Behavioural Equivalence and Indistinguishability in Higher-Order Typed Languages

Shin-ya Katsumata

Division of Informatics, University of Edinburgh, King's Buildings,
Edinburgh EH9 3JZ, Scotland

Abstract. We extend the study of the relationship between *behavioural equivalence* and *the indistinguishability relation*[4, 7] to the simply typed lambda calculus, where higher-order types are available. The relationship between these two notions is established in terms of *factorisability*[4]. The main technical tool of this study is *pre-logical relations[8]*, which give a precise characterisation of behavioural equivalence. We then consider a higher-order logic to reason about models of the simply typed lambda calculus, and relate the resulting standard satisfaction relation to behavioural satisfaction.

1 Introduction

This work is a contribution to the understanding of the relationship between *behavioural equivalence* and the *indistinguishability relation*. These notions arose from the study of data abstraction in the context of algebraic specifications. Behavioural equivalence identifies models which show the same behaviour for any program yielding an observable value. This formalises an intuitive equivalence between two programming environments that show the same behaviour to programmers, regardless of differences in the representation of non-observable data types. The indistinguishability relation is a partial equivalence relation which identifies values in a model that are interchangeable with each other in any program context. This provides an abstract view of the programming environment based on behaviour, rather than denotation.

These two notions are useful when reasoning about specifications, and their relationship has been studied in a series of papers beginning with [4] by Bidoit, Hennicker and Wirsing. They established the key idea of *factorisability* to relate behavioural equivalence and the indistinguishability relation. Their framework is infinitary first-order logic over Σ algebras. Hofmann and Sannella[7] extended the logic over Σ algebras to higher-order logic, which enables us to quantify over predicates and axiomatise the indistinguishability relation when the underlying signature is finite.

We further extend the target of reasoning to a language having higher-order types and functions. Higher-order functions enable us to write program-parameterised programs, and are useful in program development. Thus we are interested in reasoning about specifications in such languages.

In this paper, we take the simply typed lambda calculus as the formalisation of higher-order typed languages, and give the semantics of the lambda calculus by *typed combinatory algebras*, which subsume a wide range of semantic frameworks including Henkin models, type frames and full-type hierarchies.

Once we introduce higher-order types, we need to consider how to extend behavioural equivalence and the indistinguishability relation to higher-order types. In the study of the simply typed lambda calculus, there is a well-known extension method using exponential relations (the following shows the case of binary relations between two combinatory algebras $\mathcal{A}$ and $\mathcal{B}$; we can extend this to n-ary relations):

$$R^{\tau\to\tau'} = R^{\tau} \to R^{\tau'} = \{(f,g) \in A^{\tau\to\tau'} \times B^{\tau\to\tau'} \mid \forall (x,y) \in R^{\tau} \,.\, (fx, gy) \in R^{\tau'}\}$$

and the resulting relation is called a *logical relation* if it relates the interpretations in $\mathcal{A}$ and $\mathcal{B}$ of each constant. However, in this study, logical relations are not adequate for a couple of reasons:

1. Reachability at first-order types cannot be extended to higher-order types using the exponential relation. We see this by an example; let us consider the simply typed lambda calculus with zero and successor, namely z^{nat} and $s^{nat\to nat}$. We give the semantics of the lambda calculus by the full-type hierarchy. A full-type hierarchy constructed from $A^{nat} = \mathbf{N}$. We write S^{τ} for the set of reachable elements at type τ. We can reach any $n \in \mathbf{N}$ by the term $s^n(z)$, thus $S^{nat} = A^{nat}$. However the unary logical relation R constructed from $R^{nat} = S^{nat}$ does not give reachability correctly at higher-order types, since $R^{nat\to nat} = A^{nat\to nat}$ but clearly $S^{nat\to nat} \subset A^{nat\to nat}$.
2. Logical relations are not suitable to characterise behavioural equivalence. A restricted notion of behavioural equivalence, called closed observational equivalence, was studied in [10]. Mitchell showed *representation independence theorem*; if there exists a binary logical relation between two models such that the relation is bijective on the observable types, then these two models are closed observationally equivalent. He showed that the converse is also true when the underlying signature has only first-order constants. However this is not satisfactory for two reasons; one is the above restriction to first-order constants, and the other is that in general logical relations do not compose, despite the fact that behavioural equivalence is a transitive relation.

To solve these problems, we use *pre-logical relations*[8] by Honsell and Sannella instead of logical relations. They are a generalisation of logical relation, and have several characterisations; a relation is pre-logical iff it satisfies the basic lemma (theorem 1 below), and a pre-logical relation can be seen as a correspondence in the sense of Schoett[12] between two combinatory algebras. Roughly, a pre-logical relation is a relation satisfying $R^{\tau\to\tau'} \subseteq R^{\tau} \to R^{\tau'}$. Thus pre-logical relations allow flexibility at higher-order types while logical relations are determined uniquely at all types from the relations at base types. Of course logical relations are included in pre-logical relations, but also the reachability predicate and other relations are included in this class. Another advantage of pre-logical relations is that they are closed under composition, which is a desirable property for characterising behavioural equivalence.

This paper is organised as follows: section 2 introduces basic definitions of the simply typed lambda calculus, pre-logical relations and partial equivalence relations(PERs). Section 3 establishes a relation between behavioural equivalence and existence of pre-logical relations. We also introduce another model equivalence and show that it is equiv-

alent to behavioural equivalence. In section 4 we study properties of the indistinguishability relation, which turns to be a pre-logical PER over the underlying model. In section 5 we show that behavioural equivalence is factorisable by indistinguishability. We move to higher-order logic and its semantics in section 6. We introduce two semantics; one is the standard model and the other is the relative model w.r.t. some PER. We show that the quotient model of higher-order logic by a PER and the behavioural model w.r.t. the PER are logically equivalent. We prove this by showing that they are behaviourally equivalent w.r.t. boolean observations. In section 7 we apply these results to reasoning about specifications.

2 Preliminaries

2.1 Syntax of the Simply Typed Lambda Calculus

Definition 1 (Higher-Order Signature). *Let U be a set. We define the set of types $\mathbf{Typ}(U)$ by BNF $\tau ::= b \mid \tau \to \tau$ where $b \in U$. A* higher-order signature *(or simply* signature*) is a pair of sets (U, C) where U gives the set of base types and $C \subseteq C_0 \times \mathbf{Typ}(U)$ gives the set of typed constants for some universe C_0 of constant symbols. We write c^τ for the pair $(c, \tau) \in C$. We fix a higher-order signature $\Sigma = (U, C)$. We often write $\mathbf{Typ}(\Sigma)$ for $\mathbf{Typ}(U)$.*

We assume that readers are familiar with the simply typed lambda calculus. The calculus considered in this paper is the minimal fragment; it has only $\to$ types. The lambda terms are built on a countably infinite set of variables X. We define a *context* by a partial function $\Gamma : X \rightharpoonup \mathbf{Typ}(\Sigma)$. Two contexts Γ and Δ are *separated* if $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Delta) = \emptyset$. For $T \subseteq \mathbf{Typ}(\Sigma)$, we say Γ is a *T-context* if for all $x \in \mathrm{dom}(\Gamma)$, $\Gamma(x) \in T$. We say $\Gamma \vdash M : \tau$ is a *well-formed term* if $\Gamma \vdash M : \tau$ is derived only from the inference rules of the simply typed lambda calculus.

2.2 Semantics of the Simply Typed Lambda Calculus

In this study, we take typed combinatory algebras as the basis for the semantics of the simply typed lambda calculus. The reason is twofold: one is that they are general enough to subsume other classes of models, such as Henkin models and type frames, and the other is that combinatory algebras and the notion of pre-logical relation, introduced later, are compatible. Indeed the class of combinatory algebras is closed under quotient by pre-logical PERs(proposition 2).

We write C_{SK} for the extension of a set of constants C with S, K combinators:

$$C_{SK} = C \cup \{S^{(\tau\to\tau'\to\tau'')\to(\tau\to\tau')\to\tau\to\tau''} \mid \tau, \tau', \tau'' \in \mathbf{Typ}(\Sigma)\} \cup \{K^{\tau\to\tau'\to\tau} \mid \tau, \tau' \in \mathbf{Typ}(\Sigma)\}.$$

Definition 2 (Typed Combinatory Algebra). *A Σ-*typed combinatory algebra *(or simply* combinatory algebra*) is a tuple $\mathcal{A} = (A, \bullet_{\mathcal{A}}, (-)_{\mathcal{A}})$ such that:*

1. *A is a $\mathbf{Typ}(\Sigma)$-indexed family of sets (called* carrier sets*).*

2. *The application operator* $\bullet_{\mathcal{A}}^{\tau,\tau'}$ *is a family of functions having type* $A^{\tau\to\tau'} \to A^{\tau} \to A^{\tau'}$ *for any* $\tau, \tau' \in \mathbf{Typ}(\Sigma)$.
3. $c_{\mathcal{A}}^{\tau} \in A^{\tau}$ *for each* $c^{\tau} \in C_{SK}$.
4. *The combinators satisfy equations* $K \bullet x \bullet y = x$ *and* $S \bullet p \bullet q \bullet r = p \bullet r \bullet (q \bullet r)$ *for any* x, y, p, q, r *in appropriate carrier sets (superscripts and subscripts are omitted for readability).*

We write $\bullet_{\mathcal{A}}$ *as a left-associative infix operator. We may omit superscripts and subscripts if they are obvious from the context. Script letters* $(\mathcal{A}, \mathcal{B}, \cdots)$ *are used to denote combinatory algebras while the carrier sets of these algebras are referred by normal letters* $(A, B, \cdots)$. *We write* $\mathbf{CA}(\Sigma)$ *for the collection of* Σ*-combinatory algebras.*

Definition 3 (Henkin Model/Type Frame/Full Type Hierarchy). *A combinatory algebra* $\mathcal{A}$ *is called:*

- *a* Σ-Henkin model *if* extensionality *holds:* $\forall f, g \in A^{\tau\to\tau'} . (\forall x \in A^{\tau} . f \bullet_{\mathcal{A}} x = g \bullet_{\mathcal{A}} x) \Longrightarrow f = g$.
- *a* Σ-type frame *if* $A^{\tau\to\tau'} \subseteq A^{\tau} \to A^{\tau'}$ *and* $f \bullet_{\mathcal{A}} x = f(x)$.
- *a* Σ-full type hierarchy *if* $A^{\tau\to\tau'} = A^{\tau} \to A^{\tau'}$ *and* $f \bullet_{\mathcal{A}} x = f(x)$. *We note that a full-type hierarchy is uniquely determined by the carrier sets for base types.*

Example 1. The following is a higher-order signature $\Sigma_{set} = (U_{set}, C_{set})$ for the finite sets of natural numbers.

$$U_{set} = \{bool, nat, set\}$$
$$C_{set} = \{\mathsf{tt}^{bool}, \mathsf{ff}^{bool}, \mathsf{not}^{bool\to bool}, 0^{nat}, \mathsf{succ}^{nat\to nat}, \mathsf{eq}^{nat\to nat\to bool},$$
$$\emptyset^{set}, \{-\}^{nat\to set}, (- \cup -)^{set\to set\to set}, \mathsf{filter}^{(nat\to bool)\to set\to set}, \mathsf{isempty}^{set\to bool}\}$$

The constant filter takes a predicate p and a set s and yields the set which consists of the elements in s satisfying the predicate p. The constant $\mathsf{isempty}$ judges whether a given set is empty or not.

We introduce two full-type hierarchies $\mathcal{A}_{set}$ and $\mathcal{B}_{set}$ over Σ_{set}. In $\mathcal{A}_{set}$, base types are interpreted as $A_{set}^{bool} = \{tt, ff\}, A_{set}^{nat} = \mathbf{N}, A_{set}^{set} = \mathcal{P}(\mathbf{N})$. We interpret filter and $\mathsf{isempty}$ in $\mathcal{A}_{set}$ as follows:

$$\mathsf{filter}_{\mathcal{A}_{set}}\ f\ X = \{x \in X \mid f(x) = tt\}$$
$$\mathsf{isempty}_{\mathcal{A}_{set}}\ X = tt \iff X = \emptyset$$

The interpretation of the other constants is naturally defined.

In B_{set}, base types are interpreted as $B_{set}^{bool} = \{tt, ff\}, B_{set}^{nat} = \mathbf{N}, B_{set}^{set} = \mathbf{N} \to B_{set}^{bool}$. In this interpretation, a set $X \subseteq \mathbf{N}$ is represented by its characteristic function $\phi_X : \mathbf{N} \to B_{set}^{bool}$. We interpret filter and $\mathsf{isempty}$ in $\mathcal{B}_{set}$ as follows:

$$(\mathsf{filter}_{\mathcal{B}_{set}}\ p\ f)\ x = tt \iff p(x) = tt \wedge f(x) = tt$$
$$\mathsf{isempty}_{\mathcal{B}_{set}}\ f = tt \iff \forall x \in \mathbf{N} . f(x) = ff$$

The interpretation of the other constants is naturally defined.

An *environment* (ranged over by η, ρ) over a combinatory algebra $\mathcal{A}$ is a partial function $X \rightharpoonup \bigcup_{\tau \in \mathbf{Typ}(\Sigma)} A^\tau$. We write $\eta \in A^\Gamma$ for an environment η such that $\mathrm{dom}(\eta) = \mathrm{dom}(\Gamma)$ and $\eta(x) \in A^{\Gamma(x)}$ for all $x \in \mathrm{dom}(\eta)$.

Given a combinatory algebra $\mathcal{A}$, we can interpret well-formed lambda terms in $\mathcal{A}$ by the *meaning function* $\mathcal{A}[\![-]\!]-$, which maps a well-formed term $\Gamma \vdash M : \tau$ and an environment $\eta \in A^\Gamma$ to a value in A^τ. The meaning function is defined by induction on the derivation of well-formed lambda terms, and uses a trick of compiling lambda abstraction using S and K in a combinatory algebra when interpreting $\lambda x^\tau \,.\, M$. For details, see [2, 11].

Proposition 1. (Semantic Substitution Lemma) *Let Γ and Δ be separated contexts, $\Gamma, x : \tau \vdash M : \tau'$ and $\Delta \vdash N : \tau$ be well-formed terms, $\rho \in A^\Gamma$ and $\eta \in A^\Delta$. Then $\mathcal{A}[\![M]\!]\rho\{x \mapsto \mathcal{A}[\![N]\!]\eta\} = \mathcal{A}[\![M[N/x]]\!]\rho \cup \eta$.*

Definition 4 (Σ-homomorphism). *A* Σ-homomorphism *is a $\mathbf{Typ}(\Sigma)$-indexed family of functions $\{h^\tau : A^\tau \to B^\tau\}_{\tau \in \mathbf{Typ}(\Sigma)}$ such that for all $c^\tau \in C_{SK}$, $h^\tau(c^\tau_{\mathcal{A}}) = c^\tau_{\mathcal{B}}$ and for all $\tau, \tau' \in \mathbf{Typ}(\Sigma), x \in A^{\tau \to \tau'}$ and $y \in A^\tau$, $h^{\tau'}(x \bullet_{\mathcal{A}} y) = h^{\tau \to \tau'}(x) \bullet_{\mathcal{B}} h^\tau(y)$. A* Σ-isomorphism *is a Σ-homomorphism h such that h^τ is bijective on each $\tau \in \mathbf{Typ}(\Sigma)$. We write $\mathcal{A} \cong \mathcal{B}$ if there exists a Σ-isomorphism between $\mathcal{A}$ and $\mathcal{B}$.*

2.3 Pre-logical Relations

First, some definitions. A *relation* between $\mathcal{A}$ and $\mathcal{B}$ (written $R \subseteq \mathcal{A} \times \mathcal{B}$) is a $\mathbf{Typ}(\Sigma)$-indexed family of sets R satisfying $R^\tau \subseteq A^\tau \times B^\tau$ for all $\tau \in \mathbf{Typ}(\Sigma)$. We write $(\eta, \eta') \in R^\Gamma$ if $\eta \in A^\Gamma, \eta' \in B^\Gamma$ and for all $x \in \mathrm{dom}(\Gamma)$, $(\eta(x), \eta'(x)) \in R^{\Gamma(x)}$. The composition of relations $R \subseteq \mathcal{A} \times \mathcal{B}$ and $R' \subseteq \mathcal{B} \times \mathcal{C}$ is defined by type-wise composition of R and R'. The *exponential relation* of R^τ and $R^{\tau'}$ is defined by

$$R^\tau \to R^{\tau'} = \{(f, g) \in A^{\tau \to \tau'} \times B^{\tau \to \tau'} \mid \forall (x, y) \in R^\tau \,.\, (f \bullet_{\mathcal{A}} x, g \bullet_{\mathcal{B}} y) \in R^{\tau'}\}.$$

Pre-logical relations were proposed by Honsell and Sannella[8], and are a generalised notion of logical relations. In this paper, we adopt the following definition of pre-logical relations.[1]

Definition 5 (Pre-logical Relations[8]). *A relation $R \subseteq \mathcal{A} \times \mathcal{B}$ is* pre-logical *if*

1. *$R^{\tau \to \tau'} \subseteq R^\tau \to R^{\tau'}$, or equivalently for all $(f, g) \in R^{\tau \to \tau'}$ and $(x, y) \in R^\tau$, the pair $(f \bullet_{\mathcal{A}} x, g \bullet_{\mathcal{B}} y)$ is included in $R^{\tau'}$, and*
2. *for all $c^\tau \in C_{SK}$, the pair $(c^\tau_{\mathcal{A}}, c^\tau_{\mathcal{B}})$ is included in R^τ.*

We contrast the above to the definition of logical relations. A logical relation is a type-indexed family of binary relations R satisfying $R^{\tau \to \tau'} = R^\tau \to R^{\tau'}$ and for each $c^\tau \in C$, $(c^\tau_{\mathcal{A}}, c^\tau_{\mathcal{B}}) \in R^\tau$. Thus when we give a logical relation, we perform the following

[1] Originally pre-logical relations were defined over lambda applicative structures, which is a general class of set-theoretic models of the simply typed lambda calculus. In the case that the underlying models are combinatory algebras, the definition coincides with definition 5 as observed in [8].

steps: we first give a relation R on base types, then extend it to higher-order types using the above scheme and check whether the interpretations in $\mathcal{A}$ and $\mathcal{B}$ of each of the constants are related by R. In contrast, the definition of pre-logical relations lacks right-to-left inclusion in the above scheme.[2] Thus it allows flexibility of choice of relations at higher-order types. We note that logical relations are also pre-logical relations, since the above scheme implies that the relation R relates S, K combinators at all types.

In [8] various characterisation of pre-logical relations are studied. One notable characterisation is via the basic lemma.

Theorem 1 (Basic Lemma for Pre-logical Relations[8]). *Let $R \subseteq \mathcal{A} \times \mathcal{B}$. Then $\forall(\eta, \eta') \in R^{\Gamma}.(\mathcal{A}[\![M]\!]\eta, \mathcal{B}[\![M]\!]\eta') \in R^{\tau}$ holds for any well-formed term $\Gamma \vdash M : \tau$ if and only if R is a pre-logical relation.*

Another notable property of pre-logical relations is that the composition of two pre-logical relations is again a pre-logical relation.

Theorem 2 (Composability of Pre-logical Relations[8]). *Let $R \subseteq \mathcal{A} \times \mathcal{B}$ and $R' \subseteq \mathcal{B} \times \mathcal{C}$ be pre-logical relations. Then $R \circ R' \subseteq \mathcal{A} \times \mathcal{C}$ is a pre-logical relation.*

2.4 Partial Equivalence Relations

Recall that a *PER* (ranged over by E) over $\mathcal{A}$ is a relation $E \subseteq \mathcal{A} \times \mathcal{A}$ such that for all $\tau \in \mathbf{Typ}(\Sigma)$, E^{τ} is symmetric and transitive. We write the domain of E^{τ} by $|E^{\tau}| = \{x \in A^{\tau} \mid (x, x) \in E^{\tau}\}$. Then E^{τ} is just an equivalence relation over $|E^{\tau}|$, so we write $[x]$ for the equivalence class of $x \in |E^{\tau}|$ by E^{τ} and A/E^{τ} for the quotient $|E^{\tau}|/E^{\tau}$.

When a PER $E \subseteq \mathcal{A} \times \mathcal{A}$ is pre-logical (or logical), we call E a *pre-logical (or logical) PER*. The quotient of a combinatory algebra by a pre-logical PER is again a combinatory algebra.

Proposition 2 ([8]). *Let E be a pre-logical PER over $\mathcal{A}$.*

1. *The tuple $(A/E, \star, [(-)_{\mathcal{A}}])$ where $[x] \star [y] = [x \bullet_{\mathcal{A}} y]$ is a Σ-combinatory algebra. We call this the* quotient *of $\mathcal{A}$ by E, and write it by $\mathcal{A}/E$.*
2. *Let $\Gamma \vdash M : \tau$ be a well-formed term and $\eta \in A/E^{\Gamma}$. Then $\mathcal{A}/E[\![M]\!]\eta = [\mathcal{A}[\![M]\!]\rho]$ where $\rho \in A^{\Gamma}$ and $\rho(x) \in \eta(x)$ for all $x \in \mathrm{dom}(\Gamma)$.*

Definition 6 (Projection). *We define the* projection relation $\Pi(E) \subseteq \mathcal{A}/E \times \mathcal{A}$ *as the following $\mathbf{Typ}(\Sigma)$-indexed family of binary relations:*

$$\Pi(E)^{\tau} = \{([e], e) \in A/E^{\tau} \times A^{\tau} \mid e \in |E^{\tau}|\}.$$

Lemma 1. *The projection $\Pi(E)$ is a pre-logical relation.*

Proof. Clearly $\Pi(E)$ relates all constants in C_{SK}. From the definition of pre-logical PERs, E is closed under the application operator. Therefore $\Pi(E)$ is so as well. □

[2] Indeed, the reverse direction is required to hold only for lambda-definable elements because of the presence of the combinators in the set of constants.

3 Behavioural Equivalence and Pre-logical Relations

Behavioural equivalence identifies two models showing the same behaviour in response to all observations. Each observation compares the values of two terms of observable types, whose values are directly accessible to programmers. This definition of observation formalises the use of experiments to detect the difference of behaviour of visible data types between two models. Thus, intuitively speaking, if two models are behaviourally equivalent, they provide the same programming environment to programmers, even though they may have different implementations of invisible data types.

We establish a link between behavioural equivalence and pre-logical relations. The result is a natural extension of [8] to allow free variables of observable types, and an extension of [12] to handle higher-order types.

The definition of behavioural equivalence is adapted from [7]. There are other possibilities for the treatment of free variables, but we do not discuss them. For detail, see [7]. We fix a set $OBS \subseteq \mathbf{Typ}(\Sigma)$ called the *observable types*. We first introduce an auxiliary notion of OBS-surjective environment.

Definition 7. *An* OBS-surjective environment between $\mathcal{A}$ and $\mathcal{B}$ *is a tuple* (Γ, ρ, ρ') *where* Γ *is an* OBS*-context and* $\rho \in A^\Gamma$ *and* $\rho' \in B^\Gamma$ *are environments such that* $\mathrm{im}(\rho) = \bigcup_{\tau \in OBS} A^\tau$ *and* $\mathrm{im}(\rho') = \bigcup_{\tau \in OBS} B^\tau$.

We say $\mathcal{A}$ *is* OBS*-countable if the set* $\bigcup_{\tau \in OBS} A^\tau$ *is countable.*

We note that if there exists an OBS-surjective environment between $\mathcal{A}$ and $\mathcal{B}$, then $\mathcal{A}$ and $\mathcal{B}$ are OBS-countable. This is due to the cardinality of the set of variables.

Definition 8 (Behavioural Equivalence). *We say* $\mathcal{A}$ *and* $\mathcal{B}$ *are* behaviourally equivalent *w.r.t.* OBS *(written* $\mathcal{A} \equiv_{OBS} \mathcal{B}$*) if there exists an* OBS*-surjective environment* (Γ, ρ, ρ') *between* $\mathcal{A}$ *and* $\mathcal{B}$ *such that for any* $\tau \in OBS$ *and well-formed terms* $\Gamma \vdash M, N : \tau$*, we have* $\mathcal{A}[\![M]\!]\rho = \mathcal{A}[\![N]\!]\rho \iff \mathcal{B}[\![M]\!]\rho' = \mathcal{B}[\![N]\!]\rho'$.

We also give another formalisation of behavioural equivalence. We first introduce a program equivalence in a model, then we say two models are behaviourally equivalent if the program equivalence in both models coincides.

Definition 9.
1. *Let* Γ *be an* OBS*-context,* $\tau \in OBS$ *and* $\Gamma \vdash M, N : \tau$ *be well-formed terms. We write* $\mathcal{A} \models \Gamma \vdash M \sim N : \tau$ *if for all* $\eta \in A^\Gamma$*, we have* $\mathcal{A}[\![M]\!]\eta = \mathcal{A}[\![N]\!]\eta$.
2. *We write* $\mathcal{A} \sim_{OBS} \mathcal{B}$ *if* $A^\tau \cong B^\tau$ *for any* $\tau \in OBS$*, and for any* OBS*-context* Γ*,* $\tau \in OBS$ *and well-formed terms* $\Gamma \vdash M, N : \tau$*, we have* $\mathcal{A} \models \Gamma \vdash M \sim N : \tau \iff \mathcal{B} \models \Gamma \vdash M \sim N : \tau$.

We introduce *observational pre-logical relations* to characterise behavioural equivalence (c.f. Schoett's correspondence[12]).

Definition 10 (Observational Pre-logical Relations). *An* observational pre-logical relation $R \subseteq \mathcal{A} \times \mathcal{B}$ *w.r.t.* OBS *is a pre-logical relation such that for all* $\tau \in OBS$*,* $R^\tau \subseteq A^\tau \times B^\tau$ *is a bijection.*

Proposition 3. *Let* $R \subseteq \mathcal{A} \times \mathcal{B}$ *and* $S \subseteq \mathcal{B} \times \mathcal{C}$ *be observational pre-logical relations w.r.t.* OBS*. Then* $R \circ S$ *is an observational pre-logical relation w.r.t.* OBS.

The following theorem characterises behavioural equivalence in terms of observational pre-logical relations. Simultaneously, it shows that the two formalisations of behavioural equivalence coincide.[3]

Theorem 3. *The following are equivalent:*[4]

1. $\mathcal{A} \equiv_{OBS} \mathcal{B}$.
2. *$\mathcal{A}$ and $\mathcal{B}$ are OBS-countable and* $A \sim_{OBS} \mathcal{B}$.
3. *$\mathcal{A}$ and $\mathcal{B}$ are OBS-countable and there is an observational pre-logical relation* $R \subseteq \mathcal{A} \times \mathcal{B}$ *w.r.t.* OBS.

Proof. (Sketch) ($1 \Longrightarrow 2$) The assumption implies that there exists an OBS-surjective environment (Γ, ρ, ρ'). From this, $\mathcal{A}$ and $\mathcal{B}$ are OBS-countable. We then define for each $\tau \in OBS$, $h^\tau : A^\tau \to B^\tau$ by $h^\tau(a) = \rho'(x_a)$, where $x_a \in \mathrm{dom}(\Gamma)$ is a variable such that $\rho(x_a) = a$. We can show h^τ is well-defined and gives an isomorphism. Next we show $\mathcal{B} \models \Delta \vdash M \sim N : \tau$ implies $\mathcal{A} \models \Delta \vdash M \sim N : \tau$; the reverse direction is by symmetry. Let $\eta' \in B^\Delta$. From the definition of OBS-surjective environment, for all $x \in \mathrm{dom}(\Delta)$, there exists $y_x \in \mathrm{dom}(\Gamma)$ such that $\eta'(x) = \rho'(y_x)$. Then we define an environment $\eta \in A^\Delta$ by $\eta(x) = \rho(y_x)$ and a variable renaming σ by $\sigma(x) = y_x$. Now we have $\mathcal{A}[\![M\sigma]\!]\rho = \mathcal{A}[\![M]\!]\eta = \mathcal{A}[\![N]\!]\eta = \mathcal{A}[\![N\sigma]\!]\rho$. This implies $\mathcal{B}[\![M]\!]\eta' = \mathcal{B}[\![M\sigma]\!]\rho' = \mathcal{B}[\![N\sigma]\!]\rho' = \mathcal{B}[\![N]\!]\eta'$ from $\mathcal{A} \equiv_{OBS} \mathcal{B}$.

($2 \Longrightarrow 3$) From $\mathcal{A} \sim_{OBS} \mathcal{B}$, we can choose bijections R_0^τ for each $\tau \in OBS$ satisfying $(\mathcal{A}[\![M]\!], \mathcal{B}[\![M]\!]) \in R_0^\tau$ for all $\emptyset \vdash M : \tau$. Then it is easy to see that the following relation $R \subseteq \mathcal{A} \times \mathcal{B}$ is an pre-logical relation:

$$R^\tau = \{(\mathcal{A}[\![M]\!]\rho, \mathcal{B}[\![M]\!]\rho') \mid \Gamma \text{ is an } OBS\text{-context } \wedge\ \Gamma \vdash M : \tau\ \wedge\ (\rho, \rho') \in R_0^\Gamma\}$$

where R^τ is clearly a bijection for each $\tau \in OBS$.

($3 \Longrightarrow 1$) Since $\mathcal{A}$ and $\mathcal{B}$ are OBS-countable, for each $\tau \in OBS$ and each pair $(e, f) \in R^\tau$, it is possible to assign a distinct variable $x^\tau_{e,f}$. Then we define an OBS-surjective environment (Γ, ρ, ρ') by $\Gamma(x^\tau_{e,f}) = \tau$, $\rho(x^\tau_{e,f}) = e$ and $\rho'(x^\tau_{e,f}) = f$. The goal $\mathcal{A} \equiv_{OBS} \mathcal{B}$ follows from lemma 1. □

Example 2. We construct a logical relation $R_{set} \subseteq \mathcal{A}_{set} \times \mathcal{B}_{set}$ from the following relations at base types:

$$R^{bool} = \mathrm{Id}^{bool}, \quad R^{nat} = \mathrm{Id}^{nat},$$
$$R^{set} = \{(X, \phi) \in \mathcal{P}(\mathbf{N}) \times (\mathbf{N} \to B^{bool}) \mid \forall x\,.\, x \in X \iff \phi(x) = tt\}$$

We can easily show that R relates the interpretation of all constants, and by definition, it is bijective on $\{bool, nat\}$. Therefore we have $\mathcal{A}_{set} \equiv_{\{bool,nat\}} \mathcal{B}_{set}$.

Example 3. In [9], the notion of constructive data refinement is formalised in terms of the existence of a pre-logical relation. They demonstrate that an implementation δ of

[3] The proof of theorem 3 does not rely on particular properties of combinatory algebras. Thus we can expect that it holds over lambda applicative structures.

[4] In fact $2 \iff 3$ still holds when dropping the condition that $\mathcal{A}$ and $\mathcal{B}$ are OBS-countable.

real number computation in the programming language PCF forms a data refinement in their sense; for any model $\mathcal{B}$ of PCF, there exists a model $\mathcal{A}$ of real number computation such that $\mathcal{A}$ and $\mathcal{B}|_\delta$ (the δ-reduct of $\mathcal{B}$) are closed observationally equivalent w.r.t. $bool$. To show this, they give an actual construction of $\mathcal{A}$ and $R \subseteq \mathcal{A} \times \mathcal{B}|_\delta$ from any PCF model $\mathcal{B}$, where R is a pre-logical relation but not a logical relation. For details, see [9]. We believe that we can replace closed observational equivalence with behavioural equivalence and we can still construct a model $\mathcal{A}$ such that $\mathcal{A} \equiv_{\{bool\}} \mathcal{B}|_\delta$.

4 Indistinguishability Relations

We introduce an equivalence of values called indistinguishability based on their behaviour rather than their denotation. We regard two values in a model as "behaviourally" indistinguishable if they are interchangeable in any program. This is shown by performing a set of experiments; we fit one value into a program yielding a visible result, and see whether any difference is detected when we exchange the one with the other. If two values pass the above experiment over all possible programs, then we say that they are indistinguishable. This identification of values is more suitable to provide an abstract aspect of specifications.

There are several ways to formalise the above idea. In this paper we adopt the same definition of indistinguishability as [7] for combinatory algebras.

Definition 11 (Reachable). *Let $\tau \in \mathbf{Typ}(\Sigma)$. A value $v \in A^\tau$ is OBS-reachable if there exists an OBS-context Γ, a well-formed term $\Gamma \vdash M : \tau$ and $\rho \in A^\Gamma$ such that $v = \mathcal{A}[\![M]\!]\rho$.*

Definition 12 (Indistinguishability Relation). *Let $\tau \in \mathbf{Typ}(\Sigma)$. We say two values $v, w \in A^\tau$ are* indistinguishable *(written $v \approx^\tau_{\mathcal{A}} w$) if they are OBS-reachable and for any OBS-context Γ, $\tau' \in OBS$, $\rho \in A^\Gamma$ and well-formed term $\Gamma, x : \tau \vdash M : \tau'$, we have $\mathcal{A}[\![M]\!]\rho\{x \mapsto v\} = \mathcal{A}[\![M]\!]\rho\{x \mapsto w\}$.*

The indistinguishability relation is defined on each combinatory algebra. Thus $\approx$ gives rise to a family of PERs indexed by $\mathbf{CA}(\Sigma)$. The results in this section are proved for only one combinatory algebra, but readers may regard them as statements for the family of indistinguishability PERs.

In $\mathcal{A}_{set}$, $x \approx^{set}_{\mathcal{A}_{set}} y$ implies $x = y$. We note that $\approx^{set}_{\mathcal{A}_{set}}$ is a partial equivalence relation but not a total one since infinite sets of natural numbers are not OBS-reachable.

Theorem 4. *The indistinguishability relation $\approx_{\mathcal{A}}$ is a pre-logical PER such that $\approx^\tau_{\mathcal{A}} = \mathrm{Id}_{A^\tau}$ for all $\tau \in OBS$.*

Proof. (Sketch) It is easy to see that $\approx_{\mathcal{A}}$ is a PER which relates all constants and is Id^τ for all $\tau \in OBS$. We show it is closed under application. We assume $e \approx^{\tau \to \tau'}_{\mathcal{A}} f$ and $x \approx^\tau_{\mathcal{A}} y$. Let Γ be an OBS-context, $\tau'' \in OBS$, $\Gamma, z : \tau' \vdash M : \tau''$ be a well-formed term and $\rho \in A^\Gamma$. Since e is OBS-reachable, we can write $e = \mathcal{A}[\![E]\!]\eta$ with a well-formed term E and an environment such that $\mathrm{dom}(\rho) \cap \mathrm{dom}(\eta) = \emptyset$. Then from proposition 1 and $x \approx^\tau_{\mathcal{A}} y$, we have $\mathcal{A}[\![M]\!]\rho\{z \mapsto e \bullet x\} = \mathcal{A}[\![M[Ew/z]]\!](\rho \cup \eta)\{w \mapsto x\} = \mathcal{A}[\![M]\!]\rho\{z \mapsto e \bullet y\}$. We can similarly swap e and f. Thus we obtain $e \bullet x \approx^{\tau'}_{\mathcal{A}} f \bullet y$. □

By analogy with the terminology of denotational semantics, a Σ-combinatory algebra is *fully abstract* when the indistinguishability relation on $\mathcal{A}$ and the set-theoretic equality coincide (the proof is omitted).

Theorem 5. *The quotient model $\mathcal{A}/{\approx_{\mathcal{A}}}$ is $\approx$-fully abstract, i.e. for all $\tau \in \mathbf{Typ}(\Sigma)$ and $a, b \in (A/{\approx_{\mathcal{A}}})^{\tau}$, $a = b$ iff $a \approx^{\tau}_{\mathcal{A}/{\approx_{\mathcal{A}}}} b$.*

5 Factorisability

We have seen two approaches to obtain abstract models of specifications; behavioural equivalence on the one hand and indistinguishability relation on the other hand. Both of them naturally arise from the motivation of reasoning about specifications from a behavioural point of view. Thus we are interested in considering their relationship. The key idea is the notion of factorisability[4].

Definition 13 (Factorisability). *Let E be a $\mathbf{CA}(\Sigma)$-indexed family of PERs and $\equiv$ be an equivalence relation over $\mathbf{CA}(\Sigma)$. Then*

- *$\equiv$ is* left-factorisable *by E if for all $\mathcal{A}, \mathcal{B} \in \mathbf{CA}(\Sigma)$, $\mathcal{A}/E_{\mathcal{A}} \cong \mathcal{B}/E_{\mathcal{B}} \Longrightarrow \mathcal{A} \equiv \mathcal{B}$.*
- *$\equiv$ is* right-factorisable *by E if for all $\mathcal{A}, \mathcal{B} \in \mathbf{CA}(\Sigma)$, $\mathcal{A} \equiv \mathcal{B} \Longrightarrow \mathcal{A}/E_{\mathcal{A}} \cong \mathcal{B}/E_{\mathcal{B}}$.*

We say $\equiv$ is factorisable *by E if both of the above hold.*

In this section we show that behavioural equivalence is factorisable by the indistinguishability relation. First we prove left-factorisabilty.

Theorem 6 (Left-Factorisability). $\mathcal{A}/{\approx_{\mathcal{A}}} \cong \mathcal{B}/{\approx_{\mathcal{B}}} \Longrightarrow \mathcal{A} \equiv_{OBS} \mathcal{B}$.

Proof. $\mathcal{A}/{\approx_{\mathcal{A}}} \cong \mathcal{B}/{\approx_{\mathcal{B}}}$ implies $\mathcal{A}/{\approx_{\mathcal{A}}} \equiv_{OBS} \mathcal{B}/{\approx_{\mathcal{B}}}$. From theorem 4, we have $\mathcal{A}/{\approx_{\mathcal{A}}} \equiv_{OBS} \mathcal{A}$ and $\mathcal{B}/{\approx_{\mathcal{B}}} \equiv_{OBS} \mathcal{B}$. Thus $\mathcal{A} \equiv_{OBS} \mathcal{B}$ by transitivity. □

In [7], Hofmann and Sannella represented the indistinguishability relation and the "experiments" for behavioural equivalence in a higher-order logic, then showed that the satisfiability of the experiments coincide in each model when quotients of two models are isomorphic. However this approach seems not to work in this paper, since their method depends on the finiteness of specifications to represent the indistinguishability relation, while combinatory algebras have a countably infinite number of types and S, K-combinators.

The proof of right-factorisability is essentially the same as the one in [7].

Theorem 7 (Right-Factorisability). $\mathcal{A} \equiv_{OBS} \mathcal{B} \Longrightarrow \mathcal{A}/{\approx_{\mathcal{A}}} \cong \mathcal{B}/{\approx_{\mathcal{B}}}$.

Proof. (Sketch) From $\mathcal{A} \equiv_{OBS} \mathcal{B}$, there is an observational pre-logical relation $R \subseteq \mathcal{A} \times \mathcal{B}$ w.r.t. OBS. Now we define a relation $h \subseteq \mathcal{A}/{\approx_{\mathcal{A}}} \times \mathcal{B}/{\approx_{\mathcal{B}}}$:

$$h^{\tau} = \{(\mathcal{A}/{\approx_{\mathcal{A}}}[\![M]\!]\rho, \mathcal{B}/{\approx_{\mathcal{B}}}[\![M]\!]\rho') \mid \Gamma \text{ is an } OBS\text{-context} \wedge \Gamma \vdash M : \tau \wedge (\rho, \rho') \in R^{\Gamma}\}$$

We can show that h gives a partial injection in both directions. Moreover, all elements in $\mathcal{A}/{\approx_{\mathcal{A}}}$ and $\mathcal{B}/{\approx_{\mathcal{B}}}$ are OBS-reachable. Therefore h^{τ} is total and surjective, i.e. is bijective for each $\tau \in \mathbf{Typ}(\Sigma)$. It is easy to see that h is a Σ-isomorphism. □

6 Higher-Order Logic to Reason About Higher-Typed Languages

We consider a higher-order logic to reason about specifications in the higher-order typed languages. We introduce two models of the higher-order logic; one is the standard model, which equates two programs when they have the same denotations, and the other is the behavioural model, which equates two programs when they have the same behaviour. The latter model is useful when we reason about specifications based on behaviour of programs. We then relate standard satisfaction and behavioural satisfaction.

6.1 Syntax

The higher-order logic considered in this section is designed to reason about combinatory algebras over a signature Σ. Thus the logic has constants for the application operator $\bullet$ and S, K-combinators corresponding to those in combinatory algebras.

The syntax of the higher-order logic can be formalised in the framework of the simply typed lambda calculus—it is just a lambda calculus over a certain signature (which is an extension of Σ) providing a type of propositions and constants for logical connectives.

Although we can reuse definitions, syntax and terminology of the simply typed lambda calculus, we re-define them for higher-order logic to make it clear which calculus we are talking about. We use a different function type symbol $\Rightarrow$ instead of $\rightarrow$, and write $\mathbf{Typ}^{\Rightarrow}(U)$ for the set defined by BNF $\phi ::= b \mid \phi \Rightarrow \phi$ where $b \in U$.

Definition 14 (Higher-Order Logic). *The syntax of higher-order logic over Σ is given by the syntax of the simply typed lambda calculus over the signature $\Sigma_{HOL} = (U_{HOL}, C_{HOL})$ defined by:*

$$
\begin{aligned}
U_{HOL} &= \{\mathbf{\Omega}\} \cup \mathbf{Typ}(\Sigma) \\
C_{HOL} &= \{\supset^{\mathbf{\Omega}\Rightarrow\mathbf{\Omega}\Rightarrow\mathbf{\Omega}}\} \cup \{=^{\phi\Rightarrow\phi\Rightarrow\mathbf{\Omega}} \mid \phi \in \mathbf{Typ}^{\Rightarrow}(U_{HOL})\} \\
&\quad \cup \{\bullet^{\tau\rightarrow\tau'\Rightarrow\tau\Rightarrow\tau'} \mid \tau, \tau' \in \mathbf{Typ}(\Sigma)\} \cup C_{SK}
\end{aligned}
$$

We may omit types of constants in superscripts if they are obvious from the context. The constants $\supset, =$ and $\bullet$ are used as infix operators.

We call types of Σ_{HOL} *formula types* (ranged over by ϕ) and terms of Σ_{HOL} *formulas* (ranged over by F). In this logic, a lambda term M is represented by a formula $M^{\bullet}_{CL}$, which is a combinatorial representation of M by constants in C_{SK} and $\bullet^{\tau\rightarrow\tau'\Rightarrow\tau\Rightarrow\tau'}$.

Lambda abstraction plus logical constants $\supset$ and $=$ are powerful enough to derive other familiar logical constants such as $\mathtt{tt}, \mathtt{ff}, \neg, \wedge, \vee, \neq$ and quantifiers $\forall x : \phi \,.\, F$ and $\exists x : \phi \,.\, F$. See [1] for the definition and the axioms for the logical constants.

Example 4. The higher-order logic considered in this section is dedicated to reasoning about the combinatory algebras providing the semantics of the lambda calculus. Thus the higher-order logic has the axiom schema for S and K combinators (see definition 2). We may need to add extra axioms depending on the properties of the underlying combinatory algebra. If one assumes that it is a Henkin model, one adds the axiom scheme of extensionality: $\forall x, y : \tau \rightarrow \tau' \,.\, (\forall z : \tau \,.\, x \bullet z = y \bullet z) \supset x = y$.

In the higher-order logic over Σ_{set}, we would like to assume the induction principle for type nat:

$$\forall p : nat \Rightarrow \mathbf{\Omega} \,.\, p\,0 \supset (\forall x : nat \,.\, p\,x \supset p(\mathsf{succ} \bullet x)) \supset \forall x : nat \,.\, p\,x.$$

We can also specify the behaviour of constants, like

$$\mathsf{isempty} \bullet \emptyset = tt$$
$$\forall p, q : nat \to bool, s : set \,.\, \mathsf{filter} \bullet p \bullet (\mathsf{filter} \bullet q \bullet s) = \mathsf{filter} \bullet q \bullet (\mathsf{filter} \bullet p \bullet s)$$

Proof systems for the higher-order logic and its soundness and completeness are not covered in this paper. For details see [6].

6.2 Standard Satisfaction and Behavioural Satisfaction

We apply the semantic framework of the simply typed lambda calculus to give semantics to the higher-order logic. A model of the higher-order logic, say $\mathcal{M}$, is built on top of a combinatory algebra $\mathcal{A}$. Constant symbols for combinatory algebras such as S, K and $\bullet$ are interpreted by the corresponding elements $S_{\mathcal{A}}, K_{\mathcal{A}}$ and $\bullet_{\mathcal{A}}$. Do not confuse the underlying combinatory algebra $\mathcal{A}$ and the model $\mathcal{M}$ of the higher-order logic.

We introduce two models. One is the *standard model* which interprets $\mathbf{\Omega}$ as the two-point set $\mathbf{2} = \{tt, ff\}$, the function type $\Rightarrow$ as the set-theoretic function space, $\supset$ as the (curried form of) boolean implication and $=$ as the (curried form of) characteristic function of set-theoretic equality.

Definition 15 (Standard Model). *The* standard model $\mathcal{L}_{\mathcal{A}}$ *of the higher-order logic over $\mathcal{A}$ is a Σ_{HOL}-full type hierarchy over $L^{\mathbf{\Omega}}_{\mathcal{A}} = \mathbf{2}$ and $L^{\tau}_{\mathcal{A}} = A^{\tau}$ together with the following interpretation of the logical constants:*

$$\begin{aligned}
\supset^{\mathbf{\Omega}\Rightarrow\mathbf{\Omega}\Rightarrow\mathbf{\Omega}}_{\mathcal{L}_{\mathcal{A}}}(x)(y) = tt &\iff (x = tt \Longrightarrow y = tt)\\
=^{\phi\Rightarrow\phi\Rightarrow\mathbf{\Omega}}_{\mathcal{L}_{\mathcal{A}}}(x)(y) = tt &\iff x = y\\
\bullet^{\tau\to\tau'\Rightarrow\tau\Rightarrow\tau'}_{\mathcal{L}_{\mathcal{A}}}(x)(y) &= x \bullet_{\mathcal{A}} y\\
c^{\tau}_{\mathcal{L}_{A}} &= c^{\tau}_{\mathcal{A}} \quad (c^{\tau} \in C_{SK}).
\end{aligned}$$

We say that a closed formula $F : \mathbf{\Omega}$ is satisfied *(written $\mathcal{A} \models F$) if $\mathcal{L}_{\mathcal{A}}[\![F]\!] = tt$.*

The other model is the *behavioural model* w.r.t. a pre-logical PER E over $\mathcal{A}$. The standard model is not appropriate when we want to reason about specifications up to their behaviour rather than their denotation. This is because the equality may distinguish two different denotations even though they have the same behaviour. The behavioural model solves this problem by interpreting each predicate type ϕ as $|\overline{E}^{\phi}|$ where $\overline{E}$ is the extension of E to all predicate types using the exponential relation, and the equality as the equivalence relation $\overline{E}$ over $|\overline{E}|$. In particular, E is often taken as the indistinguishability relation over $\mathcal{A}$ (see theorem 4).

Definition 16 (Behavioural Model). *Given a pre-logical PER E over $\mathcal{A}$, we define a PER $\overline{E}$ over carrier sets $L_{\mathcal{A}}$ as follows:*

$$\overline{E}^{\mathbf{\Omega}} = \mathrm{Id}_{\mathbf{2}},\ \overline{E}^{\tau} = E^{\tau},\ \overline{E}^{\phi \Rightarrow \phi'} = \overline{E}^{\phi} \to \overline{E}^{\phi'}$$

Then the behavioural model $\mathcal{L}^E_{\mathcal{A}}$ *of the higher-order logic over $\mathcal{A}$ with respect to E is given by a Σ_{HOL}-type frame $(L^E_{\mathcal{A}}, (-)_{\mathcal{L}^E_{\mathcal{A}}})$ where $(L^E_{\mathcal{A}})^{\phi} = |\overline{E}^{\phi}|$ and $(-)_{\mathcal{L}^E_{\mathcal{A}}}$ gives the interpretation of the logical constants as follows:*

$$\begin{aligned}
\supset^{\mathbf{\Omega} \Rightarrow \mathbf{\Omega} \Rightarrow \mathbf{\Omega}}_{\mathcal{L}^E_{\mathcal{A}}}(x)(y) = tt &\iff (x = tt \Longrightarrow y = tt)\\
=^{\phi \Rightarrow \phi \Rightarrow \mathbf{\Omega}}_{\mathcal{L}^E_{\mathcal{A}}}(x)(y) = tt &\iff (x, y) \in \overline{E}^{\phi}\\
\bullet^{\tau \to \tau' \Rightarrow \tau \Rightarrow \tau'}_{\mathcal{L}^E_{\mathcal{A}}}(x)(y) &= x \bullet_{\mathcal{A}} y\\
c^{\tau}_{\mathcal{L}^E_{\mathcal{A}}} &= c^{\tau}_{\mathcal{A}} \quad (c^{\tau} \in C_{SK}).
\end{aligned}$$

We say that a closed formula $F : \mathbf{\Omega}$ is satisfied w.r.t. *E (written $\mathcal{A} \models^E F$) if $\mathcal{L}^E_{\mathcal{A}}[\![F]\!] = tt$.*

We show that the standard model over $\mathcal{A}/E$ and the behavioural model over $\mathcal{A}$ w.r.t. E satisfy the same formula (c.f. theorem 3.35 in [7]). We notice that this is implied by $\mathcal{L}_{\mathcal{A}/E} \equiv_{\{\mathbf{\Omega}\}} \mathcal{L}^E_{\mathcal{A}}$, since from $\mathcal{L}_{\mathcal{A}/E} \equiv_{\{\mathbf{\Omega}\}} \mathcal{L}^E_{\mathcal{A}}$, for all closed formula F, we have $\mathcal{L}_{\mathcal{A}/E}[\![F]\!] = \mathcal{L}_{\mathcal{A}/E}[\![\mathbb{tt}]\!] = tt \iff \mathcal{L}^E_{\mathcal{A}}[\![F]\!] = \mathcal{L}^E_{\mathcal{A}}[\![\mathbb{tt}]\!] = tt$.

Theorem 8. $\mathcal{L}_{\mathcal{A}/E} \equiv_{\{\mathbf{\Omega}\}} \mathcal{L}^E_{\mathcal{A}}$.

Proof. (Sketch) We construct an observational pre-logical relation $R \subseteq \mathcal{L}_{\mathcal{A}/E} \times \mathcal{L}^E_{\mathcal{A}}$ w.r.t. $\{\mathbf{\Omega}\}$ by theorem 3. First we show that there is a family of isomorphisms $h^{\phi} : L^{\phi}_{\mathcal{A}/E} \cong L^{\phi}_{\mathcal{A}}/\overline{E}^{\phi}$ for the carrier sets of $\mathcal{L}_{\mathcal{A}/E}$. Let $I^{\phi} \subseteq L^{\phi}_{\mathcal{A}} \times |\overline{E}^{\phi}|$ be the inclusion relation. Then the pre-logical relation in question is given by the composition relation $h^{\phi} \circ \Pi(\overline{E})^{\phi} \circ I^{\phi} \subseteq L^{\phi}_{\mathcal{A}/E} \times |\overline{E}^{\phi}|$. □

Corollary 1. *For all closed formula $F : \mathbf{\Omega}$, $\mathcal{A}/E \models F$ iff $\mathcal{A} \models^E F$.*

7 Reasoning about Specifications

We revisit the model theory of behavioural and abstractor specification studied in [4]. Behavioural equivalence $\equiv_{OBS}$ is factorisable by the indistinguishability relation $\approx$ (theorem 6), and for any $\mathcal{A} \in \mathbf{CA}(\Sigma)$, $\mathcal{A}/\approx$ is fully abstract (theorem 5). The latter implies that $\approx_{\mathcal{A}}$ is a regular relation.[5] One important consequence from this setting is the following relationship between behavioural and abstractor specification. Due to space limitations, we only state the theorem without giving the definitions of symbols. For details, see [4].

[5] A $\mathbf{CA}(\Sigma)$-indexed family of PERs E is regular if $\mathcal{A}/E_{\mathcal{A}}$ is fully abstract (see [4]).

Theorem 9 (Bidoit et al. [4]). *Let $SP = (\Sigma, \Phi)$ be a specification, where Φ is a set of formulas in the higher-order logic over Σ. Then we have:*

$$\begin{aligned}
\mathbf{Mod}(\text{behaviour } SP \text{ w.r.t. } \approx) &= \mathbf{Abs}^{\equiv_{OBS}}(\mathbf{FA}^{\approx}(\mathbf{Mod}(SP))) \\
\mathbf{Mod}(\text{abstract } SP \text{ w.r.t. } \equiv_{OBS}) &= \mathbf{Mod}(\text{behaviour } SP/{\approx} \text{ w.r.t. } \approx) \\
\mathbf{Th}^{\approx}(\mathbf{Mod}(\text{behaviour } SP \text{ w.r.t. } \approx)) &= \mathbf{Th}(\mathbf{FA}^{\approx}(\mathbf{Mod}(SP))) \\
\mathbf{Th}^{\approx}(\mathbf{Mod}(\text{abstract } SP \text{ w.r.t. } \equiv_{OBS})) &= \mathbf{Th}(\mathbf{Mod}(SP/{\approx}))
\end{aligned}$$

Proof. See theorem 5.16, 6.8 and 7.4 in [4]. □

Related Work

The work by Bidoit, Hennicker and Wirsing[4] established the key idea of factorisability to relate behavioural equivalence and the indistinguishability relation, and they used this to reason about the semantics of behavioural and abstractor specifications. Subsequent work by Bidoit and Hennicker[3] discussed a proof method for showing behavioural equivalence in first order logic, and considered finitary axiomatisation of behavioural equality. The above work is extended by Hofmann and Sannella[8] to higher-order logic. This work is an extension of their work from first-order Σ-algebras to combinatory algebras. Bidoit and Tarlecki[5] gave a categorical generalisation of [4]. See the end of section 5 for comments on the relationship to the present paper.

Our characterisation of behavioural equivalence is related to Mitchell's representation independence theorem[10]. Honsell and Sannella[8] removed the restriction on the constants by using pre-logical relations instead of logical relations. This paper shows a similar result about behavioural equivalence, which subsumes closed observational equivalence.

In [5], Bidoit and Tarlecki give a relationship between behavioural satisfaction, behavioural equivalence, indistinguishability and correspondences in an abstract setting using concrete categories (a faithful functor to the category of (type-indexed) sets). By instantiating their concrete categories with various real examples, such as the category of multi-sorted algebras and regular algebras, we can derive suitable notions of behavioural equivalence, indistinguishability, etc. and theorems on them.

We can instantiate their abstract framework with the category of Σ-combinatory algebras $\mathbf{CA}(\Sigma)$, and obtain various results on behavioural equivalence and indistinguishability. Pre-logical relations correspond to spans (moreover correspondences), and pre-logical PERs correspond to partial congruences in their terminology. Category $\mathbf{CA}(\Sigma)$ satisfies certain properties,[6] thus we can obtain a theorem characterising behavioural equivalence via correspondences (see theorem 28 of [5]).

Their definitions of indistinguishability and behavioural equivalence are abstract: they define the indistinguishability relation $\approx_{\mathcal{A}}$ as the largest congruence over the full subobject $\langle |\mathcal{A}|_{OBS} \rangle_{\mathcal{A}}$ of $\mathcal{A}$. Then behavioural equivalence is defined by $\mathcal{A} \equiv_{OBS} \mathcal{B}$ iff $\mathcal{A}/{\approx_{\mathcal{A}}} \cong \mathcal{B}/{\approx_{\mathcal{B}}}$. In contrast, in this paper we give an explicit definition of behavioural

[6] Category $\mathbf{CA}(\Sigma)$ admits renaming and has full subobjects and surjective quotients. All full subobjects are compatible with $\mathbf{CA}(\Sigma)$-morphisms. Pullbacks preserve surjective quotients, and quotients are fully compatible with subobjects in $\mathbf{CA}(\Sigma)$.

equivalence and indistinguishability, and show the relationship between them in an elementary way.

8 Conclusion

We have extended the study of the relationship between *behavioural equivalence* and *indistinguishability* [4, 7] to the simply typed lambda calculus, where higher-order types are available. We characterised behavioural equivalence between two combinatory algebras by pre-logical relations, and showed that behavioural equivalence is factorised by indistinguishability. We also showed that standard satisfaction over $\mathcal{A}/E$ is equivalent to behavioural satisfaction w.r.t. a PER E over $\mathcal{A}$.

It is interesting to restrict the class of models from combinatory algebras to Henkin models, where the extensionality axiom holds. This changes the properties of the class of models; in particular it is not closed under quotient by pre-logical PERs. It will be interesting to see how behavioural equivalence and the indistinguishability relation are characterised in the class of Henkin models.

Acknowledgements I am grateful to Donald Sannella for his continuous encouragement to this work, and to John Longley for helpful discussions. This work has been partly supported by an LFCS studentship.

References

1. P. Andrews. *An introduction to mathematical logic and type theory: to truth through proof.* Academic Press, 1986.
2. H. Barendregt. *The Lambda Calculus-Its Syntax and Semantics.* North Holland, 1984.
3. M. Bidoit and R. Hennicker. Behavioural theories and the proof of behavioural properties. *Theoretical Computer Science*, 165(1):3–55, 1996.
4. M. Bidoit, R. Hennicker, and M. Wirsing. Behavioural and abstractor specifications. *Science of Computer Programming*, 25(2–3):149–186, 1995.
5. M. Bidoit and A. Tarlecki. Behavioural satisfaction and equivalence in concrete model categories. In *Proc. 21st Int. Coll. on Trees in Algebra and Programming (CAAP '96)*, volume 1059 of *LNCS*, pages 241–256. Springer, 1996.
6. L. Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15:81–91, 1950.
7. M. Hofmann and D. Sannella. On behavioural satisfaction and behavioural abstraction in higher-order logic. *Theoretical Computer Science*, 167(1–2):3–45, 1996.
8. F. Honsell and D. Sannella. Pre-logical relations. In *Proc. Computer Science Logic*, volume 1683 of *LNCS*, pages 546–561. Springer, 1999. An extended version is in *Information and Computation* 178:23–43,2002.
9. Furio Honsell, John Longley, Donald Sannella, and Andrzej Tarlecki. Constructive data refinement in typed lambda calculus. In *Proc. FoSSACS*, volume 1784 of *LNCS*, pages 149–164. Springer, 2000.
10. J. Mitchell. On the equivalence of data representations. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 305–330. Academic Press, San Diego, 1991.
11. J. Mitchell. *Foundations for Programming Languages.* MIT Press, 1996.
12. O. Schoett. Behavioural correctness of data representations. *Science of Computer Programming*, 14:43–57, 1990.