written $\llbracket \Gamma \triangleright t : \sigma \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}}$, and this extends immediately to an interpretation of combinatory Σ -terms in combinatory algebras by interpreting K and S as $K_{\mathcal{A}}$ and $S_{\mathcal{A}}$. If t is closed then we write simply $\llbracket t : \sigma \rrbracket^{\mathcal{A}}$.

There are various ways of interpreting terms containing lambda abstraction in a combinatory algebra by “compiling” them to combinatory terms so that outermost β holds (see Prop. 2.4 below for what we mean by “outermost β ”). In Henkin models, all these compilations yield the same result.

An axiomatic approach to interpreting lambda abstraction requires an applicative structure equipped with an interpretation function that satisfies certain minimal requirements — cf. the notion of *acceptable meaning function* in [19].

DEFINITION 2.3. A *lambda Σ -applicative structure* consists of a Σ -applicative structure \mathcal{A} together with a function $\llbracket \cdot \rrbracket^{\mathcal{A}}$ that maps any term $\Gamma \triangleright M : \sigma$ and Γ -environment $\eta_{\mathcal{A}}$ over \mathcal{A} to an element of $\llbracket \sigma \rrbracket^{\mathcal{A}}$, such that:

- $\llbracket \Gamma \triangleright x : \sigma \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}} = \eta_{\mathcal{A}}(x)$
- $\llbracket \Gamma \triangleright c : \sigma \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}} = \llbracket c \rrbracket^{\mathcal{A}}$
- $\llbracket \Gamma \triangleright M N : \tau \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}} = \text{App}_{\mathcal{A}} \llbracket \Gamma \triangleright M : \sigma \rightarrow \tau \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}} \llbracket \Gamma \triangleright N : \sigma \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}}$
- $\llbracket \Gamma \triangleright \lambda x : \sigma . M : \sigma \rightarrow \tau \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}} = \llbracket \Gamma \triangleright \lambda y : \sigma . [y/x]M : \sigma \rightarrow \tau \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}}$ provided $y \notin \Gamma$
- $\llbracket \Gamma \triangleright M : \sigma \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}} = \llbracket \Gamma \triangleright M : \sigma \rrbracket_{\eta'_{\mathcal{A}}}^{\mathcal{A}}$ provided $\eta'_{\mathcal{A}}$ is a Γ -environment such that $\eta_{\mathcal{A}}(x) = \eta'_{\mathcal{A}}(x)$ for all $x \in \Gamma$
- $\llbracket \Gamma, x : \sigma \triangleright M : \tau \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}} = \llbracket \Gamma \triangleright M : \tau \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}}$ provided $x \notin FV(M)$

The relationship between lambda applicative structures and combinatory algebras is as follows, cf. [5].

PROPOSITION 2.4. A *lambda applicative structure \mathcal{A} such that $\text{App}_{\mathcal{A}} \llbracket \Gamma \triangleright \lambda x : \sigma . M : \sigma \rightarrow \tau \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}} a = \llbracket \Gamma, x : \sigma \triangleright M : \tau \rrbracket_{\eta_{\mathcal{A}}[x \mapsto a]}^{\mathcal{A}}$ amounts to a combinatory algebra, and vice versa.*

Proof. \Leftarrow : We define $\llbracket \cdot \rrbracket^{\mathcal{A}}$ via the standard compilation of lambda terms using K and S to combinatory terms.

\Rightarrow : $K_{\mathcal{A}}^{\sigma, \tau}$ and $S_{\mathcal{A}}^{\rho, \sigma, \tau}$ are the interpretations of the usual lambda terms. ■

The proof of this proposition shows that the interpretation of lambda terms in combinatory algebras via compilation to combinatory terms satisfies the axioms in Def. 2.3 and the additional property in the proposition. Therefore when viewing

a combinatory algebra as a lambda applicative structure, this is the interpretation function we have in mind.

3. ALGEBRAIC AND PRELOGICAL RELATIONS

We propose a weakening of the definition of logical relations which is closed under composition and which has most of the attractive properties of logical relations. First we change the two-way implication in the condition on functions to a one-way implication which requires preservation of the relation under application.

DEFINITION 3.1. Let \mathcal{A} and \mathcal{B} be Σ -applicative structures. An *algebraic relation* \mathcal{R} over \mathcal{A} and \mathcal{B} is a family of relations $\{R^\sigma \subseteq \llbracket \sigma \rrbracket^{\mathcal{A}} \times \llbracket \sigma \rrbracket^{\mathcal{B}}\}_{\sigma \in \text{Types}(\Sigma)}$ such that:

- If $R^{\sigma \rightarrow \tau}(f, g)$ then $\forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}}. \forall b \in \llbracket \sigma \rrbracket^{\mathcal{B}}. R^\sigma(a, b) \Rightarrow R^\tau(\text{App}_{\mathcal{A}} f a, \text{App}_{\mathcal{B}} g b)$.
- $R^\sigma(\llbracket c \rrbracket^{\mathcal{A}}, \llbracket c \rrbracket^{\mathcal{B}})$ for every term constant $c : \sigma$ in Σ .

In lambda applicative structures, we additionally require the relation to preserve lambda abstraction in a sense that is analogous to the definition of *admissible relation* in [19]. First, we extend a family of relations $\mathcal{R} = \{R^\sigma \subseteq \llbracket \sigma \rrbracket^{\mathcal{A}} \times \llbracket \sigma \rrbracket^{\mathcal{B}}\}_{\sigma \in \text{Types}(\Sigma)}$ to a relation on Γ -environments: $R^\Gamma(\eta_{\mathcal{A}}, \eta_{\mathcal{B}})$ if $R^\sigma(\eta_{\mathcal{A}}(x), \eta_{\mathcal{B}}(x))$ for every $x : \sigma$ in Γ .

DEFINITION 3.2. Let \mathcal{A} and \mathcal{B} be lambda Σ -applicative structures. A *prelogical relation* over \mathcal{A} and \mathcal{B} is an algebraic relation \mathcal{R} such that given Γ -environments $\eta_{\mathcal{A}}$ and $\eta_{\mathcal{B}}$ such that $R^\Gamma(\eta_{\mathcal{A}}, \eta_{\mathcal{B}})$, and a term $\Gamma, x : \sigma \triangleright M : \tau$, if $R^\sigma(a, b)$ implies $R^\tau(\llbracket \Gamma, x : \sigma \triangleright M : \tau \rrbracket_{\eta_{\mathcal{A}}[x \mapsto a]}^{\mathcal{A}}, \llbracket \Gamma, x : \sigma \triangleright M : \tau \rrbracket_{\eta_{\mathcal{B}}[x \mapsto b]}^{\mathcal{B}})$ for all $a \in \llbracket \sigma \rrbracket^{\mathcal{A}}$ and $b \in \llbracket \sigma \rrbracket^{\mathcal{B}}$, then $R^{\sigma \rightarrow \tau}(\llbracket \Gamma \triangleright \lambda x : \sigma. M : \tau \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}}, \llbracket \Gamma \triangleright \lambda x : \sigma. M : \tau \rrbracket_{\eta_{\mathcal{B}}}^{\mathcal{B}})$.

This formulation of the definition amounts to defining prelogical relations as simply the class of relations that make the Basic Lemma hold, as we shall see in Lemma 4.1 below. (Indeed, since the Basic Lemma for prelogical relations is an equivalence rather than a one-way implication, an alternative at this point would be to take the conclusion of the Basic Lemma itself as the definition of prelogical relations.)

A simpler and therefore more appealing definition is obtained if we consider combinatory algebras, where the requirement above boils down to preservation of S and K . This is probably the formulation that readers will most want to remember, although the definition for lambda applicative structures is useful for proofs of syntactic properties of lambda calculus.

PROPOSITION 3.3. Let \mathcal{A} and \mathcal{B} be Σ -combinatory algebras. An algebraic relation \mathcal{R} over \mathcal{A} and \mathcal{B} is prelogical iff $R(S_{\mathcal{A}}^{\rho, \sigma, \tau}, S_{\mathcal{B}}^{\rho, \sigma, \tau})$ and $R(K_{\mathcal{A}}^{\sigma, \tau}, K_{\mathcal{B}}^{\sigma, \tau})$ for all $\rho, \sigma, \tau \in \text{Types}(\Sigma)$.

Proof. Directly from the definitions, using Prop. 2.4 for the inverse implication. ■

If we incorporate S and K into the signature Σ , then prelogical relations are just algebraic relations on combinatory algebras. One way of understanding the definition of prelogical relations is that the reverse implication in the definition of logical

relations is required to hold only for pairs of functions that are expressible by the same lambda term. For combinatory algebras these are exactly the pairs of functions that are denoted by the same combinatory term, and thus this requirement is captured by requiring the relation to contain S and K .

The use of the combinators S and K in the above proposition is in some sense arbitrary: the same result would be achieved by taking any other combinatory basis and changing the definition of combinatory algebra and the interpretation function accordingly. It would be straightforward to modify the definitions to accommodate other variants of lambda calculus, for instance λ_I (where in $\lambda x:\sigma.M$, the term M is required to contain x) for which a combinatory basis is B, C, I, S , or linear lambda calculi. For languages that include recursion, such as PCF, one would add a Y combinator. For $\lambda^{unit, \times, \rightarrow}$, one could add pairing etc. as suggested in Sect. 10 below or alternatively use the “categorical combinators” of [4] as in Theorem 5.1 of [24].

As usual, the binary case of algebraic resp. prelogical relations over \mathcal{A} and \mathcal{B} is derived from the unary case of *algebraic* resp. *prelogical predicates* for the product structure $\mathcal{A} \times \mathcal{B}$. For instance:

DEFINITION 3.4. Let \mathcal{A} be a Σ -applicative structure. An *algebraic predicate* \mathcal{P} over \mathcal{A} is a family of predicates $\{P^\sigma \subseteq \llbracket \sigma \rrbracket^{\mathcal{A}}\}_{\sigma \in \text{Types}(\Sigma)}$ such that:

- If $P^{\sigma \rightarrow \tau}(f)$ then $\forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}}. P^\sigma(a) \Rightarrow P^\tau(\text{App}_{\mathcal{A}} f a)$.
- $P^\sigma(\llbracket c \rrbracket^{\mathcal{A}})$ for every term constant $c : \sigma$ in Σ .

For most results about prelogical relations below there are corresponding results about prelogical predicates and about algebraic relations and predicates over applicative structures. We omit these for lack of space. Similar comments apply to n -ary relations for $n > 2$.

Here is one result about prelogical predicates that does not generalize in the obvious way to prelogical relations:

PROPOSITION 3.5. *A prelogical predicate over a reachable lambda applicative structure \mathcal{A} is a logical predicate on \mathcal{A} . (A lambda Σ -applicative structure \mathcal{A} is reachable if for any $a \in \llbracket \sigma \rrbracket^{\mathcal{A}}$ there is a closed Σ -term $M : \sigma$ such that $a = \llbracket M : \sigma \rrbracket^{\mathcal{A}}$.)*

Proof. By a straightforward induction, or alternatively as a corollary of Lemma 4.2 below. ■

A prelogical relation over reachable lambda applicative structures \mathcal{A} and \mathcal{B} is not necessarily a logical relation over \mathcal{A} and \mathcal{B} , because what is required to apply Prop. 3.5 is reachability for the product structure $\mathcal{A} \times \mathcal{B}$: for any $\langle a, b \rangle \in \llbracket \sigma \rrbracket^{\mathcal{A} \times \mathcal{B}}$ there needs to be a closed Σ -term $M : \sigma$ such that $\langle a, b \rangle = \llbracket M : \sigma \rrbracket^{\mathcal{A} \times \mathcal{B}} = \langle \llbracket M : \sigma \rrbracket^{\mathcal{A}}, \llbracket M : \sigma \rrbracket^{\mathcal{B}} \rangle$.

The fact that prelogicality is strictly weaker than logicality is demonstrated by the following examples which also provide a number of general methods for defining prelogical relations.

EXAMPLE 3.6. For any signature Σ and lambda Σ -applicative structure, the predicate \mathcal{P} defined by

$$P^\sigma(v) \Leftrightarrow v \text{ is the value of a closed } \Sigma\text{-term } M : \sigma$$

is a prelogical predicate over \mathcal{A} . (In fact, \mathcal{P} is the *least* such — see Prop. 5.8 below.) Now, consider the signature Σ containing the type constant nat and term constants $0 : nat$ and $succ : nat \rightarrow nat$ and let \mathcal{A} be the full type hierarchy over \mathbb{N} where 0 and $succ$ have their usual interpretations. \mathcal{P} is not a logical predicate over \mathcal{A} : any function $f \in \llbracket nat \rightarrow nat \rrbracket^{\mathcal{A}}$, including functions that are not lambda definable, takes values in P to values in P and so must itself be in P .

EXAMPLE 3.7. The identity relation on a lambda applicative structure is a prelogical relation but it is logical iff the structure is extensional.

EXAMPLE 3.8. A Σ -homomorphism between lambda Σ -applicative structures \mathcal{A} and \mathcal{B} is a type-indexed family of functions $\{h^\sigma : \llbracket \sigma \rrbracket^{\mathcal{A}} \rightarrow \llbracket \sigma \rrbracket^{\mathcal{B}}\}_{\sigma \in Types(\Sigma)}$ such that for any term constant $c : \sigma$ in Σ , $h^\sigma(\llbracket c \rrbracket^{\mathcal{A}}) = \llbracket c \rrbracket^{\mathcal{B}}$, $h^\tau(App_{\mathcal{A}}^{\sigma, \tau} f a) = App_{\mathcal{B}}^{\sigma, \tau} h^{\sigma \rightarrow \tau}(f) h^\sigma(a)$ and $h^{\sigma \rightarrow \tau}(\llbracket \Gamma \triangleright \lambda x : \sigma. M : \sigma \rightarrow \tau \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}}) = \llbracket \Gamma \triangleright \lambda x : \sigma. M : \sigma \rightarrow \tau \rrbracket_{h(\eta_{\mathcal{A}})}^{\mathcal{B}}$ where $h(\eta_{\mathcal{A}}) = \{x \mapsto h^\sigma(\eta_{\mathcal{A}}(x))\}$ for all $x : \sigma$ in Γ . Any Σ -homomorphism is a prelogical relation. In particular, interpretation of terms in a lambda applicative structure with respect to an environment, viewed as a relation from the lambda applicative structure of terms, is a prelogical relation but is not in general a logical relation.

EXAMPLE 3.9. Prelogical predicates over a lambda applicative structure \mathcal{A} are the natural notion of *substructures* of \mathcal{A} since they require closure under the available operations, see Prop. 4.2 below. When considering extensional structures, we additionally require substructures to be extensional. Logical predicates are unnecessarily constrained for this purpose.

EXAMPLE 3.10. Let \mathcal{A} and \mathcal{B} be lambda applicative structures and define $R^\sigma \subseteq \llbracket \sigma \rrbracket^{\mathcal{A}} \times \llbracket \sigma \rrbracket^{\mathcal{B}}$ by $R^\sigma(a, b)$ for $a \in \llbracket \sigma \rrbracket^{\mathcal{A}}$, $b \in \llbracket \sigma \rrbracket^{\mathcal{B}}$ iff there is a closed term $M : \sigma$ such that $\llbracket M : \sigma \rrbracket^{\mathcal{A}} = a$ and $\llbracket M : \sigma \rrbracket^{\mathcal{B}} = b$. This is a prelogical relation but it is not in general a logical relation. Generalizing: the inverse of any prelogical relation is obviously prelogical and according to Prop. 5.6 below the composition of any two prelogical relations is prelogical. Then observe that the above relation is just the composition of closed term interpretation in \mathcal{B} (which is prelogical according to Example 3.8) and the inverse of closed term interpretation in \mathcal{A} .

EXAMPLE 3.11. Let $\mathcal{A}_1, \dots, \mathcal{A}_m$ be lambda applicative structures. There are several ways of generating an n -ary prelogical relation from an m -ary one $\mathcal{R} \subseteq \mathcal{A}_1 \times \dots \times \mathcal{A}_m$ when $n < m$. *Projection* of any n -tuple of components yields an n -

ary prelogical relation, for example $\pi_{1,2}(\mathcal{R}) \subseteq \mathcal{A}_1 \times \mathcal{A}_2$. Another notation for this is \exists , defined as follows: $(\exists \mathcal{R})^\sigma = \{\langle a_2, \dots, a_m \rangle \mid \exists a_1 \in \llbracket \sigma \rrbracket^{\mathcal{A}_1}. \langle a_1, a_2, \dots, a_m \rangle \in \mathcal{R}^\sigma\}$. The dual of this is \forall , which is another way of reducing the arity of a prelogical relation: $(\forall \mathcal{R})^\sigma = \{\langle a_2, \dots, a_m \rangle \mid \forall a_1 \in \llbracket \sigma \rrbracket^{\mathcal{A}_1}. \langle a_1, a_2, \dots, a_m \rangle \in \mathcal{R}^\sigma\}$. Finally, if $\mathcal{A}_i = \mathcal{A}_j$ for some $i, j \leq m$ then we can take the subset of tuples in \mathcal{R} having the same values in positions i and j : (*filter* \mathcal{R}) $^\sigma = \{\langle a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_m \rangle \mid \langle a_1, \dots, a_m \rangle \in \mathcal{R}^\sigma \text{ and } a_i = a_j\}$. This amounts to a projection of the intersection of \mathcal{R} with the cartesian product of the \mathcal{A}_l 's for $l \neq i, j$ and the diagonal over $\mathcal{A}_i \times \mathcal{A}_j$. All n -ary prelogical relations can be generated from m -ary prelogical relations such that $n < m$ using any of these: to obtain \mathcal{S} , apply any of the above to $\mathcal{S} \times \mathcal{A}_1 \times \dots \times \mathcal{A}_1$ or alternatively, for the case of projection and filtering, to $\{\langle a_1, \dots, a_n, a_1, \dots, a_1 \rangle \mid \langle a_1, \dots, a_n \rangle \in \mathcal{S}\}$. If \mathcal{R} is a logical relation then so is $\forall \mathcal{R}$, but the projection and filtering of \mathcal{R} are not logical relations in general.

EXAMPLE 3.12. Plotkin's *I-relations* [23] give rise to prelogical relations. The family of relations on the full type hierarchy consisting of the tuples which are in a given I-relation at a given world (alternatively, at all worlds) is a prelogical relation which is not in general a logical relation.

A related example concerning Kripke logical relations with varying arity [10] is postponed to Sect. 6 to allow the reader to first become more familiar with prelogical relations.

EXAMPLE 3.13. Let \mathcal{A} be an applicative structure. Given order relations R^b on $\llbracket b \rrbracket^{\mathcal{A}}$ for each base type b , we can define the *hereditarily monotonic functionals* as the equivalence classes of those elements of \mathcal{A} which are self-related with respect to the following inductively defined family of relations on $\mathcal{A} \times \mathcal{A}$:

$$\begin{aligned} R^{\sigma \rightarrow \tau}(f, g) \text{ iff } \forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}}. \forall b \in \llbracket \sigma \rrbracket^{\mathcal{A}}. R^\sigma(a, b) \Rightarrow & (R^\tau(App_{\mathcal{A}} f a, App_{\mathcal{A}} g a) \\ & \wedge R^\tau(App_{\mathcal{A}} f a, App_{\mathcal{A}} f b) \\ & \wedge R^\tau(App_{\mathcal{A}} g a, App_{\mathcal{A}} g b)) \end{aligned}$$

(This defines simultaneously at each type both the class of functionals we are interested in and the order relation itself.) This method defines a prelogical relation which is not in general a logical relation.

Gandy's *hereditarily strict monotonic functionals* [6] can be defined using the above technique with just a small modification of the clause for functionals.

$$\begin{aligned} R^{\sigma \rightarrow \tau}(f, g) \text{ iff } \forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}}. \forall b \in \llbracket \sigma \rrbracket^{\mathcal{A}}. \\ R^\sigma(a, b) \Rightarrow (f \neq g \Rightarrow (R^\tau \setminus \Delta^\tau)(App_{\mathcal{A}} f a, App_{\mathcal{A}} g a) \\ \wedge \\ a \neq b \Rightarrow ((R^\tau \setminus \Delta^\tau)(App_{\mathcal{A}} f a, App_{\mathcal{A}} f b) \\ \wedge (R^\tau \setminus \Delta^\tau)(App_{\mathcal{A}} g a, App_{\mathcal{A}} g b))) \end{aligned}$$

Again we have a prelogical relation (with respect to the language of λ_I) which is not in general a logical relation.

EXAMPLE 3.14. We can define the continuous functionals, as used in models of PCF, using the same technique. Let \mathcal{A} be the full type hierarchy over a given set of base types, and assume that CPO-relations R^b on $\llbracket b \rrbracket^{\mathcal{A}}$ are given. We can define simultaneously at each type both the class of functionals which are continuous and the CPO-relation itself. Namely, define inductively

$$\begin{aligned} R^{\sigma \rightarrow \tau}(f, g) \text{ iff } \forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}}. \forall b \in \llbracket \tau \rrbracket^{\mathcal{A}}. \\ R^{\sigma}(a, b) \Rightarrow (R^{\tau}(App_{\mathcal{A}} f a, App_{\mathcal{A}} g b) \\ \wedge f \text{ and } g \text{ continuous w.r.t. } R^{\sigma} \text{ and } R^{\tau}). \end{aligned}$$

In the above definition, continuity has to be taken in the obvious sense, namely (any representative of) the supremum of a directed set w.r.t. R^{σ} is mapped into (a representative of) the supremum w.r.t. R^{τ} of the images of the directed sets. For the definition to make sense we only need to check that $R^{\sigma \rightarrow \tau}$ induces a CPO-relation on the equivalence classes modulo $R^{\sigma \rightarrow \tau}$ itself. But this is straightforward.

Again, the continuous functionals are the equivalence classes of those elements of \mathcal{A} which are self-related by $R^{\sigma \rightarrow \tau}$ and the CPO-relation is the order relation induced on these equivalence classes by $R^{\sigma \rightarrow \tau}$ itself. Again, \mathcal{R} is a prelogical relation, which is not logical.

4. THE BASIC LEMMA

We will now consider the extension of the Basic Lemma to prelogical relations. In contrast to Lemma 1.2, we get a two-way implication which says that the requirements on prelogical relations are exactly strong enough to ensure that the Basic Lemma holds. The reverse implication fails for logical relations as Example 3.6 shows (for logical predicates).

LEMMA 4.1 (Basic Lemma for prelogical relations). *Let $\mathcal{R} = \{R^{\sigma} \subseteq \llbracket \sigma \rrbracket^{\mathcal{A}} \times \llbracket \sigma \rrbracket^{\mathcal{B}}\}_{\sigma \in \text{Types}(\Sigma)}$ be a family of relations over lambda Σ -applicative structures \mathcal{A} and \mathcal{B} . Then \mathcal{R} is a prelogical relation iff for all Γ -environments $\eta_{\mathcal{A}}, \eta_{\mathcal{B}}$ such that $R^{\Gamma}(\eta_{\mathcal{A}}, \eta_{\mathcal{B}})$ and every Σ -term $\Gamma \triangleright M : \sigma$, $R^{\sigma}(\llbracket \Gamma \triangleright M : \sigma \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}}, \llbracket \Gamma \triangleright M : \sigma \rrbracket_{\eta_{\mathcal{B}}}^{\mathcal{B}})$.*

Proof. \Rightarrow : The proof is by induction on the structure of M . For variables, we use the assumption that $R^{\Gamma}(\eta_{\mathcal{A}}, \eta_{\mathcal{B}})$. For constants, we use the fact that prelogical relations are required to respect constants. For an application $\Gamma \triangleright M N : \tau$, we use the inductive hypothesis for M and N and the fact that prelogical relations are closed under application. As for $\Gamma \triangleright \lambda x : \sigma. M : \sigma \rightarrow \tau$, by the induction hypothesis we know that for all $\Gamma, x : \sigma$ -environments $\eta'_{\mathcal{A}}, \eta'_{\mathcal{B}}$ such that $R^{\Gamma, x : \sigma}(\eta'_{\mathcal{A}}, \eta'_{\mathcal{B}})$ we have $R^{\sigma}(\llbracket \Gamma, x : \sigma \triangleright M : \sigma \rrbracket_{\eta'_{\mathcal{A}}}^{\mathcal{A}}, \llbracket \Gamma, x : \sigma \triangleright M : \sigma \rrbracket_{\eta'_{\mathcal{B}}}^{\mathcal{B}})$. If $R^{\sigma}(a, b)$ then applying this to $\eta'_{\mathcal{A}} = \eta_{\mathcal{A}}[x \mapsto a]$ and $\eta'_{\mathcal{B}} = \eta_{\mathcal{B}}[x \mapsto b]$ and taking the condition in the definition of prelogical relations gives the desired result.

\Leftarrow : The first condition of algebraic relations follows by taking M to be $x y$ and $\eta_{\mathcal{A}} = \{x \mapsto f, y \mapsto a\}$ $\eta_{\mathcal{B}} = \{x \mapsto g, y \mapsto b\}$ and the second condition for a term constant c follows by taking M to be c . The additional condition for prelogical relations holds *a fortiori*. ■

The “only if” direction of this result is the analogue in our setting of the general version of the Basic Lemma in [19], but where \mathcal{R} is only required to be prelogical. For combinatory algebras, the case of lambda abstraction is handled via conversion to an equivalent combinatory term; comparing this proof with the standard proof of the Basic Lemma, where this case uses the reverse implication in the definition of logical relations, exposes the main idea a little more clearly.

If one applies the Basic Lemma for prelogical relations to Henkin models, the “only if” part of the result is exactly the usual formulation (Lemma 1.2 above), except that \mathcal{R} is only required to be prelogical.

The Basic Lemma is intimately connected with the concept of lambda definability. This is most apparent in the unary case:

LEMMA 4.2 (Basic Lemma for prelogical predicates). *Let $\mathcal{P} = \{P^\sigma \subseteq \llbracket \sigma \rrbracket^{\mathcal{A}}\}_{\sigma \in \text{Types}(\Sigma)}$ be a family of predicates over a lambda Σ -applicative structure \mathcal{A} . Then \mathcal{P} is a prelogical predicate iff it is closed under lambda definability: $P^\Gamma(\eta)$ and $\Gamma \triangleright M : \sigma$ implies $P^\sigma(\llbracket \Gamma \triangleright M : \sigma \rrbracket_\eta^{\mathcal{A}})$.*

5. PROPERTIES OF PRELOGICAL RELATIONS

A logical relation on lambda applicative structures is prelogical provided it is admissible in the following sense.

DEFINITION 5.1. [Mitchell] A logical relation \mathcal{R} on lambda applicative structures \mathcal{A} and \mathcal{B} is *admissible* if given Γ -environments $\eta_{\mathcal{A}}$ and $\eta_{\mathcal{B}}$ such that $R^\Gamma(\eta_{\mathcal{A}}, \eta_{\mathcal{B}})$, and terms $\Gamma, x:\sigma \triangleright M, N : \tau$,

$$\forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}}, b \in \llbracket \sigma \rrbracket^{\mathcal{B}}. R^\sigma(a, b) \Rightarrow R^\tau(\llbracket \Gamma, x:\sigma \triangleright M : \tau \rrbracket_{\eta_{\mathcal{A}}[x \mapsto a]}^{\mathcal{A}}, \llbracket \Gamma, x:\sigma \triangleright N : \tau \rrbracket_{\eta_{\mathcal{B}}[x \mapsto b]}^{\mathcal{B}})$$

implies

$$\forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}}, b \in \llbracket \sigma \rrbracket^{\mathcal{B}}. R^\sigma(a, b) \Rightarrow R^\tau(\text{App}_{\mathcal{A}} \llbracket \Gamma \triangleright \lambda x:\sigma. M : \sigma \rightarrow \tau \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}} a, \text{App}_{\mathcal{B}} \llbracket \Gamma \triangleright \lambda x:\sigma. N : \sigma \rightarrow \tau \rrbracket_{\eta_{\mathcal{B}}}^{\mathcal{B}} b)$$

PROPOSITION 5.2. *Any admissible logical relation on lambda applicative structures is a prelogical relation.*

Proof. Admissibility plus the reverse implication in the definition of logical relations gives the property in the definition of prelogical relations. ■

COROLLARY 5.3. *Any logical relation on combinatory algebras is a prelogical relation.*

Proof. Logical relations on combinatory algebras are always admissible. Alternatively, apply Prop. 3.3. ■

To understand why the composition of logical relations \mathcal{R} over \mathcal{A} and \mathcal{B} and \mathcal{S} over \mathcal{B} and \mathcal{C} might not be a logical relation, it is instructive to look at examples. When

composition fails, the problem is often that the interpretation of some function type in \mathcal{B} has “too few values”. But even if we take logical relations over full type hierarchies, where all possible values of function types are present, composition can fail because the required “missing link” in \mathcal{B} is not a function:

EXAMPLE 5.4. Let Σ contain just two type constants, b and b' . Consider three full type hierarchies $\mathcal{A}, \mathcal{B}, \mathcal{C}$ which interpret b and b' as follows: $\llbracket b \rrbracket^{\mathcal{A}} = \{*\} = \llbracket b' \rrbracket^{\mathcal{A}}$; $\llbracket b \rrbracket^{\mathcal{B}} = \{*\}$ and $\llbracket b' \rrbracket^{\mathcal{B}} = \{\circ, \bullet\}$; $\llbracket b \rrbracket^{\mathcal{C}} = \{\circ, \bullet\} = \llbracket b' \rrbracket^{\mathcal{C}}$. Let \mathcal{R} be the logical relation over \mathcal{A} and \mathcal{B} induced by $R^b = \{\langle *, * \rangle\}$ and $R^{b'} = \{\langle *, \circ \rangle, \langle *, \bullet \rangle\}$ and let \mathcal{S} be the logical relation over \mathcal{B} and \mathcal{C} induced by $S^b = \{\langle *, \circ \rangle, \langle *, \bullet \rangle\}$ and $S^{b'} = \{\langle \circ, \circ \rangle, \langle \bullet, \bullet \rangle\}$. $\mathcal{S} \circ \mathcal{R}$ is not a logical relation because it does not relate the identity function in $\llbracket b \rrbracket^{\mathcal{A}} \rightarrow \llbracket b' \rrbracket^{\mathcal{A}}$ to the identity function in $\llbracket b \rrbracket^{\mathcal{C}} \rightarrow \llbracket b' \rrbracket^{\mathcal{C}}$. The problem is that the only two functions in $\llbracket b \rrbracket^{\mathcal{B}} \rightarrow \llbracket b' \rrbracket^{\mathcal{B}}$ are $\{* \mapsto \circ\}$ and $\{* \mapsto \bullet\}$, and \mathcal{S} does not relate these to the identity in \mathcal{C} .

EXAMPLE 5.5. In the previous example, add a type constant *bool* and a term constant $c : (b \rightarrow b') \rightarrow \text{bool}$ to Σ . Let $\llbracket \text{bool} \rrbracket^{\mathcal{A}} = \llbracket \text{bool} \rrbracket^{\mathcal{B}} = \llbracket \text{bool} \rrbracket^{\mathcal{C}} = \{\text{true}, \text{false}\}$ and take $\mathcal{R}^{\text{bool}}$ and $\mathcal{S}^{\text{bool}}$ to be the identity. In each model, let the interpretation of c take constant functions to *true* and all other functions to *false*. The resulting \mathcal{R} and \mathcal{S} are logical relations. As before, $\mathcal{S} \circ \mathcal{R}$ is not a logical relation but now the restriction of $\mathcal{S} \circ \mathcal{R}$ to base types cannot be lifted to a logical relation either: this would relate the identity function in $\llbracket b \rrbracket^{\mathcal{A}} \rightarrow \llbracket b' \rrbracket^{\mathcal{A}}$ (which is a constant function) to every function in $\llbracket b \rrbracket^{\mathcal{C}} \rightarrow \llbracket b' \rrbracket^{\mathcal{C}}$, but then the constant function in \mathcal{A} would be related to non-constant functions in \mathcal{C} , and so $\llbracket c \rrbracket^{\mathcal{A}}$ could not be related to $\llbracket c \rrbracket^{\mathcal{C}}$, otherwise *true* would be related to *false*.

PROPOSITION 5.6. *The composition $\mathcal{S} \circ \mathcal{R}$ of prelogical relations \mathcal{R} over \mathcal{A}, \mathcal{B} and \mathcal{S} over \mathcal{B}, \mathcal{C} is a prelogical relation over \mathcal{A}, \mathcal{C} .*

Proof. A proof from the definition is not at all straightforward, but Lemma 4.1 says that prelogicality is equivalent to a property of relations that is obviously closed under composition. ■

Obviously, closure under composition opens the way to the use of categories in which prelogical relations are morphisms. Many properties of prelogical relations could then be expressed using the language of category theory, but we do not pursue this here.

Composition of relations is definable in terms of product, intersection and projection:

$$\mathcal{S} \circ \mathcal{R} = \pi_{1,3}(\mathcal{A} \times \mathcal{S} \cap \mathcal{R} \times \mathcal{C})$$

Closure of prelogical relations under these operations is a more basic property than closure under composition, and is not specific to binary relations. We have:

PROPOSITION 5.7. *Prelogical relations are closed under intersection, product, projection, permutation and \forall . Logical relations are closed under product, permutation and \forall but not under intersection or projection.*

To see that logical relations are not closed under intersection, consider two logical predicates $\mathcal{P}, \mathcal{P}'$ over \mathcal{A} whose intersection at base types is empty. For any logical predicate \mathcal{Q} over \mathcal{A} which is empty at base types, it is easy to show by induction that for each type σ , either $\mathcal{Q}^\sigma = \llbracket \sigma \rrbracket^{\mathcal{A}}$ or $\mathcal{Q}^\sigma = \emptyset$. But in general $\mathcal{P} \cap \mathcal{P}'$ does not have this property.

Concerning closure under projection, recall \mathcal{S} from Example 5.4 and observe that $\pi_2(\mathcal{S})$ is not a logical predicate: $\pi_2(\mathcal{S})^b = \llbracket b \rrbracket^c$ and $\pi_2(\mathcal{S})^{b'} = \llbracket b' \rrbracket^c$ but $\pi_2(\mathcal{S})^{b \rightarrow b'}$ does not contain the identity function.

Other classes of relations satisfy different closure properties. For instance, I-relations are obviously closed under product, permutation and \forall , and it is easy to see that they are also closed under intersection. They are not closed under composition by Example 5.4 *mutatis mutandis*, and since composition is definable in terms of product, intersection and projection it follows that they are not closed under projection.

A consequence of closure under intersection is that given a property P of relations that is preserved under intersection, there is always a *least* prelogical relation satisfying P . We then have the following lambda-definability result (recall Example 3.6 above):

PROPOSITION 5.8. *The least prelogical predicate over a given lambda Σ -applicative structure contains exactly those elements that are the values of closed Σ -terms.*

In a signature with no term constants, a logical relation may be constructed by defining a relation R on base types and using the definition to “lift” R inductively to higher types. The situation is different for prelogical relations: there are in general many prelogical liftings of a given R , one being of course its lifting to a logical relation (provided this gives an admissible relation). But since the property of lifting a given R is preserved under intersection, the least prelogical lifting of R is also a well-defined relation. Similarly, the least prelogical *extension* of a given family of relations is well-defined for any signature. Notice that lifting \mathcal{R} to a logical relation is not possible in general for signatures containing higher-order term constants, see Example 5.5 (and see [19] for a way of accommodating least fixed-point operators). Extension is also problematic: the cartesian product $\mathcal{A} \times \mathcal{A}$ is a logical relation that trivially extends any binary relation on \mathcal{A} , but this is uninteresting. Further ways of defining prelogical relations are indicated by the examples at the end of Sect. 3 and in Sect. 6.

It is easy to see that prelogical relations are not closed under union. And even in a signature with no term constants, the set of prelogical relations that lift a given relation R on base types cannot be endowed with a lattice structure in general. But the only logical relation in this set is one of its maximal elements under inclusion.

6. KRIPKE LOGICAL RELATIONS WITH VARYING ARITY

In [10], Jung and Tiuryn give the following variant of the notion of logical relations:

DEFINITION 6.1. [Jung and Tiuryn] Let \mathcal{C} be a small category of sets and let \mathcal{A} be a Henkin model. A *Kripke logical relation with varying arity* (KLRwVA for short) over \mathcal{A} is a family of relations R_σ^w indexed by objects w of \mathcal{C} and types σ of $\text{Types}(\Sigma)$, where the elements of R_σ^w are tuples of elements from $\llbracket \sigma \rrbracket^{\mathcal{A}}$ indexed by the elements of w , such that:

- If $f : v \rightarrow w$ is a map in \mathcal{C} and $R_\sigma^w \langle a_j \rangle_{j \in w}$ then $R_\sigma^v \langle a_{f(i)} \rangle_{i \in v}$.
- $R_{\sigma \rightarrow \tau}^w \langle g_j \rangle_{j \in w}$ iff $\forall f : v \rightarrow w. \forall \langle a_i \rangle_{i \in v}. R_\sigma^v \langle a_i \rangle_{i \in v} \Rightarrow R_\tau^v \langle \text{App}_{\mathcal{A}} g_{f(i)} a_i \rangle_{i \in v}$
- $R_\sigma^w \langle \llbracket c \rrbracket^{\mathcal{A}} \rangle_{j \in w}$ for every term constant $c : \sigma$ in Σ .

The above formulation extends Jung and Tiuryn's definition to take term constants into account.

KLRwVAs give rise to prelogical relations in a similar way to I-relations, see Example 3.12: the family of relations consisting of the w -indexed tuples that are in a given KLRwVA at world w is a $|w|$ -ary prelogical relation which is not in general a logical relation, and those elements that are invariant under a given KLRwVA (i.e. $a \in \llbracket \sigma \rrbracket^{\mathcal{A}}$ such that $R_\sigma^w \langle a \rangle_{j \in w}$ for all w) also form a prelogical predicate. More interesting is the fact that every prelogical relation can be obtained in both these ways. We give the unary case first.

PROPOSITION 6.2. Let $\mathcal{P} = \{P^\sigma \subseteq \llbracket \sigma \rrbracket^{\mathcal{A}}\}_{\sigma \in \text{Types}(\Sigma)}$ be a family of predicates over a Henkin model \mathcal{A} . The following are equivalent.

1. \mathcal{P} is a prelogical predicate.
2. \mathcal{P} is the set of elements of \mathcal{A} that appear at some world of cardinality 1 for some KLRwVA.
3. \mathcal{P} is the set of elements of \mathcal{A} that are invariant under some KLRwVA.

Proof. $1 \Rightarrow 2$: Just as in the proof of Lemma 4 in [10], but using terms over Σ expanded by term constants for all values in \mathcal{A} . The proof assumes that \mathcal{A} is a full type hierarchy but it extends to Henkin models by a minor change of notation.

$2 \Leftarrow 1$: From the definitions.

$2 \Leftrightarrow 3$: This follows from the fact that in the KLRwVA constructed in the proof of $1 \Rightarrow 2$, there is only one world of cardinality 1, which is the final object in the category \mathcal{C} . ■

The binary and n -ary cases are obtained by applying the above construction to (n -ary) product structures, or, in what amounts to essentially the same thing, to modifying the structure of the category of worlds. If we work out this latter case we then obtain the (apparently more) general result:

PROPOSITION 6.3. Let $\mathcal{R} = \{R^\sigma \subseteq \llbracket \sigma \rrbracket^{\mathcal{A}} \times \cdots \times \llbracket \sigma \rrbracket^{\mathcal{A}}\}_{\sigma \in \text{Types}(\Sigma)}$ be a family of n -ary relations over a Henkin model \mathcal{A} . \mathcal{R} is a prelogical relation iff it is the set of n -tuples of \mathcal{A} which appear at some world of cardinality n for some KLRwVA.

Finally we point out that the modification to Jung and Tiuryn’s result hinted at in the proof, together with Theorem 3 in [10], can be used to generalize Jung and Tiuryn’s lambda-definability result (their Theorem 5) to signatures containing term constants, cf. [2].

7. PRELOGICAL RELATIONS VIA COMPOSITION OF LOGICAL RELATIONS

Our weakening of the definition of logical relations may appear to be *ad hoc*, but for extensional structures it turns out to be the minimal weakening that is closed under composition. There are variants of this result for several different classes of models. We give the version for Henkin models.

THEOREM 7.1. *Let \mathcal{A} and \mathcal{B} be Henkin models and let \mathcal{R} be a prelogical relation over \mathcal{A} and \mathcal{B} . Then \mathcal{R} factors into a composition of three logical relations over Henkin models.*

Proof idea. The key idea is to extend \mathcal{A} and \mathcal{B} to models $\mathcal{A}[Y]$ and $\mathcal{B}[Y]$ in such a way that $\mathcal{R}[Y]$, a minimally-extended version of \mathcal{R} , becomes a logical relation on $\mathcal{A}[Y]$ and $\mathcal{B}[Y]$. These models are obtained by adding indeterminates with generic behaviour to the carrier sets of \mathcal{A} and \mathcal{B} , which ensures that the condition $\forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}[Y]}. \forall b \in \llbracket \sigma \rrbracket^{\mathcal{B}[Y]}. R[Y]^\sigma(a, b) \Rightarrow R[Y]^\tau(\text{App}_{\mathcal{A}[Y]} f a, \text{App}_{\mathcal{B}[Y]} g b)$ holds only for $f \in \llbracket \sigma \rightarrow \tau \rrbracket^{\mathcal{A}}$ and $g \in \llbracket \sigma \rightarrow \tau \rrbracket^{\mathcal{B}}$ such that $R^{\sigma \rightarrow \tau}(f, g)$. Roughly speaking, we have that for any indeterminate $y : \sigma$, $R[Y]^\tau(\text{App}_{\mathcal{A}[Y]} f y, \text{App}_{\mathcal{B}[Y]} g y)$ iff $R^{\sigma \rightarrow \tau}(f, g)$. Then \mathcal{R} is the composition of the “embedding” of \mathcal{A} in $\mathcal{A}[Y]$, $\mathcal{R}[Y]$, and the inverse of the “embedding” of \mathcal{B} in $\mathcal{B}[Y]$. (The word “embedding” is inaccurate since these are not set-theoretic functions. The inverse of the “embedding” of \mathcal{A} in $\mathcal{A}[Y]$ is a partial surjection in the sense of Sect. 8.4.1 of [19], and likewise for \mathcal{B} .)

A detailed proof may be found in the appendix. ■

It is an open problem whether or not Theorem 7.1 holds if we take the composition of two logical relations rather than three.

COROLLARY 7.2. *The class of binary prelogical relations on Henkin models is the closure under composition of the class of logical relations on such structures.*

This gives the following lambda-definability result:

COROLLARY 7.3. *Let \mathcal{A} be a Henkin model and $a \in \llbracket \sigma \rrbracket^{\mathcal{A}}$. Then $\langle a, a \rangle$ belongs to all relations over $\mathcal{A} \times \mathcal{A}$ obtained by composing logical relations iff $a = \llbracket M : \sigma \rrbracket^{\mathcal{A}}$ for some closed Σ -term $M : \sigma$.*

Proof. By Corollary 7.2 and a binary version of Prop. 5.8. ■

The proof of Theorem 7.1 can be generalized to n -ary relations by repeating the same construction in each component and taking an appropriately generalized notion of composition. This yields the following structure theorem:

THEOREM 7.4. *For any arity n , the class of all n -ary prelogical relations is obtained by closing the class of all logical relations under intersection and then taking projections.*

One can see from the proof of Theorem 7.1 and 7.4 that we can rephrase these results by saying that every prelogical relation over a lambda applicative structure is the intersection of two logical relations on a superstructure of \mathcal{A} , one of which is \mathcal{A}^n itself; or equivalently, the restriction to \mathcal{A} of a logical relation on a superstructure of \mathcal{A} . Moreover, as superstructure we can always take the extension of \mathcal{A} with infinitely many indeterminates at each type. Formally, denoting by $\mathcal{R} \subseteq \mathcal{A}$ the fact that \mathcal{R} is a prelogical predicate over \mathcal{A} , and by $\mathcal{R} \preceq \mathcal{A}$ the fact that \mathcal{R} is a logical predicate over \mathcal{A} , we have:

PROPOSITION 7.5. *Suppose $\mathcal{R} \subseteq \mathcal{A}$. Then there exists \mathcal{A}' and \mathcal{R}' such that $\mathcal{A} \preceq \mathcal{A}'$, $\mathcal{R}' \preceq \mathcal{A}'$ and $\mathcal{R} = \mathcal{A} \cap \mathcal{R}'$.*

Proof. Simply take $\mathcal{R}' = \mathcal{R}[X]$ and $\mathcal{A}' = \mathcal{A}[X]$, where these extensions with indeterminates are as in the proof of Theorem 7.1 in the appendix. ■

For the remainder of this section we shall focus again on the case of binary relations.

Corollary 7.2 does not hold if we restrict ourselves to considering just finite full type hierarchies: given an element a of a finite structure, it turns out to be co-r.e. whether the pair $\langle a, a \rangle$ belongs to all binary relations which are obtainable by closing logical relations under intersection and projection (and hence by closure under composition), while by Prop. 5.8 and [12] it is not co-r.e. whether $\langle a, a \rangle$ belongs to all binary prelogical relations. In the case of arbitrary full type hierarchies, the question is open: the proof of Theorem 7.1 fails if we take a full type hierarchy in place of $\mathcal{A}[X]$, and we conjecture that Corollary 7.2 does not hold.

For non-extensional structures the notion of prelogical relations is not the minimal weakening that gives closure under composition. The following variant is the minimal weakening for this case.

DEFINITION 7.6. An algebraic relation is *extensional* if whenever $R^{\sigma \rightarrow \tau}(f, g)$, f is extensionally equal to f' and g is extensionally equal to g' , we have $R^{\sigma \rightarrow \tau}(f', g')$.

All prelogical relations over extensional structures are automatically extensional, and all logical relations over applicative structures (even non-extensional ones) are automatically extensional as well.

PROPOSITION 7.7. *Let \mathcal{A} and \mathcal{B} be combinatory algebras and let \mathcal{R} be an extensional prelogical relation over \mathcal{A} and \mathcal{B} . Then \mathcal{R} factors into a composition of three logical relations.*

Proof. As for Theorem 7.1 except that we need to take the extensional collapse over \mathcal{A} and \mathcal{B} respectively in the construction of $\mathcal{A}[X]$ and $\mathcal{B}[X]$. The fact that \mathcal{R} is extensional is needed to show that the embeddings are logical relations. ■

COROLLARY 7.8. *The class of extensional prelogical relations on combinatory algebras is the closure under composition of the class of logical relations on such structures.*

These results may suggest that our definition of prelogical relations on non-extensional structures should be strengthened by requiring the relation to be extensional, but this would make the reverse implication of the Basic Lemma fail. So although the notion of extensional prelogical relations is the minimal weakening that gives closure under composition, these are stronger than necessary to give the Basic Lemma.

It would be interesting to continue the above investigations to characterize classes of relations generated by logical relations under operations such as intersection, projection or filtering, particularly on full type hierarchies.

8. REPRESENTATION INDEPENDENCE AND DATA REFINEMENT

Logical relations have been applied to explain the fact that the behaviour of programs does not depend on the way that data types are represented, but only on what can be observed about them using the operations that are provided. “Behaviour of programs” is captured by the notion of observational equivalence.

DEFINITION 8.1. Let \mathcal{A} and \mathcal{B} be lambda Σ -applicative structures and let OBS , the *observable types*, be a subset of $Types(\Sigma)$. Then \mathcal{A} is *observationally finer* than \mathcal{B} with respect to OBS , written $\mathcal{A} \leq_{OBS} \mathcal{B}$, if for any two closed terms $M, N : \sigma$ for $\sigma \in OBS$ such that $\llbracket M : \sigma \rrbracket^{\mathcal{A}} = \llbracket N : \sigma \rrbracket^{\mathcal{A}}$ we have $\llbracket M : \sigma \rrbracket^{\mathcal{B}} = \llbracket N : \sigma \rrbracket^{\mathcal{B}}$.

\mathcal{A} and \mathcal{B} are *observationally equivalent* with respect to OBS , written $\mathcal{A} \equiv_{OBS} \mathcal{B}$, if $\mathcal{A} \leq_{OBS} \mathcal{B}$ and $\mathcal{B} \leq_{OBS} \mathcal{A}$.

It is usual to take OBS to be the “built-in” types for which equality is decidable, for instance *bool* and/or *nat*. Then \mathcal{A} and \mathcal{B} are observationally equivalent iff it is not possible to distinguish between them by performing computational experiments.

Mitchell [19] (cf. [18]) gives the following representation independence result:

THEOREM 8.2 (Mitchell). *Let Σ be a signature that includes a type constant nat , and let \mathcal{A} and \mathcal{B} be Henkin models, with $\llbracket nat \rrbracket^{\mathcal{A}} = \llbracket nat \rrbracket^{\mathcal{B}} = \mathbb{N}$. If there is a logical relation \mathcal{R} over \mathcal{A} and \mathcal{B} with R^{nat} the identity relation on natural numbers, then $\mathcal{A} \equiv_{\{nat\}} \mathcal{B}$. Conversely, if $\mathcal{A} \equiv_{\{nat\}} \mathcal{B}$, Σ provides a closed term for each element of \mathbb{N} having the same value in \mathcal{A} and \mathcal{B} , and Σ contains no higher-order functions, then there is a logical relation \mathcal{R} over \mathcal{A} and \mathcal{B} with R^{nat} the identity relation.*

The statement of this theorem is slightly different from that in [19] — there is an additional minor technical requirement in the second part — because we use a more general notion of observational equivalence than Mitchell does in order to state Theorem 8.4 below.

The following example (Exercise 8.5.6 in [19]) shows that the requirement that Σ contains no higher-order functions is necessary.

EXAMPLE 8.3. Let Σ have type constant nat and term constants $0, 1, 2, \dots : nat$ and $f : (nat \rightarrow nat) \rightarrow nat$. Let \mathcal{A} be the full type hierarchy over $\llbracket nat \rrbracket^{\mathcal{A}} = \mathbb{N}$ with $0, 1, 2, \dots$ interpreted as usual and $\llbracket f \rrbracket^{\mathcal{A}}(g) = 0$ for all $g : \mathbb{N} \rightarrow \mathbb{N}$. Let \mathcal{B} be like \mathcal{A} but with $\llbracket f \rrbracket^{\mathcal{B}}(g) = 0$ if g is computable and $\llbracket f \rrbracket^{\mathcal{B}}(g) = 1$ otherwise. Since the difference between \mathcal{A} and \mathcal{B} cannot be detected by evaluating terms,

$\mathcal{A} \equiv_{\{nat\}} \mathcal{B}$. But there is no logical relation over \mathcal{A} and \mathcal{B} which is the identity relation on nat : if \mathcal{R} is logical then $R^{nat \rightarrow nat}(g, g)$ for any $g : \mathbb{N} \rightarrow \mathbb{N}$, and then $R^{nat}(App_{\mathcal{A}} \llbracket f \rrbracket^{\mathcal{A}} g, App_{\mathcal{B}} \llbracket f \rrbracket^{\mathcal{B}} g)$, which gives a contradiction if g is non-computable.

We will strengthen this result by showing that prelogical relations characterize observational equivalence for *all* signatures. We also generalize to arbitrary sets of observable types and remove the requirement that elements of observable types are denoted by closed terms. This characterization is obtained as a corollary of the following theorem which is a strengthening of Lemma 8.2.17 in [19], again made possible by using prelogical relations in place of logical relations.

THEOREM 8.4. *Let \mathcal{A} and \mathcal{B} be lambda Σ -applicative structures and let $OBS \subseteq Types(\Sigma)$. Then $\mathcal{A} \leq_{OBS} \mathcal{B}$ iff there exists a prelogical relation over \mathcal{A} and \mathcal{B} which is a partial function on OBS .*

Proof. \Leftarrow : Suppose that \mathcal{R} is a prelogical relation over \mathcal{A} and \mathcal{B} which is a partial function on OBS and let $\llbracket M : \sigma \rrbracket^{\mathcal{A}} = \llbracket N : \sigma \rrbracket^{\mathcal{A}}$ for $\sigma \in OBS$. Apply the Basic Lemma to both sides and use the fact that \mathcal{R}^{σ} is a partial function to get $\llbracket M : \sigma \rrbracket^{\mathcal{B}} = \llbracket N : \sigma \rrbracket^{\mathcal{B}}$.

\Rightarrow : Take the relation defined in Example 3.10. ■

Mitchell's Lemma 8.2.17 is the “if” direction of this theorem for Henkin models where $OBS = Types(\Sigma)$ but \mathcal{R} is required to be logical rather than just prelogical. Notice that the “only if” direction of Theorem 8.4 does not hold for logical relations, even in the absence of term constants.

COROLLARY 8.5. *Let \mathcal{A} and \mathcal{B} be lambda Σ -applicative structures and let $OBS \subseteq Types(\Sigma)$. Then $\mathcal{A} \equiv_{OBS} \mathcal{B}$ iff there exists a prelogical relation over \mathcal{A} and \mathcal{B} which is a partial function on OBS in both directions.*

The following result (instantiated to Henkin models) is the special case of Corollary 8.5 that most directly corresponds to Theorem 8.2.

COROLLARY 8.6. *Let Σ be a signature that includes a type constant nat and let \mathcal{A} and \mathcal{B} be lambda Σ -applicative structures with $\llbracket nat \rrbracket^{\mathcal{A}} = \llbracket nat \rrbracket^{\mathcal{B}} = \mathbb{N}$ such that Σ provides a closed term for each element of \mathbb{N} having the same value in \mathcal{A} and \mathcal{B} . There is a prelogical relation \mathcal{R} over \mathcal{A} and \mathcal{B} with R^{nat} the identity relation on natural numbers iff $\mathcal{A} \equiv_{\{nat\}} \mathcal{B}$.*

Proof. Under the condition that $\llbracket nat \rrbracket^{\mathcal{A}} = \llbracket nat \rrbracket^{\mathcal{B}} = \mathbb{N}$, the availability of a closed term for each element of \mathbb{N} having the same value in \mathcal{A} and \mathcal{B} means that the requirement that R^{nat} is the identity relation is equivalent to the requirement that it is a partial function in both directions. ■

EXAMPLE 8.7. Revisiting Example 8.3, the prelogical relation constructed in Example 3.10 has the required property, and it does not relate non-computable functions since they are not lambda definable.

Following [29], it would be interesting to try to characterize finer notions of observational equivalence, e.g. elementary equivalence in first-order logic with equality restricted to observable types. In Example 8.3, note that the sentence $\forall g, h : \text{nat} \rightarrow \text{nat}. f(g) = f(h)$ holds in \mathcal{A} but not in \mathcal{B} .

The standard treatment of data refinement in the context of typed lambda calculus, originating with Reynolds but described most clearly in Sect. 2 of [34], uses logical relations to prove the correctness of refinements: write $\mathcal{A} \mathcal{R} \mathcal{B}$ to indicate that \mathcal{B} is a refinement of \mathcal{A} as witnessed by the logical relation \mathcal{R} over \mathcal{A} and \mathcal{B} , which is required to be identity on observable types. But then, given refinements $\mathcal{A} \mathcal{R} \mathcal{B}$ and $\mathcal{B} \mathcal{S} \mathcal{C}$, the composition $\mathcal{S} \circ \mathcal{R}$ may not be a logical relation and so cannot in general be used as a witness for the composed refinement $\mathcal{A} \rightsquigarrow \mathcal{C}$. (In fact, sometimes there is no witness for $\mathcal{A} \rightsquigarrow \mathcal{C}$ at all.) This is at odds with the *stepwise* nature of refinement, and the transitivity of the underlying concept of refinement expressed in terms of observational equivalence. It is one source of examples demonstrating the incompleteness of this proof method; there are other examples that do not involve composition of refinement steps, see for instance Sect. 5 of [8]. A consequence of Theorem 8.2 is that this proof method *is* complete in the absence of higher-order term constants (if the signature provides a closed term for each element of observable type having the same value in \mathcal{A} and \mathcal{B}).

If prelogical relations are used in place of logical relations, then the fact that the composition of prelogical relations is again a prelogical relation (Prop. 5.6) explains why stepwise refinement is sound. It follows directly from Corollary 8.5 that the result is a sound and complete proof method for proving the correctness of data refinements (provided we relax the condition on observable types to require a partial function in both directions rather than identity, which also allows us to lift the requirement on closed terms of observable types). This opens the way to further development of the foundations of data refinement along the lines of the first-order algebraic treatment in [30], and this is pursued in [8]. There, *constructive prelogical refinement* is a relation between specifications, written $SP \overset{OBS}{\underset{\delta}{\rightsquigarrow}} SP'$. This incorporates a *derived signature morphism* δ defining the types and constants in the signature of SP by giving terms over the signature of SP' , which allows any model \mathcal{B} over the signature of SP' to be transformed to a model $\mathcal{B}|_{\delta}$ over the signature of SP . This refinement is correct if for any $\mathcal{B} \in Mod(SP')$ there is some $\mathcal{A} \in Mod(SP)$ with a prelogical relation \mathcal{R} over \mathcal{A} and $\mathcal{B}|_{\delta}$ which is a partial function on OBS in both directions. Making refinement into a relation on specifications and adding the construction described by δ gives a non-symmetric relation which incorporates the idea that refinement is a *reduction* of one as-yet-unsolved problem to another, which is a better fit with the real-life phenomenon being modelled.

9. OTHER APPLICATIONS

There are many other applications of logical relations. Take for instance the proof of strong normalization of λ^{\rightarrow} in [19]: one defines an admissible logical predicate on a lambda applicative structure of terms by lifting the predicate on base types consisting of the strongly normalizing terms to higher types, proves that the predicate implies strong normalization, and then applies the general version of the

Basic Lemma to give the result. The pattern for proofs of confluence, completeness of leftmost reduction, etc., is the same, sometimes with logical relations in place of logical predicates. There are also constructions that do not involve the Basic Lemma because the relations defined are not logical relations, but that include proofs following the same lines as the proof of the Basic Lemma. Examples include Gandy's proof that the hereditarily strict monotonic functionals model λ_I terms [6], Plotkin's proof that lambda terms satisfy any I-relation [23], and Jung and Tiuryn's proof that lambda terms satisfy any KLRwVA at each arity (Theorem 3 of [10]).

All of these can be cast into a common mould by using prelogical relations rather than logical relations. If a relation or predicate on a lambda applicative structure is logical and admissible, then it is prelogical, and then the Basic Lemma for prelogical relations gives the result. Plotkin's, Jung and Tiuryn's, and Gandy's relations can be shown to be prelogical (in Gandy's case with respect to λ_I), see Example 3.12, Sect. 6 and Example 3.13 respectively, and then the application of the Basic Lemma for prelogical relations gives the result in these cases as well. In each case, however, the interesting part of the proof is not the application of the Basic Lemma (or the argument that replaces its application in the case of Gandy, Plotkin, and Jung and Tiuryn) but rather the construction of the relation and the proof of its properties. The point of the analysis is not to say that this view makes the job easier but rather to bring forward the common pattern in all of these proofs, which is suggestive of a possible methodology for such proofs.

Proofs of completeness and other applications would make use of the following.

DEFINITION 9.1. A family of binary relations $\{R^\sigma \subseteq \llbracket \sigma \rrbracket^{\mathcal{A}} \times \llbracket \sigma \rrbracket^{\mathcal{A}}\}_{\sigma \in \text{Types}(\Sigma)}$ over a Σ -applicative structure \mathcal{A} is a *partial equivalence relation* (abbreviated *PER*) if it is symmetric and transitive for each type.

PROPOSITION 9.2. *Let \mathcal{R} be a PER on a Σ -applicative structure \mathcal{A} which is algebraic. Define the quotient of \mathcal{A} by \mathcal{R} , written \mathcal{A}/\mathcal{R} , as follows:*

- $\llbracket \sigma \rrbracket^{\mathcal{A}/\mathcal{R}} = \llbracket \sigma \rrbracket^{\mathcal{A}}/R^\sigma$, i.e. the set of \mathcal{R} -equivalence classes of objects $a \in \llbracket \sigma \rrbracket^{\mathcal{A}}$ such that $R^\sigma(a, a)$.
- $\text{App}_{\mathcal{A}/\mathcal{R}}^{\sigma, \tau} [f]_{\mathcal{A}/\mathcal{R}} [a]_{\mathcal{A}/\mathcal{R}} = [\text{App}_{\mathcal{A}}^{\sigma, \tau} f a]_{\mathcal{A}/\mathcal{R}}$
- $\llbracket c \rrbracket^{\mathcal{A}/\mathcal{R}} = \llbracket c \rrbracket^{\mathcal{A}}_{\mathcal{A}/\mathcal{R}}$ for each term constant $c : \sigma$ in Σ .

Then:

1. *Let \mathcal{A} be a lambda applicative structure. Then \mathcal{A}/\mathcal{R} is a lambda applicative structure, with $\llbracket \Gamma \triangleright M : \sigma \rrbracket_{\eta_{\mathcal{A}/\mathcal{R}}}^{\mathcal{A}/\mathcal{R}} = \llbracket \llbracket \Gamma \triangleright M : \sigma \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}} \rrbracket_{\mathcal{A}/\mathcal{R}}$ (where $\eta_{\mathcal{A}}(x)$ is a representative of the equivalence class $\eta_{\mathcal{A}/\mathcal{R}}(x)$), iff \mathcal{R} is prelogical.*

2. *Let \mathcal{A} be a combinatory algebra. Then \mathcal{A}/\mathcal{R} is a combinatory algebra iff \mathcal{R} is prelogical.*

3. *\mathcal{A}/\mathcal{R} is an extensional applicative structure iff its restriction to the substructure of \mathcal{A} consisting of the elements in $\text{Dom}(\mathcal{R})$ is a logical relation.*

Proof. The first point amounts to showing that the interpretation function is well-defined, namely that $R^\Gamma(\eta_{\mathcal{A}}, \eta'_{\mathcal{A}}) \Rightarrow R^\sigma(\llbracket \Gamma \triangleright M : \sigma \rrbracket_{\eta_{\mathcal{A}}}^{\mathcal{A}}, \llbracket \Gamma \triangleright M : \sigma \rrbracket_{\eta'_{\mathcal{A}}}^{\mathcal{A}})$, but this follows immediately from the Basic Lemma for lambda applicative structures. The second point follows again from the Basic Lemma. The last point is straightforward from the definitions. ■

The last part of the above proposition says that one application of logical relations, that is their use in obtaining extensional structures by quotienting non-extensional structures — the so-called *extensional collapse* — requires a relation that is logical (on a substructure) rather than merely prelogical.

The above proposition allows us to prove completeness for different classes of structures using the traditional technique of quotienting an applicative structure of terms by a suitable relation defined by provability in a calculus. For non-extensional structures, this is not possible using logical relations because the relation defined by provability is prelogical or algebraic rather than logical.

At this point one could develop a theory analogous to that of homomorphisms, quotients and substructures in universal algebra, but we refrain from doing this here. One would expect analogues of the usual theorems relating these three notions.

10. BEYOND λ^\rightarrow AND APPLICATIVE STRUCTURES

Up to now we have been working in λ^\rightarrow , the simplest version of typed lambda calculus. We will now briefly indicate how other type constructors could be treated so as to obtain corresponding results for extended languages.

As a template, we shall discuss the case of product types. The syntax of types is extended by adding the type form $\sigma \times \tau$ and the syntax of terms is extended by adding pairing $\langle M, N \rangle$ and projections $\pi_1 M$ and $\pi_2 M$. If we regard these as additional term constants in the signature, e.g. $\langle \cdot, \cdot \rangle : \sigma \rightarrow \tau \rightarrow \sigma \times \tau$ for all σ, τ , rather than as new term forms, then the definition of prelogical relations remains the same: the condition on term constants says that e.g. $R^{\sigma \rightarrow \tau \rightarrow \sigma \times \tau}(\llbracket \langle \cdot, \cdot \rangle \rrbracket^{\mathcal{A}}, \llbracket \langle \cdot, \cdot \rangle \rrbracket^{\mathcal{B}})$ and this is all that is required. For models that satisfy surjective pairing, this implies the corresponding condition on logical relations, namely

$$R^{\sigma \times \tau}(a, b) \text{ iff } R^\sigma(\pi_1 a, \pi_1 b) \text{ and } R^\tau(\pi_2 a, \pi_2 b).$$

The treatment of sum types $\sigma + \tau$ is analogous: we just require that $inl : \sigma \rightarrow \sigma + \tau$, $inr : \tau \rightarrow \sigma + \tau$ and $case : \sigma + \tau \rightarrow (\sigma \rightarrow \rho) \rightarrow (\tau \rightarrow \rho) \rightarrow \rho$ take related arguments to related results. Notice that this is also the only way to give the definition for logical relations with sum types since applying the usual pattern yields

$$\begin{aligned} R^{\sigma + \tau}(a, b) \text{ iff} \\ \forall \rho. (\forall f \in \llbracket \sigma \rightarrow \rho \rrbracket^{\mathcal{A}}, f' \in \llbracket \sigma \rightarrow \rho \rrbracket^{\mathcal{B}}, g \in \llbracket \tau \rightarrow \rho \rrbracket^{\mathcal{A}}, g' \in \llbracket \tau \rightarrow \rho \rrbracket^{\mathcal{B}}. \\ R^{\sigma \rightarrow \rho}(f, f') \wedge R^{\tau \rightarrow \rho}(g, g') \Rightarrow R^\rho(case\ a\ f\ g, case\ b\ f'\ g')) \end{aligned}$$

which is not an inductive definition. This demonstrates that the pattern of definition for prelogical relations is more robust than that for logical relations.

A type constructor that has received less attention in the literature is (finite) powerset, $\mathcal{P}(\sigma)$. The treatment would be like that of products and coproducts, given models containing a standard interpretation of *bool*: we add term constants

$\emptyset : \mathcal{P}(\sigma)$, $\{\cdot\} : \sigma \rightarrow \mathcal{P}(\sigma)$, $\cup : \mathcal{P}(\sigma) \rightarrow \mathcal{P}(\sigma) \rightarrow \mathcal{P}(\sigma)$ and $\in : \sigma \rightarrow \mathcal{P}(\sigma) \rightarrow \text{bool}$ and require $R^{\mathcal{P}(\sigma)}(\llbracket \emptyset \rrbracket^{\mathcal{A}}, \llbracket \emptyset \rrbracket^{\mathcal{B}})$ etc. For models in which the interpretation of the powerset type and these new constants are as usual, this amounts to imposing the following condition:

$$R^{\mathcal{P}(\sigma)}(\alpha, \beta) \quad \text{iff} \quad \forall a \in \alpha. \exists b \in \beta. R^\sigma(a, b) \text{ and } \forall b \in \beta. \exists a \in \alpha. R^\sigma(a, b).$$

Note that this is the same pattern used in defining bisimulations.

Various other kinds of types can be considered, including inductive and co-inductive data types (see [3]), universally and existentially quantified types (see [20]), and various flavours of dependent types. We have not yet considered these in any detail, but we are confident that for any of them, one could take any existing treatment of logical relations and modify it by weakening the condition on functions as above without sacrificing the Basic Lemma. We expect that this would even yield improved results as it has above, but this is just speculation.

We have so far restricted attention to structures modelling total functions. Admitting partial functions involves consideration of *typed partial combinatory algebras*, cf. Sect. 5.6.2 of [19]: these are combinatory algebras where App is a partial function, such that $K x y \downarrow$ (and $K x y = x$), $S x y \downarrow$ and $S x y z \simeq (x z) (y z)$. (As usual, $t \downarrow$ denotes the fact that the term t is defined and $t \simeq t'$ means that the two terms t and t' are either both defined and equal or both undefined.) Typed partial combinatory algebras are the appropriate setting for discussing call-by-value combinatory logic or lambda calculus. There is a standard compilation of lambda terms into combinatory terms which maps values (variables, constants and abstractions) to defined terms.

The appropriate notion of prelogical relation for typed partial combinatory algebras coincides with the standard one for combinatory algebras except that the condition for arrow types has to take undefinedness into account. We use the condition

$$\begin{aligned} \text{(A) If } R^{\sigma \rightarrow \tau}(f, g) \text{ then} \\ \forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}}. \forall b \in \llbracket \sigma \rrbracket^{\mathcal{B}}. R^\sigma(a, b) \text{ implies } f a \uparrow \wedge g b \uparrow \\ \text{or } f a \downarrow \wedge g b \downarrow \wedge R^\tau(f a, g b) \end{aligned}$$

which yields both the Basic Lemma (modified slightly to take undefinedness into account) and closure under composition.

Notice that if we had instead taken

$$\begin{aligned} \text{(B) If } R^{\sigma \rightarrow \tau}(f, g) \text{ then} \\ \forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}}. \forall b \in \llbracket \sigma \rrbracket^{\mathcal{B}}. R^\sigma(a, b) \text{ implies } f a \downarrow \wedge g b \downarrow \wedge R^\tau(f a, g b), \end{aligned}$$

then there would be structures for which there are no prelogical relations, for instance when such structures interpret term constants that are not in the domain of the interpretation of other term constants. If we had been too liberal, by taking

$$\begin{aligned} \text{(C) If } R^{\sigma \rightarrow \tau}(f, g) \text{ then} \\ \forall a \in \llbracket \sigma \rrbracket^{\mathcal{A}}. \forall b \in \llbracket \sigma \rrbracket^{\mathcal{B}}. R^\sigma(a, b) \text{ implies } f a \downarrow \wedge g b \downarrow \Rightarrow R^\tau(f a, g b), \end{aligned}$$

then closure under composition would fail. In both cases a revised version of the Basic Lemma would hold. Interestingly, the notion of logical relation arising from

(A) is not the one usually used for extracting a typed partial combinatory algebra from an untyped partial combinatory algebra. This notion is the one arising from (B).

A different dimension of generalization is to consider models having additional structure — e.g. Kripke applicative structures [21], presheaf models or cartesian closed categories — for which logical relations have been studied. We have not yet examined the details of this generalization but a corresponding weakening of the definition would be interesting to consider. It is worth noting that Prop. 6.2 links prelogical relations to KLRwVAs, which have a logical formulation over appropriate presheaf categories as hinted in [10] and have been extended to cartesian closed categories in [2].

11. RELATED WORK

The definition of prelogical relations is not new. In [31], Schoett uses a first-order version of algebraic relations which he calls *correspondences* (see also simulations in [16] and weak homomorphisms in [7]), and he conjectures (p. 281) that for Henkin models, what we have called prelogical relations (formulated as in Prop. 3.3) would be closed under composition and yield the Basic Lemma. In [17], Mitchell makes the same suggestion, referring to Schoett and also crediting Abramsky and Plotkin, but as an assertion rather than a conjecture. The idea is not developed any further. An independent but equivalent definition of prelogical relations over cartesian closed categories, based on the account of logical relations in [14], is given in [24] where they are called *lax logical relations*. It is shown that these compose, that the Basic Lemma holds, and that they coincide with prelogical relations, and an axiomatic account is provided. Earlier, a closely related notion called *L-relations* was defined in [11] and shown to compose. In contrast to [24] and [11], our treatment is elementary rather than categorical, and covers also combinatory algebras. As far as assessing possible categorical generalizations of prelogical relations, some real scientific work ought to be done first in relating the work in [2], [11], [24] and [28] (cf. [25], [26]). In an appropriate categorical setting, the very notion of logical relation appears to have already some desired properties of prelogical relations, which set-theoretical logical relations do not have: e.g. they characterize λ -definable points [2], or capture observational equivalence [28] (cf. [25], [26]). The appropriate categorical generalization of prelogical relations ought to have *all* the properties of prelogical relations in Theorem 12.1 below (*mutatis mutandis*) as well as characterizing observational equivalence. It appears that a slight weakening of lax logical relations might be such a notion.

Another very interesting connection appears with some recent work of Longley, see [13]. He has used *applicative morphisms*, which are essentially total prelogical relations, to study the relationship between various notions of computability.

Two papers that we came across only after writing this paper but which have some intriguing connections to some of our results and techniques, are [32] and [20]. We believe that the concept of prelogical relation would have a beneficial impact on the presentation and understanding of their results.

In [32], Statman gives a characterization of the lambda-definable functionals of an arbitrary type structure as the “stable solutions to systems of functional equations”. Without giving all the details, we just say that a solution is stable if it is α -stable

for all ordinals α , where 1-stability amounts to uniqueness of the solution and a solution is $\alpha + 1$ -stable if it yields α -stable solutions under image and preimage of partial surjective homomorphisms.

Now, partial surjective homomorphisms are just special logical relations, and so α -stability implies stability under suitable α -fold composition of such logical relations. Not surprisingly, the numerology of the number 3 surfaces also in this context, and Statman brilliantly proves that it is enough to look at 3-stable solutions!

Statman does not deal with systems of equations with arbitrary parameters (i.e. signatures). He does not make direct use of indeterminates in proving his results, but these are implicit in the term model of typed lambda calculus. It is not hard to imagine that the notion of prelogical relation and the constructions in the proof of our Theorem 7.1 could shed some light into the essence of this result as well as providing an alternative presentation and possibly a generalization to arbitrary signatures.

Although addressing the issue of logical relations for second-order lambda calculus, the gist of some of Mitchell and Meyer’s results in [20] bears a similar connection to some of our results on prelogical relations. In particular, in their Theorems 4 and 5 Mitchell and Meyer characterize lambda-definable elements of a model and an appropriate notion of observational equivalence between models using logical relations over superstructures defined by adding indeterminates to the given models.

Since [20] is an extended abstract, the proofs of these results cannot be analyzed. But on the basis of our Prop. 7.5 and Corollary 8.5, both Theorems 4 and 5 in [20] appear rather plausible. Again, the use of prelogical relation — if they had been available — would probably have allowed them to phrase their results in a slightly more general fashion.

12. CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

Our feeling is that by introducing the notion of prelogical relation we have, metaphorically and a little immodestly, removed a “blind spot” in the existing intuition of the use and scope of logical relations and related techniques. This is not to say that some specialists in the field have not previously contemplated generalizations similar to ours, but they have not carried the investigation far enough. We believe that in this paper we have exposed very clearly the fact that in many situations the use of logical relations is unnecessarily restrictive. Using prelogical relations instead, we get improved statements of some results (e.g. Theorem 8.4 and its corollaries), we encompass constructions that had previously escaped the logical paradigm (e.g. Example 3.13), and we isolate the necessary and sufficient hypotheses for many arguments to go through (e.g. Lemma 4.1). We have given several characterizations of prelogical relations, summarized in the following theorem (for the unary case):

THEOREM 12.1. *Let $\mathcal{P} = \{P^\sigma \subseteq \llbracket \sigma \rrbracket^{\mathcal{A}}\}_{\sigma \in \text{Types}(\Sigma)}$ be a family of predicates over a Henkin model \mathcal{A} . The following are equivalent.*

1. \mathcal{P} is a prelogical predicate.
2. \mathcal{P} is closed under lambda definability.
3. \mathcal{P} is the set of elements of \mathcal{A} that are invariant under some KLRwVA.

$4.\mathcal{P}$ is the set of elements of \mathcal{A} that are invariant under (or alternatively, the projection or \forall of) the composition of (three) logical relations.

Proof. $1 \Leftrightarrow 2$ is Prop. 4.2, $1 \Leftrightarrow 3$ is Prop. 6.2, and $1 \Leftrightarrow 4$ is by Corollary 7.2. ■

The fact that there are so many conceptually independent ways of defining the same class of relations, together with the fact that they characterize observational equivalence, suggests that it is a truly intrinsic notion. Notice that Thm. 12.1(3) gives an inductive flavour to this concept which is not explicit in the definition of prelogical relation; this apparent lack has been regarded as a weakness of the concept, see e.g. p. 428–429 of [17].

Throughout the paper we have indicated possible directions of future investigation, e.g. suggesting in Sect. 10 how to generalize to less elementary type structures. As we point out, there is a standard methodology here: simply require the interpretations of the “relevant” constants in the two structures to be related. Despite its simplicity, this methodology is extremely rewarding, and it allows to harvest serendipitous results also in related areas. A case in point is offered by PER models of System F, where the extra latitude and flexibility given by defining the exponential PER prelogically allows for a number of possibly novel natural model constructions. It is plausible that sharper characterizations of representation independence similar to the one presented here for simple types will appear in richer type disciplines.

There is a vast literature on logical relations in connection with areas such as parametricity, abstract interpretation, etc. A treatment of these topics in terms of prelogical relations is likely to be as fruitful and illuminating as it has proved to be for the classical example of simply-typed lambda calculus presented here. One possible direction among many would be to study the impact of prelogical relations on the presentation of fully abstract models as in e.g. [27, 22, 15].

In view of the numerous results, connections, new perspectives, and yes, also some criticism, that prelogical relations have brought with them, one can say that changing a \Leftrightarrow to \Rightarrow in the definition of logical relations was like rubbing Aladdin’s lamp — or, some would say, like opening Pandora’s box!

ACKNOWLEDGMENT

Thanks to Samson Abramsky, Jo Hannay, Martin Hofmann, Shin-ya Katsumata, Andrew Kennedy, Yoshiki Kinoshita, Hans Leiß, John Longley, John Mitchell, Peter O’Hearn, Gordon Plotkin, John Power, Ian Stark, Bob Tennent and an anonymous referee for helpful comments. This work has been partially supported by EPSRC grant GR/K63795, an SOEID/RSE Support Research Fellowship, the ESPRIT-funded CoFI and TYPES working groups, and the MURST TOSCA grant.

REFERENCES

1. S. Abramsky. Abstract interpretation, logical relations, and Kan extensions. *Journal of Logic and Computation* 1:5–40 (1990).
2. M. Alimohamed. A characterization of lambda definability in categorical models of implicit polymorphism. *Theoretical Computer Science* 146:5–23 (1995).
3. T. Altenkirch. Logical relations and inductive/coinductive types. *Proc. Computer Science Logic, CSL’98*, Brno. Springer LNCS 1584, 343–354 (1998).

4. P.-L. Curien. *Categorical Combinators, Sequential Algorithms, and Functional Programming*, 2nd edition. Birkhäuser (1993).
5. P. Di Gianantonio and F. Honsell. An abstract notion of application. *Proc. Intl. Conf. on Typed Lambda Calculi and Applications*, Utrecht. Springer LNCS 664, 124–138 (1993).
6. R. Gandy. Proofs of strong normalization. In: *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, 457–477. Academic Press (1980).
7. A. Ginzburg. *Algebraic Theory of Automata*. Academic Press (1968).
8. F. Honsell, J. Longley, D. Sannella and A. Tarlecki. Constructive data refinement in typed lambda calculus. *Proc. 2nd Intl. Conf. on Foundations of Software Science and Computation Structures*, European Joint Conferences on Theory and Practice of Software (ETAPS'2000), Berlin. Springer LNCS 1784, 149–164 (2000).
9. F. Honsell and D. Sannella. Pre-logical relations. *Proc. Computer Science Logic, CSL'99*, Madrid. Springer LNCS 1683, 546–561 (1999).
10. A. Jung and J. Tiuryn. A new characterization of lambda definability. *Proc. Intl. Conf. on Typed Lambda Calculi and Applications*, Utrecht. Springer LNCS 664, 245–257 (1993).
11. Y. Kinoshita, P. O'Hearn, J. Power, M. Takeyama and R. Tennent. An axiomatic approach to binary logical relations with applications to data refinement. *Proc. TACS'97*, Springer LNCS 1281, 191–212 (1997).
12. R. Loader. The undecidability of λ -definability. In: *Logic, Meaning and Computation: Essays in Memory of Alonzo Church*. Kluwer Academic (2001).
13. J. Longley. Matching typed and untyped realizability. *Proc. Tutorial Workshop on Realizability Semantics and Applications*, Trento. *Electronic Notes in Theoretical Computer Science* 23 No. 1 (1999).
14. Q. Ma and J. Reynolds. Types, abstraction, and parametric polymorphism, part 2. *Proc. 7th Intl. Conf. on Mathematical Foundations of Programming Semantics*, Pittsburgh. Springer LNCS 598, 1–40 (1992).
15. M. Marz, A. Rohr and T. Streicher. Full abstraction and universality via realisability. *Proc. 14th IEEE Symp. on Logic in Computer Science*, Trento (1999).
16. R. Milner. An algebraic definition of simulation between programs. *Proc. 2nd Intl. Joint Conf. on Artificial Intelligence*. British Computer Society, 481–489 (1971).
17. J. Mitchell. Type Systems for Programming Languages. Chapter 8 of *Handbook of Theoretical Computer Science, Vol B*. Elsevier (1990).
18. J. Mitchell. On the equivalence of data representations. In: *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy* (V. Lifschitz, ed.). Academic Press, 305–330 (1991).
19. J. Mitchell. *Foundations for Programming Languages*. MIT Press (1996).
20. J. Mitchell and A. Meyer. Second-order logical relations. *Proc. of the Conf. on Logic of Programs*, Brooklyn. Springer LNCS 193, 225–236 (1985).
21. J. Mitchell and E. Moggi. Kripke-style models for typed lambda calculus. *Annals of Pure And Applied Logic* 51:99–124 (1991).
22. P. O'Hearn and J. Riecke. Kripke logical relations and PCF. *Information and Computation* 120:107–116 (1995).
23. G. Plotkin. Lambda-definability in the full type hierarchy. In: *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, 363–373. Academic Press (1980).
24. G. Plotkin, J. Power, D. Sannella and R. Tennent. Lax logical relations. *Proc. 27th Intl. Colloq. on Automata, Languages and Programming*, Geneva. Springer LNCS 1853, 85–102 (2000).
25. J. Power and E. Robinson. Logical relations and data abstraction. *Proc. Computer Science Logic, CSL 2000*, Fischbachau. Springer LNCS 1862 (2000).
26. J. Power and E. Robinson. Logical relations, data abstraction and structured fibrations. *Proc. 2nd Intl. Conf. on Principles and Practice of Declarative Programming*, Montreal (2000).
27. J. Riecke and A. Sandholm. A relational account of call-by-value sequentiality. *Proc. 12th IEEE Symp. on Logic in Computer Science*, Warsaw, 258–267 (1997). *Information and Computation*, to appear.
28. E. Robinson. Logical relations and data abstraction. Report 730, Queen Mary and Westfield College (1996).

29. D. Sannella and A. Tarlecki. On observational equivalence and algebraic specifications. *Journal of Computer and System Sciences* 34:150–178 (1987).
30. D. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: implementations revisited. *Acta Informatica* 25:233–281 (1988).
31. O. Schoett. Data Abstraction and the Correctness of Modular Programming. Ph.D. thesis CST-42-87, Univ. of Edinburgh (1987).
32. R. Statman. λ -definable functionals and $\beta\eta$ conversion. *Arch. math. Logik* 23:21–26 (1983).
33. R. Statman. Logical relations and the typed lambda calculus. *Information and Control* 65:85–97 (1985).
34. R. Tennent. Correctness of data representations in Algol-like languages. In: *A Classical Mind: Essays in Honour of C.A.R. Hoare*. Prentice Hall (1994).

APPENDIX

Here we give a detailed proof of the following theorem.

THEOREM 7.1. *Let \mathcal{A} and \mathcal{B} be Henkin models and let \mathcal{R} be a prelogical relation over \mathcal{A} and \mathcal{B} . Then \mathcal{R} factors into a composition of three logical relations over Henkin models.*

See Sect. 7 for the idea of the proof. We proceed as follows:

- We define the model $\mathcal{A}[Y]$ and the relation that embeds \mathcal{A} in $\mathcal{A}[Y]$, and show that this is a logical relation (Lemma A.7). The same applies to \mathcal{B} and $\mathcal{B}[Y]$.
- We define $\mathcal{R}[Y] \subseteq \mathcal{A}[Y] \times \mathcal{B}[Y]$ and show that it is a logical relation (Lemma A.9).
- Finally we show that \mathcal{R} is the composition of the embedding of \mathcal{A} in $\mathcal{A}[Y]$, $\mathcal{R}[Y]$, and the inverse of the embedding of \mathcal{B} in $\mathcal{B}[Y]$ (Lemma A.10).

Let Y be a set of typed variables containing an infinite number of variables for each type in $Types(\Sigma)$.

DEFINITION A.1. Let $\Sigma\mathcal{A}$ be Σ augmented by a term constant $c_a : \sigma$ for each element $a \in \llbracket \sigma \rrbracket^{\mathcal{A}}$. Let $\Sigma[Y]$ (resp. $\Sigma\mathcal{A}[Y]$) be Σ (resp. $\Sigma\mathcal{A}$) augmented by a term constant $c_y : \sigma$, called an *indeterminate*, for each variable $y : \sigma$ in Y .

DEFINITION A.2. The Henkin model $\mathcal{A}[Y]$ over $\Sigma[Y]$ is the (closed) $\Sigma\mathcal{A}[Y]$ -term model of the simply-typed lambda calculus with β and η , and with δ -reductions describing the behaviour of each constant c_a with respect to other such constants (and with no δ -reductions involving the indeterminates). This calculus is Church-Rosser and strongly normalizing so each term M has a unique normal form \widehat{M} . Although elements in $\mathcal{A}[Y]$ are equivalence classes of terms, it is convenient to refer to them using single terms, by which is meant the equivalence class containing that term. Since all terms in a given equivalence class have the same normal form, we will sometimes write \widehat{a} for an element a in $\mathcal{A}[Y]$. If η is a Γ -environment over \mathcal{A} then η_c is the Γ -environment over $\mathcal{A}[Y]$ which assigns each variable x to (the equivalence class of) $c_{\eta(x)}$.

Please note that $\mathcal{A}[Y]$ is not the construction corresponding to the “free combinatory Σ -algebra” generated by the constants in \mathcal{A} , which is more frequently used in the literature. Rather, it is the construction corresponding to the “free Σ -algebra”

generated by the constants in \mathcal{A} . So the interpretations of pure closed lambda terms, such as K or S , are not preserved in the generic extension. For instance, $K \neq \llbracket c_K \rrbracket^{\mathcal{A}[Y]}$.

LEMMA A.3. *If $M \rightarrow_{\beta\eta\delta}^* N$ for $\Gamma \triangleright_{\Sigma\mathcal{A}} M, N : \sigma$, then for any Γ -environment η over \mathcal{A} , $\llbracket M \rrbracket_{\eta}^{\mathcal{A}} = \llbracket N \rrbracket_{\eta}^{\mathcal{A}}$ in which we interpret each of the constants c_a as a .*

Proof. By induction on the structure of the derivation of $M \rightarrow_{\beta\eta\delta}^* N$. ■

LEMMA A.4. *Let $\Gamma \triangleright_{\Sigma} M, N : \sigma$. If $\llbracket N \rrbracket_{\eta_c}^{\mathcal{A}[Y]} = \llbracket M \rrbracket_{\eta'_c}^{\mathcal{A}[Y]}$ then $\llbracket N \rrbracket_{\eta}^{\mathcal{A}} = \llbracket M \rrbracket_{\eta'}^{\mathcal{A}}$.*

Proof. If $\llbracket N \rrbracket_{\eta_c}^{\mathcal{A}[Y]} = \llbracket M \rrbracket_{\eta'_c}^{\mathcal{A}[Y]}$ then $N[\eta_c(x)/x$ for all $x]$ and $M[\eta'_c(x)/x$ for all $x]$ reduce to the same normal form. By Lemma A.3, the reduction sequences that yield this normal form can be reproduced in \mathcal{A} to give $\llbracket N \rrbracket_{\eta}^{\mathcal{A}} = \llbracket M \rrbracket_{\eta'}^{\mathcal{A}}$. ■

LEMMA A.5. *Let $\Gamma \triangleright_{\Sigma} M : \sigma$ and $\Gamma \triangleright_{\Sigma[Y]} N : \sigma$. If $\llbracket M \rrbracket_{\eta_c}^{\mathcal{A}[Y]} = \llbracket N \rrbracket_{\eta_c}^{\mathcal{A}[Y]}$ then $\widehat{\llbracket N \rrbracket_{\eta_c}^{\mathcal{A}[Y]}}$ is a $\Sigma\mathcal{A}$ -term, and moreover the $\beta\eta$ -normal form of N contains no indeterminates.*

Proof. The presence of indeterminates does not give rise to extra reductions, nor can indeterminates be erased by the δ -reductions associated to the constants c_a . Any indeterminate in the $\beta\eta$ -normal form of N would therefore falsify the hypothesis. ■

DEFINITION A.6. The embedding *emb* of \mathcal{A} into $\mathcal{A}[Y]$ is defined as follows:

$$emb^{\sigma} = \{ \langle \llbracket M \rrbracket_{\eta}^{\mathcal{A}}, \llbracket M \rrbracket_{\eta_c}^{\mathcal{A}[Y]} \rangle \mid \Gamma \triangleright_{\Sigma} M : \sigma \text{ and } \eta \text{ a } \Gamma\text{-environment over } \mathcal{A} \}$$

Although we call *emb* an embedding, this is inaccurate since it is never a set-theoretic function. To be precise, it is the natural lifting to a logical relation of the embedding at base types. It is easy to check however that emb^{-1} is a partial surjection.

LEMMA A.7. *emb is a logical relation.*

Proof. It is easy to see that each constant c in Σ is in relation to itself: just take $M = c$. Furthermore, suppose $emb^{\sigma \rightarrow \tau}(\llbracket M \rrbracket_{\eta}^{\mathcal{A}}, \llbracket M \rrbracket_{\eta_c}^{\mathcal{A}[Y]})$ and $emb^{\sigma}(\llbracket N \rrbracket_{\eta'}^{\mathcal{A}}, \llbracket N \rrbracket_{\eta'_c}^{\mathcal{A}[Y]})$; we need to show $emb^{\tau}(App_{\mathcal{A}} \llbracket M \rrbracket_{\eta}^{\mathcal{A}} \llbracket N \rrbracket_{\eta'}^{\mathcal{A}}, App_{\mathcal{A}[Y]} \llbracket M \rrbracket_{\eta_c}^{\mathcal{A}[Y]} \llbracket N \rrbracket_{\eta'_c}^{\mathcal{A}[Y]})$. Without loss of generality (because we can rename variables, and denotations depend only on variables which occur) we can assume that $\eta = \eta'$ and then the result follows from the definition of interpretation: $App_{\mathcal{A}} \llbracket M \rrbracket_{\eta}^{\mathcal{A}} \llbracket N \rrbracket_{\eta}^{\mathcal{A}} = \llbracket M N \rrbracket_{\eta}^{\mathcal{A}}$ and similarly for $\mathcal{A}[Y]$.

It remains to prove the opposite direction of the implication. Suppose that whenever $emb^{\sigma}(a, d)$, $emb^{\tau}(App_{\mathcal{A}} f a, App_{\mathcal{A}[Y]} g d)$. We need to show $emb^{\sigma \rightarrow \tau}(f, g)$.

Consider the normal form \widehat{g} of g . We claim that \widehat{g} is a $\Sigma\mathcal{A}$ -term, i.e. it contains no indeterminates. We know that $emb^\sigma(a, \llbracket x \rrbracket_{\eta_c}^{\mathcal{A}[Y]})$ with $\eta_c(x) = c_a$. Since $emb^\tau(App_{\mathcal{A}} f a, App_{\mathcal{A}[Y]} g \llbracket x \rrbracket_{\eta_c}^{\mathcal{A}[Y]})$, there is a term $\Gamma \triangleright_{\Sigma} N : \tau$ and Γ -environment ξ such that $App_{\mathcal{A}[Y]} g \llbracket x \rrbracket_{\eta_c}^{\mathcal{A}[Y]} = \llbracket N \rrbracket_{\xi_c}^{\mathcal{A}[Y]}$. Again w.l.o.g. $\xi = \eta$, so $App_{\mathcal{A}[Y]} \widehat{g} \llbracket x \rrbracket_{\eta_c}^{\mathcal{A}[Y]}$ is a $\Sigma\mathcal{A}$ -term by Lemma A.5, and then so is \widehat{g} . Therefore, consider the Σ -term M and environment η'_c such that $\llbracket M \rrbracket_{\eta'_c}^{\mathcal{A}[Y]} = \widehat{g}$: M is \widehat{g} with each constant c_a replaced by a variable which η'_c maps to c_a . But now we claim that f is extensionally equal to $\llbracket M \rrbracket_{\eta'_c}^{\mathcal{A}}$. We have $emb^\tau(App_{\mathcal{A}} f a, App_{\mathcal{A}[Y]} \llbracket M \rrbracket_{\eta'_c}^{\mathcal{A}[Y]} c_a)$ so $emb^\tau(App_{\mathcal{A}} f a, \llbracket Mz \rrbracket_{\eta'_c[z \mapsto c_a]}^{\mathcal{A}[Y]})$ for a fresh variable z . By the definition of emb , there is a term $\Gamma \triangleright_{\Sigma} M' : \tau$ and Γ -environment η'' such that $\llbracket M' \rrbracket_{\eta''}^{\mathcal{A}} = App_{\mathcal{A}} f a$ and $\llbracket M' \rrbracket_{\eta''}^{\mathcal{A}[Y]} = \llbracket Mz \rrbracket_{\eta'_c[z \mapsto c_a]}^{\mathcal{A}[Y]}$. Again w.l.o.g., $\eta'' = \eta'_c[z \mapsto c_a]$. Then $\llbracket M' \rrbracket_{\eta''}^{\mathcal{A}} = \llbracket Mz \rrbracket_{\eta''}^{\mathcal{A}}$ by Lemma A.4, and thus $App_{\mathcal{A}} f a = App_{\mathcal{A}} \llbracket M \rrbracket_{\eta'_c}^{\mathcal{A}} a$ which shows that f is extensionally equal to $\llbracket M \rrbracket_{\eta'_c}^{\mathcal{A}}$. So by extensionality, $\llbracket M \rrbracket_{\eta'_c}^{\mathcal{A}} = f$ giving $emb^{\sigma \rightarrow \tau}(f, g)$. ■

We will need to refer to both $emb \subseteq \mathcal{A} \times \mathcal{A}[Y]$ and $emb \subseteq \mathcal{B} \times \mathcal{B}[Y]$ so we use the notation $emb_{\mathcal{A}}$ for the former and $emb_{\mathcal{B}}$ for the latter.

DEFINITION A.8. The relation $\mathcal{R}[Y] \subseteq \mathcal{A}[Y] \times \mathcal{B}[Y]$ is defined as follows:

$$\mathcal{R}[Y]^\sigma = \{ \langle \llbracket M \rrbracket_{\eta_c}^{\mathcal{A}[Y]}, \llbracket M \rrbracket_{\eta'_c}^{\mathcal{B}[Y]} \rangle \mid \Gamma \triangleright_{\Sigma[Y]} M : \sigma \text{ and } \eta, \eta' \text{ are } \Gamma\text{-environments for } \mathcal{A}, \mathcal{B} \text{ such that } R(\eta, \eta') \}$$

LEMMA A.9. $\mathcal{R}[Y]$ is a logical relation.

Proof. Each constant c in $\Sigma[Y]$ is in relation to itself: just take $M = c$. Furthermore, suppose $\mathcal{R}[Y](\llbracket M \rrbracket_{\eta_c}^{\mathcal{A}[Y]}, \llbracket M \rrbracket_{\eta'_c}^{\mathcal{B}[Y]})$ and $\mathcal{R}[Y](\llbracket N \rrbracket_{\eta_c}^{\mathcal{A}[Y]}, \llbracket N \rrbracket_{\eta'_c}^{\mathcal{B}[Y]})$. (We should assume that we have different environments for $\llbracket M \rrbracket_{\eta_c}^{\mathcal{A}[Y]}$ and $\llbracket N \rrbracket_{\eta_c}^{\mathcal{A}[Y]}$ and for $\llbracket M \rrbracket_{\eta'_c}^{\mathcal{B}[Y]}$ and $\llbracket N \rrbracket_{\eta'_c}^{\mathcal{B}[Y]}$, but without loss of generality we can assume that the $\mathcal{A}[Y]$ -environments are the same and similarly for $\mathcal{B}[Y]$.) We need to show that then $\mathcal{R}[Y](App_{\mathcal{A}[Y]} \llbracket M \rrbracket_{\eta_c}^{\mathcal{A}[Y]} \llbracket N \rrbracket_{\eta_c}^{\mathcal{A}[Y]}, App_{\mathcal{B}[Y]} \llbracket M \rrbracket_{\eta'_c}^{\mathcal{B}[Y]} \llbracket N \rrbracket_{\eta'_c}^{\mathcal{B}[Y]})$. We get the result by the definition of interpretation: $App_{\mathcal{A}[Y]} \llbracket M \rrbracket_{\eta_c}^{\mathcal{A}[Y]} \llbracket N \rrbracket_{\eta_c}^{\mathcal{A}[Y]} = \llbracket MN \rrbracket_{\eta_c}^{\mathcal{A}[Y]}$ and similarly for $\mathcal{B}[Y]$.

It remains to prove the opposite direction of the implication. Suppose we have $f \in \mathcal{A}[Y]$ and $g \in \mathcal{B}[Y]$ such that for every $\Sigma[Y]$ -term P and environments η, η' for \mathcal{A} and \mathcal{B} such that $R(\eta, \eta')$, $\mathcal{R}[Y](App_{\mathcal{A}[Y]} f \llbracket P \rrbracket_{\eta_c}^{\mathcal{A}[Y]}, App_{\mathcal{B}[Y]} g \llbracket P \rrbracket_{\eta'_c}^{\mathcal{B}[Y]})$. Then take P to be an indeterminate c_y which does not occur in \widehat{f} or \widehat{g} . (We need an infinite number of indeterminates to ensure that there is one that is different from all those that are already used in the terms \widehat{f} and \widehat{g} .) We know that there exists a $\Sigma[Y]$ -term M and environments ξ, ξ' for \mathcal{A} and \mathcal{B} such that $R(\xi, \xi')$, $App_{\mathcal{A}[Y]} f \llbracket c_y \rrbracket_{\xi_c}^{\mathcal{A}[Y]} = \llbracket M \rrbracket_{\xi_c}^{\mathcal{A}[Y]}$ and $App_{\mathcal{B}[Y]} g \llbracket c_y \rrbracket_{\xi'_c}^{\mathcal{B}[Y]} = \llbracket M \rrbracket_{\xi'_c}^{\mathcal{B}[Y]}$. Consider the $\Sigma[Y]$ -term $Q = \lambda c_y. M$. (Formally speaking, we should really take $\lambda x. M[x/c_y]$ so that we are abstracting a variable rather than a constant, but the meaning is clear.) We claim that f is extensionally equal to $\llbracket Q \rrbracket_{\xi_c}^{\mathcal{A}[Y]}$. Let $a \in \mathcal{A}[Y]$. Now consider the reduction sequences from $App_{\mathcal{A}[Y]} f \llbracket c_y \rrbracket_{\xi_c}^{\mathcal{A}[Y]}$ and $App_{\mathcal{A}[Y]} \llbracket \lambda c_y. M \rrbracket_{\xi_c}^{\mathcal{A}[Y]} \llbracket c_y \rrbracket_{\xi_c}^{\mathcal{A}[Y]}$ to a common reduct S . If

we replace all unbound occurrences of c_y in these reduction sequences with a , we obtain valid reduction sequences from $App_{\mathcal{A}[Y]} f a$ and $App_{\mathcal{A}[Y]} [\lambda c_y. M]_{\xi_c}^{\mathcal{A}[Y]} a$ to the common reduct $S[a/c_y]$. This works because c_y is “sterile”, with generic behaviour. Similarly, g is extensionally equal to $[[Q]_{\xi'_c}^{\mathcal{B}[Y]}]$. Therefore, since $\mathcal{A}[Y]$ and $\mathcal{B}[Y]$ are extensional, $\mathcal{R}[Y](f, g)$. ■

LEMMA A.10. *The composition of $emb_{\mathcal{A}}$, $\mathcal{R}[Y]$ and $(emb_{\mathcal{B}})^{-1}$ is R .*

Proof. It is easy to see that R is included in the composition: take M in Definition A.6 to be a fresh variable, then if $R(a, b)$, we have $\eta(x) = a$, $\eta_c(x) = c_a$, $\eta'_c(x) = c_b$, $\eta'(x) = b$. As for the converse, suppose $((emb_{\mathcal{B}})^{-1} \circ \mathcal{R}[Y] \circ emb_{\mathcal{A}})(a, b)$. By definition of $emb_{\mathcal{A}}$ there exists a Σ -term N and environment $\eta\mathcal{A}$ such that $a = [[N]_{\eta\mathcal{A}}^{\mathcal{A}}]$. By definition of $emb_{\mathcal{B}}$ there exists a Σ -term N' and environment $\eta\mathcal{B}$ such that $b = [[N']_{\eta\mathcal{B}}^{\mathcal{B}}]$. By definition of $\mathcal{R}[Y]$ there exists a $\Sigma[Y]$ -term M and environments η, η' such that $[[N]_{\eta\mathcal{A}_c}^{\mathcal{A}[Y]}] = [[M]_{\eta_c}^{\mathcal{A}[Y]}]$, $[[N']_{\eta\mathcal{B}_c}^{\mathcal{B}[Y]}] = [[M]_{\eta'_c}^{\mathcal{B}[Y]}]$ and $R(\eta, \eta')$. W.l.o.g. we can assume that $\eta\mathcal{A} = \eta$ and $\eta\mathcal{B} = \eta'$ and that M is in $\beta\eta$ -normal form. By Lemma A.5 we know that $[[M]_{\eta_c}^{\mathcal{A}[Y]}]$ and $[[M]_{\eta'_c}^{\mathcal{B}[Y]}]$ are $\Sigma\mathcal{A}$ - and $\Sigma\mathcal{B}$ -terms respectively, and that M contains no indeterminates. By Lemma A.4 we know that $a = [[N]_{\eta}^{\mathcal{A}}] = [[M]_{\eta}^{\mathcal{A}}]$ and $b = [[N']_{\eta'}^{\mathcal{B}}] = [[M]_{\eta'}^{\mathcal{B}}]$. So by the Basic Lemma, $R(a, b)$. ■