# On behavioural abstraction and behavioural satisfaction in higher-order logic*

Martin Hofmann† and Donald Sannella‡
Laboratory for Foundations of Computer Science
University of Edinburgh

**Abstract**

The behavioural semantics of specifications with higher-order logical formulae as axioms is analyzed. A characterization of behavioural abstraction via behavioural satisfaction of formulae in which the equality symbol is interpreted as indistinguishability, which is due to Reichel and was recently generalized to the case of first-order logic by Bidoit *et al*, is further generalized to this case. The fact that higher-order logic is powerful enough to express the indistinguishability relation is used to characterize behavioural satisfaction in terms of ordinary satisfaction, and to develop new methods for reasoning about specifications under behavioural semantics.

## 1 Introduction

An important ingredient in the use of algebraic specifications to describe data abstractions is the concept of *behavioural equivalence* between algebras, which seems to appropriately capture the "black box" character of data abstractions, see e.g. [GGM76], [GM82], [ST87] and [ST92]. Roughly speaking (since there are different choices of definition), two algebras $A, B$ over a signature $\Sigma$ are behaviourally equivalent with respect to a distinguished set $OBS$ of *observable types* if all computations that can be expressed using the functions in $\Sigma$ and that yield a result of a type in $OBS$ produce the same result in both $A$ and $B$. (The set $OBS$ is typically taken to include primitive types like Booleans and natural numbers.) A specification of a data abstraction should characterize a class of algebras that is closed under behavioural equivalence; otherwise it forbids some realizations that are indistinguishable from acceptable ones. Closure can be ensured by the specification framework (by making all specification-building operations deliver closed classes, see e.g. [NO88]) or by the specifier (by applying a specification-building operation, sometimes known as *behavioural abstraction*, to form the closure, see e.g. [SW83], [ST87]). The term "behavioural semantics" is sometimes used to characterize approaches that take the need for behavioural closure into account. Behavioural abstraction seems to be an implicit ingredient of model-oriented approaches to specification such as VDM and Z, where a specification spells out one or more concrete models but any program that delivers the same results is regarded as an acceptable realization.

An unfortunate problem with behavioural semantics in general and the behavioural abstraction operation in particular is that it complicates the task of reasoning about specifications. For example, if a specification $SP$ satisfies a formula $\varphi$ then the behavioural abstraction of $SP$ need not satisfy $\varphi$. Reasoning methods that are appropriate in the context of behavioural semantics have been developed, but these are either insufficiently powerful (e.g. [ST87], cf. Section 5 of [Sch92]) or tend

*A condensed version of this paper will appear in: *Proc. 20th Colloq. on Trees in Algebra and Programming*, Intl. Joint Conf. on Theory and Practice of Software Development (TAPSOFT), Aarhus. Springer LNCS (1995).

†E-mail mxh@dcs.ed.ac.uk. Supported by a Human Capital and Mobility fellowship, contract number ERBCHBICT930420.

‡E-mail dts@dcs.ed.ac.uk. Supported by an EPSRC Advanced Fellowship and EPSRC grants GR/H73103 and GR/J07303.

to be too complicated for convenient use in practice (e.g. [Hen91], [Far92]). One avenue of attack on this problem is to consider the relationship between the class of algebras produced by applying the behavioural abstraction operation to a specification $\langle\Sigma, \Phi\rangle$, and the class of algebras obtained by simply interpreting equality in the axioms $\Phi$ as *indistinguishability* rather than as identity. The latter approach, sometimes known as *behavioural satisfaction*, was pioneered by Reichel [Rei85] who showed that these two classes coincide when the axioms involved are conditional equations, provided that the conditions used are equations between terms of types in $OBS$. This yields a reasoning method for specifications involving behavioural abstraction: given a sound proof system for behavioural satisfaction, any consequence $\varphi$ of a specification $\langle\Sigma, \Phi\rangle$ that can be proved in that system will hold in the behavioural abstraction of $\langle\Sigma, \Phi\rangle$, provided $\Phi$ and $\varphi$ have the required form.

The usefulness of this reasoning method is limited by the fact that conditional equations are not powerful enough for convenient practical use in writing specifications. But in a recent development, Bidoit *et al* have generalized Reichel's result to the case of specifications with first-order equational formulae and reachability constraints as axioms, and to arbitrary relations of behavioural equivalence and indistinguishability. In [BHW94] they show that the coincidence of classes described above holds in this context as well, whenever the class of models of $\langle\Sigma, \Phi\rangle$ (under ordinary satisfaction) is closed under quotienting w.r.t. indistinguishability of values, provided that indistinguishability is *regular* and that behavioural equivalence is *factorizable* by indistinguishability. A companion paper [BH94a] uses this characterization as the basis for a reasoning method.

In this paper we examine these issues for the case of (flat) specifications with higher-order logical formulae as axioms. Our first main contribution is a generalization[1] of the framework and results of [BHW94]. Although it is not made explicit there, the main results in [BHW94] including the characterization theorem do not strongly depend on the form of axioms. The same result holds for any logical system for which behavioural satisfaction of a formula $\varphi$ in $A$ coincides with ordinary satisfaction of $\varphi$ in the quotient of $A$ w.r.t. indistinguishability and for which isomorphisms preserve and reflect satisfaction. In Sections 2 and 3 we give syntax and semantics for higher-order formulae and show that these properties hold for such formulae (Theorem 3.28 and Corollary 3.12 respectively). In Section 4 we formulate definitions of behavioural equivalence and indistinguishability, and we show that the former is factorizable by the latter (Theorem 5.15) and that indistinguishability is regular (Proposition 4.4). This leads directly to a characterization result analogous to the one in [BHW94] (Theorem 6.8). Although the generalization to higher-order logic results in certain complications, it also yields a simplification: since equality and reachability constraints may be expressed directly in higher-order logic, they need not be given specialized treatment, and the rôle of equality in the context of behavioural semantics is revealed as special case of something more general.

Higher-order logic provides sufficient power to express the indistinguishability relation as a predicate (Theorem 5.2, cf. [Sch94]). A second main contribution is the application of this fact to develop methods for reasoning about specifications under behavioural semantics. In Section 5 we characterize behavioural satisfaction in terms of ordinary satisfaction, by giving a translation that takes any formula $\varphi$ to a "relativized" formula $\ulcorner\varphi\urcorner$ such that the latter is satisfied exactly when the former is behaviourally satisfied (Corollary 5.7). This translation plays an important rôle in the comparison of various alternative definitions of behavioural equivalence, differing in the set of "experiments" used to test algebras, which leads to the conclusion that the three definitions considered yield the same relation (Corollary 5.16). These results, together with the characterization theorem of Section 6, lead directly to various proof methods that are summarized in Section 7.

# 2 The language of higher-order logic

The syntax of the typed variant of higher-order logic we will use is described below. The logic is higher-order because quantification over predicates (i.e. sets) is allowed in addition to the usual quantification over individuals. For the sake of simplicity, functions of higher type are not catered for, but see Section 8 for comments on a possible extension.

[1] In fact, [BHW94] uses the infinitary logic $L_{\omega_1\omega}$, so strictly speaking our framework is not a generalization. The extension to infinitary logic is easy and raises no interesting issues, so we omit it.

**Definition 2.1** *A* signature $\Sigma$ *consists of a set* $B$ *of* base types *and a set* $C$ *of* constants *such that each* $c \in C$ *has an arity* $n \geq 0$*, an* $n$*-tuple of argument types* $b_1, \ldots, b_n \in B$ *and a result type* $b \in B$*, which we abbreviate* $c : b_1 \times \cdots \times b_n \rightarrow b$*.*

Let $\Sigma = \langle B, C \rangle$ be a signature.

**Definition 2.2** *The* types *over* $\Sigma$ *are given by the following grammar:*

$$\tau \quad ::= \quad b \mid [\tau_1, \ldots, \tau_n] \qquad\qquad (n \geq 0)$$

*where* $b \in B$*.* $\mathit{Types}(\Sigma)$ *denotes the set of all types over* $\Sigma$*.*

A type of the form $[\tau_1, \ldots, \tau_n]$ may be regarded as the type of $n$-ary predicates taking arguments of types $\tau_1, \ldots, \tau_n$. For example, $\mathit{is\text{-}zero} : [\mathit{int}]$, $<: [\mathit{int}, \mathit{int}]$, $\mathit{has\text{-}property} : [\mathit{int}, [\mathit{int}]]$, where $\mathit{is\text{-}zero}(3)$ does not hold but $\mathit{has\text{-}property}(0, \mathit{is\text{-}zero})$ does. A more suggestive syntax for a type $[\tau_1, \ldots, \tau_n]$ might be $\tau_1 \times \cdots \times \tau_n \rightarrow \mathbf{Prop}$, where $\mathbf{Prop}$ is the type of propositions; in particular, the type $[]$ may be thought of as $\mathbf{Prop}$. However, since $\lambda$-abstraction can be used to form predicates but not ordinary functions (see below), we need to distinguish between the arrow used to write the types of constants and the arrow in $\tau_1 \times \cdots \times \tau_n \rightarrow \mathbf{Prop}$.

Let $X$ be a fixed infinite set of variables, ranged over by $x$.

**Definition 2.3** *The* terms *over* $\Sigma$ *are given by the following grammar:*

$$t \quad ::= \quad x \mid c(t_1, \ldots, t_n) \mid \lambda(x_1{:}\tau_1, \ldots, x_n{:}\tau_n).t \mid t(t_1, \ldots, t_n) \mid t \Rightarrow t' \mid \forall x{:}\tau.t \qquad\qquad (n \geq 0)$$

*where* $c \in C$*. As usual, we regard* $\alpha$*-convertible terms as equal, where the binding constructs are* $\lambda$ *and* $\forall$*. We write* $c$ *as an abbreviation for* $c()$ *when* $c : \rightarrow b$*, and* $\forall x_1, \ldots, x_n{:}\tau.t$ *as an abbreviation for* $\forall x_1{:}\tau. \cdots .\forall x_n{:}\tau.t$*.*

Function application (written $c(t_1, \ldots, t_n)$) is distinguished from predicate application (written $t(t_1, \ldots, t_n)$) although both notations are similar. $\lambda$-abstraction is for forming predicates; implication ($\Rightarrow$) and universal quantification are for forming propositions. There is just one syntax class for terms: terms that denote individuals (e.g. $+(3, 2)$) are not distinguished syntactically from terms denoting predicates (e.g. $\lambda(x{:}\mathit{int}, y{:}\mathit{int}).\mathit{prime}(+(x, y))$) or propositions (e.g. $\forall P{:}[\mathit{int}].(\forall x{:}\mathit{int}.P(x)) \Rightarrow P(3)$). But in order for a term to denote anything at all, it has to be typable according to the following definitions.

**Definition 2.4** *A* context $\Gamma$ *is a sequence of the form* $x_1 : \tau_1, \ldots, x_n : \tau_n$ *where* $x_i \neq x_j$ *for all* $i \neq j$*. We write* $\Gamma(x_j)$ *for* $\tau_j$ *and* $\mathit{Vars}(\Gamma)$ *for* $\{x_1, \ldots, x_n\}$*, and we identify* $\Gamma$ *with the* $\mathit{Types}(\Sigma)$*-sorted set of variables such that* $\Gamma_\tau = \{x \in \mathit{Vars}(\Gamma) \mid \Gamma(x) = \tau\}$ *for all* $\tau \in \mathit{Types}(\Sigma)$*. Concatenation of contexts, written* $\Gamma, \Gamma'$*, is required to yield a context, i.e. it is required that* $\mathit{Vars}(\Gamma) \cap \mathit{Vars}(\Gamma') = \emptyset$*. Let* $T \subseteq \mathit{Types}(\Sigma)$ *be a subset of the set of types over* $\Sigma$*; then* $\Gamma$ *is called a* $T$-context *if* $\Gamma(x) \in T$ *for all* $x \in \mathit{Vars}(\Gamma)$*.*

**Definition 2.5** *We write* $\Gamma \vdash t : \tau$ *if this judgement is derivable using the rules in Figure 1, and then we call* $t$ *a* term in context $\Gamma$*. A term* $t$ *is* closed *if it is typable in the empty context, i.e. if* $\vdash t : \tau$*. A* predicate *(*in context $\Gamma$*) is a term* $t$ *such that* $\Gamma \vdash t : [\tau_1, \ldots, \tau_n]$*. A* formula *(*in context $\Gamma$*) is a term* $\varphi$ *such that* $\Gamma \vdash \varphi : []$*.*

**Proposition 2.6** *The following weakening and permutation rules are admissible in the system of rules given in Figure 1:*

$$\frac{\Gamma \vdash t : \tau}{\Gamma, x : \tau' \vdash t : \tau} \qquad (\textsc{weak})$$

$$\frac{\Gamma, \Gamma' \vdash t : \tau}{\Gamma', \Gamma \vdash t : \tau} \qquad (\textsc{perm})$$

Proof: *Obvious.*

$$\frac{}{\Gamma \vdash x : \Gamma(x)} \qquad (\textsc{var})$$

$$\frac{\Gamma, x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : [\,]}{\Gamma \vdash \lambda(x_1{:}\tau_1, \ldots, x_n{:}\tau_n).t : [\tau_1, \ldots, \tau_n]} \qquad (\lambda)$$

$$\frac{c : b_1 \times \cdots \times b_n \rightarrow b \qquad \Gamma \vdash t_1 : b_1 \quad \cdots \quad \Gamma \vdash t_n : b_n}{\Gamma \vdash c(t_1, \ldots, t_n) : b} \qquad (\textsc{fun})$$

$$\frac{\Gamma \vdash t : [\tau_1, \ldots, \tau_n] \qquad \Gamma \vdash t_1 : \tau_1 \quad \cdots \quad \Gamma \vdash t_n : \tau_n}{\Gamma \vdash t(t_1, \ldots, t_n) : [\,]} \qquad (\textsc{pred})$$

$$\frac{\Gamma \vdash t : [\,] \qquad \Gamma \vdash t' : [\,]}{\Gamma \vdash t \Rightarrow t' : [\,]} \qquad (\Rightarrow)$$

$$\frac{\Gamma, x : \tau \vdash t : [\,]}{\Gamma \vdash \forall x{:}\tau.t : [\,]} \qquad (\forall)$$

Figure 1: Typing rules

There is no need to include equality as a built-in predicate, since it is expressible using higher-order quantification. That is, suppose $\Gamma \vdash t : \tau$ and $\Gamma \vdash t' : \tau$; then

$$t =_\tau t' \quad \text{abbreviates} \quad \forall P{:}[\tau].P(t) \Rightarrow P(t')$$

where $P$ is chosen arbitrarily such that $P \notin \mathit{Vars}(\Gamma)$.

Existential quantification and the missing connectives are expressible as usual in terms of $\forall$ and $\Rightarrow$:

$$\begin{array}{rll}
\mathit{true} & \text{abbreviates} & \forall P{:}[\,].P \Rightarrow P \\
\mathit{false} & \text{abbreviates} & \forall P{:}[\,].P \\
\neg\varphi & \text{abbreviates} & \varphi \Rightarrow \mathit{false} \\
\varphi \vee \varphi' & \text{abbreviates} & (\neg\varphi) \Rightarrow \varphi' \\
\varphi \wedge \varphi' & \text{abbreviates} & \neg(\neg\varphi \vee \neg\varphi') \\
\exists x{:}\tau.\varphi & \text{abbreviates} & \neg\forall x{:}\tau.\neg\varphi
\end{array}$$

Finally, there is no need to treat reachability constraints as a special case, since induction principles are expressible. For example, suppose that $\mathit{nat} \in B$ and $0 : \;\rightarrow \mathit{nat}$ and $\mathit{succ} : \mathit{nat} \rightarrow \mathit{nat}$ are in $C$. The following formula (call this $\mathit{GENNAT}$ for future reference) asserts that $\mathit{nat}$ is generated by 0 and $\mathit{succ}$:

$$\forall P{:}[\mathit{nat}].(\, P(0) \wedge \forall n{:}\mathit{nat}.(P(n) \Rightarrow P(\mathit{succ}(n)))\,) \Rightarrow \forall n{:}\mathit{nat}.P(n)$$

The following example gives a taste of the expressive power of the language thus defined.

**Example 2.7** Consider the signature with base types $\mathit{sched}$ (schedule) and $\mathit{proc}$ (process) and constants $\mathit{start} :\rightarrow \mathit{sched}$, $\mathit{step} : \mathit{sched} \rightarrow \mathit{sched}$ and $\mathit{who} : \mathit{sched} \rightarrow \mathit{proc}$. We would like to require that $\mathit{start}$ is a fair schedule, i.e. that it schedules each process infinitely often. The following is essentially a translation of a formula in the modal mu-calculus [Sti92] into higher-order logic.

The following predicate expresses that a predicate $P$ holds infinitely often in a schedule $s$:

$$\mathit{infinitely\text{-}often} =_{\mathrm{def}} \lambda(P{:}[\mathit{sched}], s{:}\mathit{sched}).\ \mathit{always}((\lambda(s'{:}\mathit{sched}).\mathit{eventually}(P, s'))\ ,\ s)$$

Then the required fairness property is $\mathit{fair}(\mathit{start})$, where $\mathit{fair}$ is expressed in terms of $\mathit{infinitely\text{-}often}$ as follows:

$$\mathit{fair} =_{\mathrm{def}} \lambda(s{:}\mathit{sched}).\ \forall p{:}\mathit{proc}.\mathit{infinitely\text{-}often}((\lambda(s'{:}\mathit{sched}).\mathit{who}(s') = p)\ ,\ s)$$

The predicates *always* and *eventually* are expressed as greatest resp. least fixed points. The least and greatest fixed point operators can be expressed directly as follows:

$$\begin{array}{lll} \mu & =_{\text{def}} & \lambda(\Phi{:}[[sched], sched]\ ,\ s{:}sched).\,\forall P{:}[sched].(\forall s'{:}sched.\Phi(P,s') \Rightarrow P(s')) \Rightarrow P(s) \\ \nu & =_{\text{def}} & \lambda(\Phi{:}[[sched], sched]\ ,\ s{:}sched).\,\exists P{:}[sched].(\forall s'{:}sched.P(s') \Rightarrow \Phi(P,s')) \wedge P(s) \end{array}$$

and the definitions of *always* and *eventually* are then:

$$\begin{array}{lll} always & =_{\text{def}} & \lambda(P{:}[sched], s{:}sched). \\ & & \quad \nu(\lambda(always\text{-}P{:}[sched], s'{:}sched).\ P(s') \wedge always\text{-}P(step(s'))\ ,\ s) \\ eventually & =_{\text{def}} & \lambda(P{:}[sched], s{:}sched). \\ & & \quad \mu(\lambda(eventually\text{-}P{:}[sched], s'{:}sched).\ P(s') \vee eventually\text{-}P(step(s'))\ ,\ s) \end{array}$$

Expanding *fair*(*start*) gives a single formula expressing the required property. □

The language defined above is a trimmed version of the "classical theory of simple types" as introduced by Henkin in [Hen50]. Henkin considers non-standard models for which a natural Gentzen-style proof system is sound and complete. A good reference is also Chapter 4 of Schütte's monograph [Sch77] where cut-elimination for this system is established.

# 3 Semantics of higher-order logic

Let $\Sigma = \langle B, C\rangle$ be a signature.

Terms over $\Sigma$ are interpreted in the context of a $\Sigma$-algebra which gives meaning to the base types and the constants in $\Sigma$.

**Definition 3.1** *A* $\Sigma$-algebra $A$ *consists of a* carrier set $[\![b]\!]_A$ *for every* $b \in B$, *and interpretations of constants* $[\![c]\!]_A \in ([\![b_1]\!]_A \times \cdots \times [\![b_n]\!]_A \rightarrow [\![b]\!]_A)$ *for every* $c : b_1 \times \cdots \times b_n \rightarrow b$ *in* $C$. *The class of all* $\Sigma$-*algebras is denoted* $Alg(\Sigma)$. $\Sigma$-homomorphisms *and* $\Sigma$-isomorphisms *are as usual; we write* $A \cong A'$ *if there is a* $\Sigma$-*isomorphism* $h : A \rightarrow A'$.

Let $A$ be a $\Sigma$-algebra.

We define two interpretations for terms. The first is the obvious "standard" interpretation with respect to an environment mapping free variables to values. The second interpretation is modulo a partial congruence relation on $A$. In the latter interpretation, quantification (and $\lambda$-abstraction) is over only those elements of types that respect the congruence; as a result, equality in formulae refers to the congruence rather than to identity of values. The particular partial congruence of interest will be a relation of indistinguishability with respect to a given set of observable base types, to be defined in Section 4. Theorem 3.28 below demonstrates a relationship between the two interpretations that will be crucial in the sequel.

Our use of *partial* congruences in Section 3.2 below stems from the need to establish an appropriate relationship between indistinguishability and behavioural equivalence, see Theorem 5.15, in order to apply the characterization theorems in Section 6. If the indistinguishability relation were not defined as a partial congruence, the desired relationship with the behavioural equivalence relation would not hold.

## 3.1 Standard interpretation

**Definition 3.2** *Types of the form* $[\tau_1, \ldots, \tau_n]$ *are interpreted as follows:*

$$[\![[\tau_1, \ldots, \tau_n]]\!]_A = Pow([\![\tau_1]\!]_A \times \cdots \times [\![\tau_n]\!]_A).$$

Thus, $[\![[]]\!]_A$ is $\{\{\}, \{*\}\}$ where $*$ is the empty tuple. Recalling that $[]$ means **Prop**, $\{\}$ may be thought of as denoting **false** and $\{*\}$ as denoting **true**, so we will use the abbreviation *ff* for $\{\}$ and *tt* for $\{*\}$.

Let $\Gamma$ be a context.

**Definition 3.3** *A* $\Gamma$-environment (on $A$) *is a* $Types(\Sigma)$*-sorted function* $\rho = \langle \rho_\tau : \Gamma_\tau \to [\![\tau]\!]_A \rangle_{\tau \in Types(\Sigma)}$. *We write* $[x_1 \mapsto v_1, \ldots, x_n \mapsto v_n]$ *to denote the evident environment, and the notation* $\rho[x_1 \mapsto v_1, \ldots, x_n \mapsto v_n]$ *denotes the environment* $\rho$ *superseded at* $x_1, \ldots, x_n$ *by* $v_1, \ldots, v_n$ *respectively. When* $x \in Vars(\Gamma)$ *we write* $\rho(x)$ *for* $\rho_{\Gamma(x)}(x)$*. Let* $T \subseteq Types(\Sigma)$*; a* $\Gamma$*-environment* $\rho$ *is* $T$-surjective *if* $\rho_\tau : \Gamma_\tau \to [\![\tau]\!]_A$ *is surjective for each* $\tau \in T$.

**Definition 3.4** *Let* $\rho$ *be a* $\Gamma$*-environment. The interpretation of constants is extended to terms in context* $\Gamma$ *as follows:*

$$
\begin{array}{rcl}
[\![x]\!]_{\rho,A} & = & \rho(x) \\
[\![c(t_1,\ldots,t_n)]\!]_{\rho,A} & = & [\![c]\!]_A([\![t_1]\!]_{\rho,A},\ldots,[\![t_n]\!]_{\rho,A}) \\
[\![\lambda(x_1{:}\tau_1,\ldots,x_n{:}\tau_n).t]\!]_{\rho,A} & = & \{(v_1,\ldots,v_n) \mid v_1 \in [\![\tau_1]\!]_A \textit{ and } \cdots \textit{ and } v_n \in [\![\tau_n]\!]_A \\
 & & \qquad \textit{and } [\![t]\!]_{\rho[x_1\mapsto v_1,\ldots,x_n\mapsto v_n],A} = tt\} \\
[\![t(t_1,\ldots,t_n)]\!]_{\rho,A} & = & \textit{if } ([\![t_1]\!]_{\rho,A},\ldots,[\![t_n]\!]_{\rho,A}) \in [\![t]\!]_{\rho,A} \textit{ then } tt \textit{ else } \mathit{ff} \\
[\![t \Rightarrow t']\!]_{\rho,A} & = & \textit{if } [\![t]\!]_{\rho,A} = tt \textit{ then } [\![t']\!]_{\rho,A} \textit{ else } tt \\
[\![\forall x{:}\tau.t]\!]_{\rho,A} & = & \textit{if } [\![t]\!]_{\rho[x\mapsto v],A} = tt \textit{ for all } v \in [\![\tau]\!]_A \textit{ then } tt \textit{ else } \mathit{ff}
\end{array}
$$

The following substitution property will be handy in proofs in later sections. As usual, $s[x := t]$ denotes the result of simultaneously replacing all occurrences of the variable $x$ in the term $s$ by the term $t$, with appropriate changes of bound variable names to avoid variable capture.

**Proposition 3.5** *For any* $\Gamma \vdash t : \tau$ *and* $\Gamma, x : \tau \vdash s : \tau'$ *and any* $\Gamma$*-environment* $\rho$, $[\![s]\!]_{\rho[x\mapsto [\![t]\!]_{\rho,A}],A} = [\![s[x := t]]\!]_{\rho,A}$.

PROOF: *By induction on the structure of* $s$. □

The following shows that the above interpretation of terms and types is sound with respect to the typing relation.

**Proposition 3.6** *If* $\Gamma \vdash t : \tau$ *and* $\rho$ *is a* $\Gamma$*-environment then* $[\![t]\!]_{\rho,A} \in [\![\tau]\!]_A$.

PROOF: *By induction on the structure of the derivation of* $\Gamma \vdash t : \tau$. □

The following proposition demonstrates that $=_\tau$ really is equality (i.e. identity of values).

**Proposition 3.7** *Suppose* $v, v' \in [\![\tau]\!]_A$ *for some type* $\tau$*. Then for any environment* $\rho$, $[\![x =_\tau y]\!]_{\rho[x\mapsto v, y\mapsto v'],A} = tt$ *iff* $v = v'$.

PROOF:

$\Longleftarrow$: *Obvious.*

$\Longrightarrow$: *Suppose that* $[\![\forall P{:}[\tau].P(x) \Rightarrow P(y)]\!]_{\rho[x\mapsto v, y\mapsto v'],A} = tt$ *and consider the predicate* $\{v\} \in [\![[\tau]]\!]_A$. *We have* $[\![P(x)]\!]_{\rho[x\mapsto v, P\mapsto\{v\}],A} = tt$ *and so* $[\![P(y)]\!]_{\rho[y\mapsto v', P\mapsto\{v\}],A} = tt$*; thus* $v' \in \{v\}$*, i.e.* $v = v'$. □

This entitles us to use $\varphi \Leftrightarrow \varphi'$ as an abbreviation for $\varphi =_{[]} \varphi'$.

**Definition 3.8** *Let* $B' \subseteq B$ *be a subset of the set of base types in* $\Sigma$*, and let* $b \in B$*. A value* $v \in [\![b]\!]_A$ *is* $B'$-reachable *if there is a* $B'$*-context* $\Gamma$*, a term* $t$ *with* $\Gamma \vdash t : b$*, and a* $\Gamma$*-environment* $\rho$*, such that* $[\![t]\!]_{\rho,A} = v$.

Intuitively, $v$ is $B'$-reachable if $v$ can be obtained by application of constants to values of types in $B'$.

Recall the earlier claim that the formula $GENNAT$ asserts that the type $nat$ is generated by $0 : \to nat$ and $succ : nat \to nat$. Indeed, for any algebra $A$ over the relevant signature, $[\![GENNAT]\!]_{[],A} = tt$ iff every value of type $nat$ in $A$ is $\emptyset$-reachable. (The "if" direction is obvious; to see that the "only if" direction holds, simply instantiate $GENNAT$ with the predicate $P = \{v \in [\![nat]\!]_A \mid v \text{ is } \emptyset\text{-reachable}\} \in [\![[nat]]\!]_A$.)

It is easy to see that the abbreviations defined for the connectives $\neg$, $\vee$, $\wedge$ and for $\exists$ and $true$ have the expected meaning. The following shows that the abbreviation defined for $false$ is also correct.

**Proposition 3.9** *For any environment $\rho$, $[\![false]\!]_{\rho,A} = ff$.*

PROOF: *Recall that false is $\forall P{:}[\,].P$ and take $P = ff \in [\![[\,]]\!]_A$.* $\square$

**Definition 3.10** *Let $\varphi$ be a formula in context $\Gamma$. Suppose $\rho$ is a $\Gamma$-environment; then we write $A \models_\rho \varphi$ if $[\![\varphi]\!]_{\rho,A} = tt$. We write $A \models \varphi$ ($A$* satisfies *$\varphi$) if $A \models_\rho \varphi$ for all $\Gamma$-environments $\rho$. If $\varphi'$ is also a formula in context $\Gamma$, we write $\varphi \models\!\mid \varphi'$ ($\varphi$ is* equivalent *to $\varphi'$) if for all $A \in Alg(\Sigma)$ and all $\Gamma$-environments $\rho$, $A \models_\rho \varphi$ iff $A \models_\rho \varphi'$. Finally, if $\Phi$ is a set of formulae in context $\Gamma$ then we write $A \models \Phi$ if $A \models \varphi$ for all $\varphi \in \Phi$.*

The following proposition is used to show that isomorphisms preserve and reflect satisfaction, as in almost any conceivable logical system.

**Proposition 3.11** *Let $h : A \to A'$ be an isomorphism. Extend $h$ to bracket types by taking $h_{[\tau_1,\ldots,\tau_n]}(p) = \{(h_{\tau_1}(v_1),\ldots,h_{\tau_n}(v_n)) \mid (v_1,\ldots,v_n) \in p\} \in [\![[\tau_1,\ldots,\tau_n]]\!]_{A'}$ for $p \in [\![[\tau_1,\ldots,\tau_n]]\!]_A$. Let $t$ be a term in context $\Gamma$ and let $\rho$ be a $\Gamma$-environment; then $h([\![t]\!]_{\rho,A}) = [\![t]\!]_{h\circ\rho,A'}$.*

PROOF: *By induction on the structure of $t$.* $\square$

**Corollary 3.12** *If $A \cong A'$ then $A \models \varphi$ iff $A' \models \varphi$.*

PROOF: *By Proposition 3.11, since $h_{[\,]}$ is the identity and since $h_\tau$ is a bijection for every type $\tau$.* $\square$

## 3.2 Interpretation w.r.t. a partial congruence

**Definition 3.13** *A* partial congruence *$\approx$ on $A$ is a family of* partial equivalence relations *(i.e., symmetric and transitive relations) $\langle \approx_b \subseteq [\![b]\!]_A \times [\![b]\!]_A \rangle_{b\in B}$ such that for all $c : b_1 \times \cdots \times b_n \to b$ in $C$ and all $v_1, v_1' \in [\![b_1]\!]_A, \ldots, v_n, v_n' \in [\![b_n]\!]_A$, if $v_1 \approx_{b_1} v_1'$ and $\cdots$ and $v_n \approx_{b_n} v_n'$ then $[\![c]\!]_A(v_1,\ldots,v_n) \approx_b [\![c]\!]_A(v_1',\ldots,v_n')$. A (*total*)* congruence *is a reflexive partial congruence.*

Let $\approx$ be a partial congruence on $A$. As suggested at the beginning of Section 3, the partial congruence of interest will be a relation of indistinguishability to be defined later. The reader may find it helpful to keep this in mind in order to understand the motivation behind some of the definitions and results below. We do not restrict attention to this particular partial congruence at this point because much of the sequel does not depend on the special features of this relation, and because there are several different indistinguishability relations of potential interest (although we will consider only one).

The idea behind the development which follows is to generalise the usual definition of satisfaction up to a partial congruence in first-order equational logic to higher-order logic. Whereas in the first-order case it is enough to interpret the primitive equality symbol as the partial congruence and to restrict all quantifiers to values lying in the domain of the partial congruence, the situation is more complicated here. We must make sure that the predicate variables only range over predicates which "respect" the partial congruence. What this means exactly is not entirely obvious for types with nested brackets. That the definition we give is indeed the right generalisation of the first-order case is shown by Proposition 3.24 and Theorem 3.28. In the first-order case, Proposition 3.24 is obvious from the definition of satisfaction.

The following definition explains how to extend the partial congruence $\approx$, which relates values of base types only, to a so-called *logical relation* (see e.g. [Mit90]) over all types. The resulting relation will be used below to give an interpretation of bracket types.

**Definition 3.14** *We extend $\approx$ to "bracket" types by taking $p \approx_{[\tau_1,\ldots,\tau_n]} p'$ for $p, p' \in [\![[\tau_1,\ldots,\tau_n]]\!]_A$ iff for all $v_1, v_1' \in [\![\tau_1]\!]_A, \ldots, v_n, v_n' \in [\![\tau_n]\!]_A$, if $v_1 \approx_{\tau_1} v_1'$ and $\cdots$ and $v_n \approx_{\tau_n} v_n'$ then $(v_1,\ldots,v_n) \in p$ iff $(v_1',\ldots,v_n') \in p'$. We say that $v \in [\![\tau]\!]_A$* respects *$\approx$ if $v \approx_\tau v$.*

A predicate $p \in [\![[\tau_1,\ldots,\tau_n]]\!]_A$ respects $\approx$ if it does not differentiate between values that are related by $\approx$. Note that trivially $\emptyset \approx_{[\tau_1,\ldots,\tau_n]} \emptyset$, and that $v \approx_{[\,]} v'$ iff $v = v'$.

**Proposition 3.15** *$\approx_\tau$ is a partial equivalence relation for any type $\tau$.*

PROOF: *Obvious.* □

Note that extending a (total) congruence to bracket types does not in general yield a (total) equivalence relation.

**Corollary 3.16** *If $v \approx_\tau v'$ then $v$ and $v'$ respect $\approx$.*

PROOF: *Apply symmetry and transitivity.* □

The difference between the standard interpretation of terms and their interpretation with respect to a partial congruence stems from the following definition.

**Definition 3.17** *Interpretation of types w.r.t. $\approx$ is defined as follows:*

$$\begin{array}{rcl} [\![b]\!]_A^{\approx} & = & \{v \in [\![b]\!]_A \mid v \textit{ respects} \approx\} \\ [\![[\tau_1,\ldots,\tau_n]]\!]_A^{\approx} & = & \{p \in Pow([\![\tau_1]\!]_A^{\approx} \times \cdots \times [\![\tau_n]\!]_A^{\approx}) \mid p \textit{ respects} \approx\} \end{array}$$

We have $[\![[\,]]\!]_A^{\approx} = [\![[\,]]\!]_A = \{\mathit{ff}, \mathit{tt}\}$. Note that if $\approx$ is a congruence, then $[\![b]\!]_A^{\approx} = [\![b]\!]_A$. The second clause of the above definition is well-formed because of the following proposition.

**Proposition 3.18** *$[\![\tau]\!]_A^{\approx} \subseteq [\![\tau]\!]_A$ for any type $\tau$.*

PROOF: *By induction on the structure of $\tau$. (Thus the proof that $[\![[\tau_1,\ldots,\tau_n]]\!]_A^{\approx}$ is well-defined depends on $[\![\tau_1]\!]_A^{\approx},\ldots,[\![\tau_n]\!]_A^{\approx}$, which have been shown to be well-defined at a previous stage.)* □

The following proposition shows that the extension of $\approx$ to bracket types, restricted to type interpretations $[\![[\tau_1,\ldots,\tau_n]]\!]_A^{\approx}$, is trivial in the sense that it does not identify distinct values.

**Proposition 3.19** *For all $p, p' \in [\![[\tau_1,\ldots,\tau_n]]\!]_A^{\approx}$, if $p \approx_{[\tau_1,\ldots,\tau_n]} p'$ then $p = p'$.*

PROOF: *Let $v_1 \in [\![\tau_1]\!]_A^{\approx},\ldots,v_n \in [\![\tau_n]\!]_A^{\approx}$; then $v_1 \approx_{\tau_1} v_1$ and $\cdots$ and $v_n \approx_{\tau_n} v_n$. If $p \approx_{[\tau_1,\ldots,\tau_n]} p'$ then $(v_1 \ldots, v_n) \in p$ iff $(v_1 \ldots, v_n) \in p'$, i.e. $p = p'$.* □

Let $\Gamma$ be a context.

**Definition 3.20** *A* $\Gamma$-environment *(*w.r.t. $\approx$, on $A$*) is a $Types(\Sigma)$-sorted function $\rho = \langle \rho_\tau : \Gamma_\tau \to [\![\tau]\!]_A^{\approx} \rangle_{\tau \in Types(\Sigma)}$. We adopt the previously-explained notations for environments.*

**Definition 3.21** *Let $\rho$ be a $\Gamma$-environment w.r.t. $\approx$. The interpretation w.r.t. $\approx$ of terms that are typable in context $\Gamma$ is defined as follows:*

$$\begin{array}{rcl} [\![x]\!]_{\rho,A}^{\approx} & = & \rho(x) \\ [\![c(t_1,\ldots,t_n)]\!]_{\rho,A}^{\approx} & = & [\![c]\!]_A([\![t_1]\!]_{\rho,A}^{\approx},\ldots,[\![t_n]\!]_{\rho,A}^{\approx}) \\ [\![\lambda(x_1{:}\tau_1,\ldots,x_n{:}\tau_n).t]\!]_{\rho,A}^{\approx} & = & \{(v_1,\ldots,v_n) \mid v_1 \in [\![\tau_1]\!]_A^{\approx} \textit{ and } \cdots \textit{ and } v_n \in [\![\tau_n]\!]_A^{\approx} \\ & & \qquad \textit{and } [\![t]\!]_{\rho[x_1\mapsto v_1,\ldots,x_n\mapsto v_n],A}^{\approx} = tt\} \\ [\![t(t_1,\ldots,t_n)]\!]_{\rho,A}^{\approx} & = & \textit{if } ([\![t_1]\!]_{\rho,A}^{\approx},\ldots,[\![t_n]\!]_{\rho,A}^{\approx}) \in [\![t]\!]_{\rho,A}^{\approx} \textit{ then } tt \textit{ else } \mathit{ff} \\ [\![t \Rightarrow t']\!]_{\rho,A}^{\approx} & = & \textit{if } [\![t]\!]_{\rho,A}^{\approx} = tt \textit{ then } [\![t']\!]_{\rho,A}^{\approx} \textit{ else } tt \\ [\![\forall x{:}\tau.t]\!]_{\rho,A}^{\approx} & = & \textit{if } [\![t]\!]_{\rho[x\mapsto v],A}^{\approx} = tt \textit{ for all } v \in [\![\tau]\!]_A^{\approx} \textit{ then } tt \textit{ else } \mathit{ff} \end{array}$$

A comparison of the above definition with the corresponding definition for the standard interpretation (Definition 3.4) reveals that the only difference is the change to the meaning of $\lambda$-abstraction and universal quantification induced by the different interpretation of types.

The proof of soundness does not go through directly; a stronger induction hypothesis is required.

**Proposition 3.22** *If $\Gamma \vdash t : \tau$ and $\rho, \rho'$ are $\Gamma$-environments w.r.t. $\approx$ such that $\rho(x) \approx_{\Gamma(x)} \rho'(x)$ for each $x \in Vars(\Gamma)$, then $[\![t]\!]^{\approx}_{\rho,A}, [\![t]\!]^{\approx}_{\rho',A} \in [\![\tau]\!]_A$ and $[\![t]\!]^{\approx}_{\rho,A} \approx_\tau [\![t]\!]^{\approx}_{\rho',A}$.*

PROOF: *By induction on the structure of the derivation of $\Gamma \vdash t : \tau$. The proof that $[\![t]\!]^{\approx}_{\rho,A}, [\![t]\!]^{\approx}_{\rho',A} \in [\![\tau]\!]_A$ is exactly the same as the proof of Proposition 3.6.* $\square$

**Corollary 3.23** *If $\Gamma \vdash t : \tau$ and $\rho$ is a $\Gamma$-environment w.r.t. $\approx$ then $[\![t]\!]^{\approx}_{\rho,A} \in [\![\tau]\!]^{\approx}_A$.*

PROOF: *Apply Proposition 3.22 with $\rho' = \rho$.* $\square$

The following proposition shows that $=_\tau$ refers to the partial congruence $\approx$ under interpretation of terms w.r.t. $\approx$. This is due to the fact that the quantifier in the formula $\forall P{:}[\tau].P(t) \Rightarrow P(t')$ (which $t =_\tau t'$ abbreviates) ranges over predicates that respect $\approx$.

**Proposition 3.24** *Suppose $v, v' \in [\![\tau]\!]^{\approx}_A$ for some type $\tau$. Then for any environment $\rho$ w.r.t. $\approx$, $[\![x =_\tau y]\!]^{\approx}_{\rho[x\mapsto v, y\mapsto v'],A} = tt$ iff $v \approx_\tau v'$.*

PROOF:

$\Longleftarrow$: *Suppose $v \approx_\tau v'$. Any predicate $p \in [\![[\tau]]\!]^{\approx}_A$ respects $\approx$, i.e. $v \in p$ iff $v' \in p$.*

$\Longrightarrow$: *Suppose that $[\![\forall P{:}[\tau].P(x) \Rightarrow P(y)]\!]^{\approx}_{\rho[x\mapsto v, y\mapsto v'],A} = tt$ and consider the predicate $p = \{w \mid w \approx_\tau v\} \in [\![[\tau]]\!]^{\approx}_A$. We have $[\![P(x)]\!]^{\approx}_{\rho[x\mapsto v, P\mapsto p],A} = tt$ so $[\![P(y)]\!]^{\approx}_{\rho[y\mapsto v', P\mapsto p],A} = tt$; thus $v' \in p$, i.e. $v' \approx_\tau v$.* $\square$

The interpretation of the formula $GENNAT$ with respect to $\approx$ is also different from what it was under the standard interpretation. For any algebra $A$ over the relevant signature, $[\![GENNAT]\!]^{\approx}_{[],A} = tt$ iff every value in $[\![nat]\!]^{\approx}_A$ is congruent to a $\emptyset$-reachable value. (The "if" direction depends on the requirement that $P \in [\![[nat]]\!]^{\approx}_A$ rather than $P \in [\![[nat]]\!]_A$; to see that the "only if" direction holds, instantiate $GENNAT$ with $P = \{v \in [\![nat]\!]^{\approx}_A \mid \exists v' \in [\![nat]\!]^{\approx}_A . v \approx_{nat} v'$ and $v'$ is $\emptyset$-reachable$\} \in [\![[nat]]\!]^{\approx}_A$. Note that there are choices of $A$ and $\approx$ for which $\{v \in [\![nat]\!]^{\approx}_A \mid v$ is $\emptyset$-reachable$\} \notin [\![[nat]]\!]^{\approx}_A$.)

**Definition 3.25** *Let $\varphi$ be a formula in context $\Gamma$. Suppose $\rho$ is a $\Gamma$-environment w.r.t. $\approx$; then we write $A \models^{\approx}_\rho \varphi$ if $[\![\varphi]\!]^{\approx}_{\rho,A} = tt$. We write $A \models^{\approx} \varphi$ ($A$* satisfies *$\varphi$* w.r.t. *$\approx$) if $A \models^{\approx}_\rho \varphi$ for all $\Gamma$-environments $\rho$ w.r.t. $\approx$. If $\Phi$ is a set of formulae in context $\Gamma$ then we write $A \models^{\approx} \Phi$ if $A \models^{\approx} \varphi$ for all $\varphi \in \Phi$.*

When $\approx$ is the indistinguishability relation (see Definition 4.1 below), $\models^{\approx}$ is known as *behavioural satisfaction*.

## 3.3 Relating $\models$ and $\models^{\approx}$

Let $\approx$ be a partial congruence on $A$.

**Definition 3.26** *Suppose $v \in [\![b]\!]_A$ for $b \in B$ such that $v \approx_b v$; then the* congruence class *of $v$ w.r.t. $\approx$ is defined as $[v]_{\approx_b} = \{v' \in [\![b]\!]_A \mid v \approx_b v'\}$. The* quotient *of $A$ by $\approx$, written $A/{\approx}$, is then defined as follows:*

$$\begin{aligned} [\![b]\!]_{A/\approx} &= \{[v]_{\approx_b} \mid v \in [\![b]\!]_A \text{ and } v \approx_b v\} \text{ for all } b \in B \\ [\![c]\!]_{A/\approx}([v_1]_{\approx_{b_1}}, \ldots, [v_n]_{\approx_{b_n}}) &= [[\![c]\!]_A(v_1, \ldots, v_n)]_{\approx_b} \text{ for all } c : b_1 \times \cdots \times b_n \to b \text{ in } C. \end{aligned}$$

Since $\approx$ is a partial congruence, the choice of representatives $v_1, \ldots, v_n$ in the definition of $[\![c]\!]_{A/\approx}$ doesn't matter. Note that if $\approx$ is a congruence, then $A/{\approx}$ is the usual quotient algebra, with $[\![b]\!]_{A/\approx} = [\![b]\!]_A / {\approx_b}$.

**Proposition 3.27** *$A/\!\approx$ is a $\Sigma$-algebra, that is $[\![c]\!]_{A/\approx} \in ([\![b_1]\!]_{A/\approx} \times \cdots \times [\![b_n]\!]_{A/\approx} \to [\![b]\!]_{A/\approx})$ for every $c : b_1 \times \cdots \times b_n \to b$ in $C$.*

Proof: *Easy, since if $v_1 \in [\![b_1]\!]_{A/\approx}, \ldots, v_n \in [\![b_n]\!]_{A/\approx}$ then $v_1 \approx_{b_1} v_1$ and $\cdots$ and $v_n \approx_{b_n} v_n$ so $[\![c]\!]_A(v_1, \ldots, v_n) \approx_b [\![c]\!]_A(v_1, \ldots, v_n)$.* □

The following theorem demonstrates a fundamental relationship between the two interpretations defined above. In the first-order case, it says that standard satisfaction of a formula $\varphi$ in a quotient algebra $A/\!\approx$ is equivalent to satisfaction of $\varphi$, with the symbol $=$ interpreted as $\approx$, in $A$ itself.

**Theorem 3.28** *$A/\!\approx\, \models \varphi$ iff $A \models^{\approx} \varphi$.*

Proof: *Define two families of functions $\langle \psi_\tau : [\![\tau]\!]^{\approx}_A \to [\![\tau]\!]_{A/\approx} \rangle_{\tau \in Types(\Sigma)}$ and $\langle \chi_\tau : [\![\tau]\!]_{A/\approx} \to [\![\tau]\!]^{\approx}_A \rangle_{\tau \in Types(\Sigma)}$ by induction on $\tau$ as follows:*

$$
\begin{array}{l}
\text{for all } v \in [\![b]\!]^{\approx}_A, \\
\qquad \psi_b(v) = [v]_{\approx_b} \\
\text{for all } v \in [\![b]\!]_{A/\approx}, \\
\qquad \chi_b(v) = \text{some arbitrary element of } v \\
\text{for all } p \in [\![[\tau_1, \ldots, \tau_n]]\!]^{\approx}_A, \\
\quad \psi_{[\tau_1,\ldots,\tau_n]}(p) = \{(v_1, \ldots, v_n) \in [\![\tau_1]\!]_{A/\approx} \times \cdots \times [\![\tau_n]\!]_{A/\approx} \mid (\chi_{\tau_1}(v_1), \ldots, \chi_{\tau_n}(v_n)) \in p\} \\
\text{for all } p \in [\![[\tau_1, \ldots, \tau_n]]\!]_{A/\approx}, \\
\quad \chi_{[\tau_1,\ldots,\tau_n]}(p) = \{(v_1, \ldots, v_n) \in [\![\tau_1]\!]^{\approx}_A \times \cdots \times [\![\tau_n]\!]^{\approx}_A \mid (\psi_{\tau_1}(v_1), \ldots, \psi_{\tau_n}(v_n)) \in p\}
\end{array}
$$

*These functions are well-defined. First, $\psi$ and $\chi_{[\tau_1,\ldots,\tau_n]}$ are not affected by the choice of $\chi_b$. Second, for all $v \in [\![\tau]\!]_{A/\approx}$, $\chi_\tau(v)$ respects $\approx$ (for base types $b$, we have $v \approx_b v$ for $v \in [\![b]\!]_{A/\approx}$; for bracket types, it follows from the fact that $v \approx_\tau v'$ implies $\psi_\tau(v) = \psi_\tau(v')$). We have for all $v \in [\![\tau]\!]_{A/\approx}$, $\psi_\tau(\chi_\tau(v)) = v$ and for all $v \in [\![\tau]\!]^{\approx}_A$, $\chi_\tau(\psi_\tau(v)) \approx_\tau v$, both by induction on the structure of $\tau$ (the latter uses again the fact that $v \approx_\tau v'$ implies $\psi_\tau(v) = \psi_\tau(v')$). The former implies that $\psi_\tau$ is onto for all $\tau$. Note that $\psi_{[]}$ is the identity.*

*The remainder of the proof relies on the following:*

> **Lemma** *For any $\Gamma \vdash t : \tau$ and $\Gamma$-environment $\rho$ w.r.t. $\approx$, $[\![t]\!]_{\psi\circ\rho, A/\approx} = \psi_\tau([\![t]\!]^{\approx}_{\rho,A})$.*
>
> Proof: *By induction on the structure of $t$. For function application, we use the fact that $\psi_b([\![c]\!]_A(v_1, \ldots, v_n)) = [\![c]\!]_{A/\approx}(\psi_{b_1}(v_1), \ldots, \psi_{b_n}(v_n))$ for $c : b_1 \times \cdots \times b_n \to b$ in $C$. For universal quantification, we use the fact that $\psi$ is onto.* □

*This gives $A/\!\approx\, \models \varphi \Longrightarrow A \models^{\approx} \varphi$, as follows. Suppose that $A/\!\approx\, \models \varphi$, i.e. $\Gamma \vdash \varphi : []$ and for all $\Gamma$-environments $\rho$ on $A/\!\approx$, $A/\!\approx\, \models_\rho \varphi$. Then suppose $\rho$ is a $\Gamma$-environment on $A$ w.r.t. $\approx$. We need to show that $A \models^{\approx}_\rho \varphi$. But $\psi \circ \rho$ is a $\Gamma$-environment on $A/\!\approx$, and so $[\![\varphi]\!]^{\approx}_{\rho,A} = \psi_{[]}([\![\varphi]\!]^{\approx}_{\rho,A}) = [\![\varphi]\!]_{\psi\circ\rho, A/\approx} = tt$.*

*Applying $\chi$ to both sides of the equation in the lemma gives $\chi_\tau([\![t]\!]_{\psi\circ\rho,A/\approx}) = \chi_\tau(\psi_\tau([\![t]\!]^{\approx}_{\rho,A})) \approx_\tau [\![t]\!]^{\approx}_{\rho,A}$. This gives $A/\!\approx\, \models \varphi \Longleftarrow A \models^{\approx} \varphi$, as follows. Suppose that $A \models^{\approx} \varphi$, i.e. $\Gamma \vdash \varphi : []$ and for all $\Gamma$-environments $\rho$ w.r.t. $\approx$, $A \models^{\approx}_\rho \varphi$. Then suppose $\rho$ is a $\Gamma$-environment on $A/\!\approx$. We need to show that $A/\!\approx\, \models_\rho \varphi$. Since $\psi$ is onto, $\rho$ factors through $\psi$: $\rho = \psi \circ \rho'$ for some $\Gamma$-environment $\rho'$ w.r.t. $\approx$. Thus $[\![\varphi]\!]_{\psi\circ\rho',A/\approx} = \chi_{[]}([\![\varphi]\!]_{\psi\circ\rho',A/\approx}) \approx_{[]} [\![\varphi]\!]^{\approx}_{\rho',A} = tt$, i.e. $A/\!\approx\, \models_\rho \varphi$.* □

A trivial consequence of Theorem 3.28 is the fact that when $\approx$ is equality, $\models^{\approx}$ coincides with $\models$. Theorem 3.28 for the case of first-order equational logic with reachability constraints is Theorem 3.17 of [BHW94], where the proof method is analogous.

We believe that the above development would do through, *mutatis mutandis*, for Henkin models [Hen50] as well as in a constructive framework like that of topos theory [Pho92]. In the absence of the axiom of choice, e.g. in topos theory, one must replace the function $\chi$ in the proof of Theorem 3.28 by a relation which is functional up to $\approx$.

# 4 Behavioural equivalence and indistinguishability

We now consider specific definitions of indistinguishability and behavioural equivalence. Let $\Sigma = \langle B, C\rangle$ be a signature, and let $OBS$, the *observable base types* of $\Sigma$, be a subset of $B$. The intention is that $OBS$ includes just those base types that are directly visible to clients; typically this would include types like *bool* and *nat*. All other types, including all bracket types, are *hidden* in the sense that their values may only be inspected indirectly by performing experiments (i.e. evaluating terms) that yield a result of a type in $OBS$.

The following defines the indistinguishability relation used in [NO88]. Two values $v, v'$ are indistinguishable if no experiment of observable type with additional observable inputs is able to distinguish between them.

**Definition 4.1** *Let the family of partial congruences* $\approx_{OBS} = \langle \approx_{OBS,A}\rangle_{A\in Alg(\Sigma)}$ *be such that for any* $\Sigma$*-algebra* $A$*, base type* $b \in B$ *and* $v, v' \in [\![b]\!]_A$*,* $v \approx_{OBS,A,b} v'$ *(*$v$ *and* $v'$ *are* indistinguishable*) iff* $v$ *and* $v'$ *are* $OBS$*-reachable, and for any* $OBS$*-context* $\Gamma$*, variable* $x \notin Vars(\Gamma)$*, term* $t$ *with* $\Gamma, x : b \vdash t : b'$ *for* $b' \in OBS$*, and* $\Gamma$*-environment* $\rho$*,* $[\![t]\!]_{\rho[x\mapsto v],A} = [\![t]\!]_{\rho[x\mapsto v'],A}$*.*

**Proposition 4.2** *For any* $\Sigma$*-algebra* $A$*,* $\approx_{OBS,A}$ *is a partial congruence on* $A$*.*

PROOF: *The proof that* $\approx_{OBS,A}$ *is preserved by the constants proceeds by induction on the structure of the "experiment term". The only non-trivial part of the proof is to show that* $[\![c]\!]_A(v_1,\ldots,v_n) = [\![c]\!]_A(v'_1,\ldots,v'_n)$ *for* $c : b_1 \times \cdots \times b_n \to b$ *in* $C$ *and* $b \in OBS$*, where* $v_1 \approx_{OBS,A,b_1} v'_1$ *and* $\cdots$ *and* $v_n \approx_{OBS,A,b_n} v'_n$*. This involves a chain of* $n$ *equations. The first of these is* $[\![c]\!]_A(v_1, v_2,\ldots,v_n) = [\![c]\!]_A(v'_1, v_2,\ldots,v_n)$ *which follows from* $v_1 \approx_{OBS,A,b_1} v'_1$ *by considering the term* $c(x, t_2,\ldots,t_n)$ *for terms* $t_2,\ldots,t_n$ *such that* $[\![t_j]\!]_{\rho,A} = v_j$ *for* $2 \le j \le n$*, where* $\rho$ *is a* $\Gamma$*-environment and* $\Gamma$ *is an* $OBS$*-context; these terms exist since* $v_2,\ldots,v_n$ *are* $OBS$*-reachable.* □

Note that each $\approx_{OBS,A}$ can be extended to a congruence by simply removing the requirement of $OBS$-reachability from Definition 4.1. This would not be an unreasonable modification but for the fact that then Proposition 4.9 and hence Theorem 5.15 below would not hold, meaning that the results in Section 6 would not be applicable.

By analogy with the terminology of denotational semantics (see e.g. [Win93]), a $\Sigma$-algebra $A$ is called *fully abstract* when the indistinguishability relation on $A$ is simply equality. Such an $A$ is called an *algebra of minimal redundancy* in [Rei85].

**Definition 4.3 ([BHW94])** *Let* $\approx\, = \langle\approx_A\rangle_{A\in Alg(\Sigma)}$ *be a family such that each* $\approx_A$ *is a partial congruence on* $A$*. A* $\Sigma$*-algebra* $A$ *is* $\approx$-fully abstract *when* $\approx_A$ *is the equality in* $A$*, that is, when for all* $b \in B$ *and* $v, v' \in [\![b]\!]_A$ *we have* $v \approx_A v'$ *iff* $v = v'$*. For any class* $\mathcal{A} \subseteq Alg(\Sigma)$ *of* $\Sigma$*-algebras,* $FA_{\approx}(\mathcal{A}) \subseteq \mathcal{A}$ *is the subclass of* $\approx$*-fully abstract algebras, that is:*

$$FA_{\approx}(\mathcal{A}) = \{A \in \mathcal{A} \mid A \text{ is } \approx\text{-fully abstract}\}.$$

*The family* $\approx$ *is* regular *if* $A/\approx_A$ *is* $\approx$*-fully abstract for every* $A \in Alg(\Sigma)$*.*

Regularity ensures that the partial congruences which $\approx$ associates with different algebras are related in a natural way.

**Proposition 4.4** $\approx_{OBS}$ *is regular.*

PROOF: *We have to show that for all* $b \in B$ *and* $v, v' \in [\![b]\!]_{A/\approx_{OBS,A}}$*,* $v \approx_{OBS,A/\approx_{OBS,A}} v'$ *iff* $v = v'$*.*

$\Longleftarrow$: *We only need to show that* $v(= v')$ *is* $OBS$*-reachable. This follows from the lemma in the proof of Theorem 3.28 and the fact that* $v$ *is a congruence class of* $OBS$*-reachable values in* $A$*.*

$\Longrightarrow$: *Suppose that* $v \approx_{OBS,A/\approx_{OBS,A}} v'$*. Then for any* $OBS$*-context* $\Gamma$*,* $x \notin Vars(\Gamma)$*, term* $t$ *such that* $\Gamma, x : b \vdash t : b'$ *for* $b' \in OBS$ *and* $\Gamma$*-environment* $\rho$ *on* $A/\approx_{OBS,A}$*,* $[\![t]\!]_{\rho[x\mapsto v],A/\approx_{OBS,A}} = [\![t]\!]_{\rho[x\mapsto v'],A/\approx_{OBS,A}}$*. Let* $\widehat{v}, \widehat{v'} \in [\![b]\!]_A$ *be such that* $[\widehat{v}]_{\approx_{OBS,A}} = v$ *and* $[\widehat{v'}]_{\approx_{OBS,A}} = v'$*; by the* *lemma in the proof of Theorem 3.28, it follows that* $[[\![t]\!]_{\rho'[x\mapsto\widehat{v}],A}]_{\approx_{OBS,A}} = [[\![t]\!]_{\rho'[x\mapsto\widehat{v'}],A}]_{\approx_{OBS,A}}$ *for any* $\Gamma$*-environment* $\rho'$ *on* $A$*, because for any* $\rho$ *there is a* $\rho'$ *such that* $\rho = [\cdot]_{\approx_{OBS,A}} \circ \rho'$*, and vice versa since* $\Gamma$ *is an* $OBS$*-context so* $\rho'(x)$ *is trivially* $OBS$*-reachable for any* $x \in Vars(\Gamma)$*. But* $\approx_{OBS,A}$ *on* $b' \in OBS$ *is equality so* $\widehat{v} \approx_{OBS,A} \widehat{v'}$*, i.e.* $v = v'$*.* □

We will now define what it means for two $\Sigma$-algebras to be behaviourally equivalent. The definition resembles that of indistinguishability in the sense that it is based on the idea of performing experiments to probe for differences between the two algebras. But in this case performing an experiment means testing satisfaction of a formula rather than evaluating a term of base type. The formulae of importance are equations between terms of observable type, with variables of observable type.

**Definition 4.5** *Let* $\Gamma$ *be an* $OBS$*-context. An* observable equation *is a formula in context* $\Gamma$ *of the form* $t =_b t'$ *where* $b \in OBS$*. Let* $ObsEq_\Gamma(\Sigma)$ *be the set of observable equations in context* $\Gamma$*.*

**Definition 4.6** *Let* $A, A' \in Alg(\Sigma)$*.* $A$ *is* behaviourally equivalent *to* $A'$ *(*via equations*), written* $A \equiv_{OBS} A'$*, if there is an* $OBS$*-context* $\Gamma$ *and* $\Gamma$*-environments* $\rho_A$ *on* $A$ *and* $\rho_{A'}$ *on* $A'$ *that are* $OBS$*-surjective such that for any equation* $\varphi \in ObsEq_\Gamma(\Sigma)$*,* $A \models_{\rho_A} \varphi$ *iff* $A' \models_{\rho_{A'}} \varphi$*.*

**Proposition 4.7** $\equiv_{OBS} \subseteq Alg(\Sigma) \times Alg(\Sigma)$ *is an equivalence relation.*

PROOF: *Reflexivity and symmetry are obvious. Transitivity follows from the observation that the choice of variable names in the context* $\Gamma$ *is arbitrary.* □

It might seem surprising that the definition of $\equiv_{OBS}$ does not make use of the higher-order features of the language, except as a result of the way that equality is expressed via quantification over predicates. So $\equiv_{OBS}$ is just the same as in e.g. [SW83], [MG85], [NO88]. The reason for this choice is that the natural modification of the definition of $\equiv_{OBS}$ to make use of higher-order formulae (Definition 5.13) gives exactly the same relation, see Corollary 5.16.

The following definition is the key to understanding the relationship between indistinguishability of values on the one hand and behavioural equivalence of algebras on the other. The idea is that a family of partial congruences naturally induces an equivalence on $Alg(\Sigma)$. If behavioural equivalence is the relation that is induced by indistinguishability (as will turn out to be the case, see Theorem 5.15) then it is possible to translate constructions phrased in terms of behavioural equivalence into constructions phrased in terms of indistinguishability, and vice versa. There is a close analogy with the case of finite state machines, where two machines $M, M'$ are equivalent if quotienting $M$ and $M'$ by the so-called *Nerode equivalence* on states yields isomorphic machines.

**Definition 4.8 ([BHW94])** *Let* $\approx\ = \langle\approx_A\rangle_{A\in Alg(\Sigma)}$ *be a family such that each* $\approx_A$ *is a partial congruence on* $A$*, and let* $\equiv\ \subseteq Alg(\Sigma) \times Alg(\Sigma)$ *be an equivalence relation. Then* $\equiv$ *is* factorizable *by* $\approx$ *if for any* $A, A' \in Alg(\Sigma)$*,* $A \equiv A'$ *iff* $A/{\approx_A} \cong A'/{\approx_{A'}}$*. (Factorizability can be decomposed as follows:* $\equiv$ *is* left-factorizable *by* $\approx$ *if for any* $A, A' \in Alg(\Sigma)$*,* $A \equiv A' \Longleftarrow A/{\approx_A} \cong A'/{\approx_{A'}}$*, and it is* right-factorizable *by* $\approx$ *if for any* $A, A' \in Alg(\Sigma)$*,* $A \equiv A' \Longrightarrow A/{\approx_A} \cong A'/{\approx_{A'}}$*.)*

The following proposition gives right-factorizability of $\equiv_{OBS}$ by $\approx_{OBS}$. Left-factorizability can be proved directly, but we obtain it instead by applying a more general result, see Corollary 5.11 below.

**Proposition 4.9** *For any* $A, A' \in Alg(\Sigma)$*, if* $A \equiv_{OBS} A'$ *then* $A/{\approx_{OBS,A}} \cong A'/{\approx_{OBS,A'}}$*.*

PROOF: *Suppose that* $A \equiv_{OBS} A'$ *via* $OBS$*-context* $\Gamma$ *and* $\Gamma$*-environments* $\rho_A$ *on* $A$ *and* $\rho_{A'}$ *on* $A'$*. The proof uses the following:*

> **Lemma** *Let* $t, t'$ *be terms such that* $\Gamma \vdash t : b$ *and* $\Gamma \vdash t' : b$*. Then* $[\![t]\!]_{\rho_A,A} \approx_{OBS,A,b} [\![t']\!]_{\rho_A,A}$ *iff* $[\![t]\!]_{\rho_{A'},A'} \approx_{OBS,A',b} [\![t']\!]_{\rho_{A'},A'}$*.*
>
> PROOF:

$\Longrightarrow$: *Suppose that* $[\![t]\!]_{\rho_A,A} \approx_{OBS,A,b} [\![t']\!]_{\rho_A,A}$. *We know that* $[\![t]\!]_{\rho_{A'},A'}$ *and* $[\![t']\!]_{\rho_{A'},A'}$ *are* $OBS$*-reachable since* $\Gamma$ *is an* $OBS$*-context. Let* $\Gamma'$ *be an* $OBS$*-context,* $x \notin Vars(\Gamma')$ *be a variable,* $s$ *be a term with* $\Gamma', x : b \vdash s : b'$ *for* $b' \in OBS$*, and* $\rho$ *be a* $\Gamma$*-environment on* $A'$. *We need to show that* $[\![s]\!]_{\rho[x \mapsto [\![t]\!]_{\rho_{A'},A'}],A'} = [\![s]\!]_{\rho[x \mapsto [\![t']\!]_{\rho_{A'},A'}],A'}$. *W.l.o.g. (since* $\rho_{A'}$ *is* $OBS$*-surjective) we can restrict attention to the case where* $\Gamma = \Gamma'$ *and* $\rho = \rho_{A'}$*; then by Proposition 3.5 it suffices to show that* $[\![s[x := t]]\!]_{\rho_{A'},A'} = [\![s[x := t']]\!]_{\rho_{A'},A'}$. *Now consider the observable equation* $\varphi =_{\rm def} s[x := t] =_{b'} s[x := t']$. *We have* $A \models_{\rho_A} \varphi$ *since* $[\![t]\!]_{\rho_A,A} \approx_{OBS,A,b} [\![t']\!]_{\rho_A,A}$*; then* $A' \models_{\rho_{A'}} \varphi$ *since* $A \equiv_{OBS} A'$. *Hence* $[\![s[x := t]]\!]_{\rho_{A'},A'} = [\![s[x := t']]\!]_{\rho_{A'},A'}$ *by Proposition 3.7.*

$\Longleftarrow$: *Similarly.* $\Box$

*Now define a function* $h : A/{\approx_{OBS,A}} \to A'/{\approx_{OBS,A'}}$ *by* $h([v]_{\approx_{OBS,A}}) = [[\![t]\!]_{\rho_{A'},A'}]_{\approx_{OBS,A'}}$ *for* $b \in B$ *and* $v \in [\![b]\!]_A$*, where* $t$ *is a term in context* $\Gamma$ *such that* $[\![t]\!]_{\rho_A,A} = v$. *We know that such a* $t$ *exists because* $v$ *is* $OBS$*-reachable by definition of* $\approx_{OBS,A}$ *and because* $\rho_A$ *is* $OBS$*-surjective. To see that the choice of the term* $t$ *and the representative* $v$ *don't matter, suppose we have terms* $t, t'$ *in context* $\Gamma$ *such that* $[\![t]\!]_{\rho_A,A} = v \approx_{OBS,A,b} v' = [\![t']\!]_{\rho_A,A}$*; then* $[\![t]\!]_{\rho_{A'},A'} \approx_{OBS,A',b} [\![t']\!]_{\rho_{A'},A'}$ *by the lemma. Thus* $h$ *is well-defined, and it is easy to see that* $h$ *is a* $\Sigma$*-homomorphism.*

*To see that* $h$ *is surjective, consider any* $b \in B$ *and representative* $v$ *of any congruence class in* $[\![b]\!]_{A'/\approx_{OBS,A'}}$. *Pick a term* $t$ *in context* $\Gamma$ *such that* $[\![t]\!]_{\rho_{A'},A'} = v$*; we know that such a* $t$ *exists because* $v$ *is* $OBS$*-reachable (since* $v \approx_{OBS,A'} v$*) and because* $\rho_{A'}$ *is* $OBS$*-surjective. Then* $[v]_{\approx_{OBS,A'}} = h([[\![t]\!]_{\rho_A,A}]_{\approx_{OBS,A}})$.

*To see that* $h$ *is injective, suppose* $b \in B$ *and* $v, v' \in [\![b]\!]_A$ *such that* $h([v]_{\approx_{OBS,A}}) = h([v']_{\approx_{OBS,A}})$*, where* $t, t'$ *are terms in context* $\Gamma$ *such that* $[\![t]\!]_{\rho_A,A} = v$ *and* $[\![t']\!]_{\rho_A,A} = v'$. *Then* $[\![t]\!]_{\rho_{A'},A'} \approx_{OBS,A'} [\![t']\!]_{\rho_{A'},A'}$*, so by the lemma we have* $[\![t]\!]_{\rho_A,A} \approx_{OBS,A} [\![t']\!]_{\rho_A,A}$*, i.e.* $[v]_{\approx_{OBS,A}} = [v']_{\approx_{OBS,A}}$.

*Thus we have shown that* $h : A/{\approx_{OBS,A}} \to A'/{\approx_{OBS,A'}}$ *is a* $\Sigma$*-isomorphism.* $\Box$

Proposition 4.9 essentially amounts to one direction of Example 3.25 of [BHW94], where the proof method is the same.

In this paper, we consider only the particular definitions of indistinguishability (Definition 4.1) and behavioural equivalence (Definition 4.6) given above. There are two other candidates for each of these definitions, as described in [BHW94]. The first variant, which has been studied by [Rei85], is obtained by allowing $\Gamma$ to be an arbitrary $B$-context in both definitions, removing the requirement of $OBS$-reachability in Definition 4.1, and changing the requirement of $OBS$-surjectivity to $B$-surjectivity in Definition 4.6. The second variant is obtained by eliminating the context $\Gamma$ and environments from both definitions, and changing the requirement of $OBS$-reachability to $\emptyset$-reachability in Definition 4.1; the resulting definition of behavioural equivalence has been studied in Section 2 of [ST87]. These alternatives are not studied here, although all of the proofs required should be similar to those given here. In our opinion, the first variant is simply wrong because the resulting behavioural equivalence relation fails to identify algebras that differ only in their behaviour on values of non-observable types that are not $OBS$-reachable: see [ONS91] for an example. The second variant seems to be unnecessarily restrictive in the presence of parameterised specifications, since (as discussed in [ST89]) $OBS$ will normally include the parameter types and these types typically lack generators; this leads to a behavioural equivalence relation that is too coarse.

## 5 Expressible congruences and relativization

The language of higher-order logic is powerful enough to express the indistinguishability relation $\approx_{OBS}$ by means of a family of predicates, i.e. terms in the language (cf. [Sch94]). We can use this fact to characterize behavioural satisfaction of a formula $\varphi$ in terms of ordinary satisfaction of a "relativized" version of $\varphi$.

Let $\Sigma = \langle B, C\rangle$ be a signature for which $B$ and $C$ are finite. The assumption of finiteness is required to obtain finite terms in Theorem 5.2 below.

Notations like $\bigforall_{b\in B} x_b{:}b.t$ and $\lambda(\langle P_b{:}[b,b]\rangle_{b\in B}).t$ will be used below to abbreviate obvious (finite) terms. The latter assumes some fixed enumeration of the elements of $B$; this is not needed for the former since a sequence of quantifiers can be permuted without affecting meaning.

**Definition 5.1** *Let* $\approx\, = \langle \approx_A\rangle_{A\in Alg(\Sigma)}$ *be a family of partial congruences, and let* $\sim\, = \langle \sim_b\rangle_{b\in B}$ *be a family of closed predicates such that* $\vdash \sim_b : [b,b]$ *for every base type* $b \in B$*. Then* $\approx$ *is* expressible by $\sim$ *if* $[\![\sim_b]\!]_{[],A} = \approx_{A,b}$ *for every* $b \in B$.

**Theorem 5.2** *The indistinguishability relation* $\approx_{OBS}$ *is expressible by the family of predicates* $\langle INDIST_{\widehat{b}}\rangle_{\widehat{b}\in B}$*, defined as follows:*

*If* $\widehat{b} \in OBS$ *then* $REACH_{\widehat{b}} =_{\mathrm{def}} \lambda(x{:}\widehat{b}).true$.

*If* $\widehat{b} \notin OBS$ *then*

$$REACH_{\widehat{b}} =_{\mathrm{def}}$$

$$\lambda(x{:}\widehat{b}).\bigforall_{b\notin OBS} P_b : [b].$$

$$\left(\bigwedge_{\substack{c:b_1\times\cdots\times b_n\to b' \text{ in } C\\ b'\notin OBS}} \ \bigforall_{1\le i\le n} x_i : b_i.\left(\bigwedge_{\substack{1\le j\le n\\ b_j\notin OBS}} P_{b_j}(x_j)\right) \Rightarrow P_{b'}(c(x_1,\ldots,x_n))\right)$$

$$\Rightarrow P_{\widehat{b}}(x)$$

$$CONG \quad =_{\mathrm{def}}$$

$$\lambda(\langle P_b{:}[b,b]\rangle_{b\in B}).$$

$$\bigwedge_{c:b_1\times\cdots\times b_n\to b \text{ in } C} \ \bigforall_{1\le i\le n} x_i, x'_i{:}b_i.\left(\bigwedge_{1\le j\le n} P_{b_j}(x_j,x'_j)\right) \Rightarrow P_b(c(x_1,\ldots,x_n),c(x'_1,\ldots,x'_n))$$

$$OBSEQ \quad =_{\mathrm{def}} \quad \lambda(\langle P_b{:}[b,b]\rangle_{b\in B}).\bigwedge_{b\in OBS} \forall x,x'{:}b.\ P_b(x,x') \Leftrightarrow x =_b x'$$

$$INDIST_{\widehat{b}} \quad =_{\mathrm{def}}$$

$$\lambda(x{:}\widehat{b}, y{:}\widehat{b}).\bigexists_{b\in B} P_b{:}[b,b].$$

$$CONG(\langle P_b\rangle_{b\in B}) \wedge OBSEQ(\langle P_b\rangle_{b\in B}) \wedge REACH_{\widehat{b}}(x) \wedge REACH_{\widehat{b}}(y) \wedge P_{\widehat{b}}(x,y)$$

PROOF: *Let $A$ be a $\Sigma$-algebra. We will use the following lemma.*

> **Lemma** *Suppose* $\widehat{v} \in [\![\widehat{b}]\!]_A$*. Then* $A \models_{[x\mapsto\widehat{v}]} REACH_{\widehat{b}}(x)$ *iff* $\widehat{v}$ *is OBS-reachable.*
>
> PROOF: *If* $\widehat{b} \in OBS$ *then the proof is trivial. So suppose that* $\widehat{b} \notin OBS$.
>
> $\Longrightarrow$: *Instantiate* $REACH_{\widehat{b}}(x)$ *with* $P_b = \{v \in [\![b]\!]_A \mid v \text{ is } OBS\text{-reachable}\} \in [\![[b]]\!]_A$ *for all* $b \notin OBS$*. It then suffices to show that the closure property on the left-hand side of the main implication is satisfied. This is easy: for each* $c : b_1 \times \cdots \times b_n \to b'$ *in* $C$*, the required term is simply the application of* $c$ *to the terms that witness the OBS-reachability of* $x_1,\ldots,x_n$.

$\Longleftarrow$: *Suppose that $\widehat{v}$ is OBS-reachable; then there is an OBS-context $\Gamma$, term $t$ with $\Gamma \vdash t : \widehat{b}$, and $\Gamma$-environment $\rho$ such that $[\![t]\!]_{\rho,A} = \widehat{v}$. We need to show that $A \models_{[x\mapsto\widehat{v}]} REACH_{\widehat{b}}(x)$; by Proposition 3.5 it suffices to show that $A \models_\rho REACH_{\widehat{b}}(t)$. This follows by induction on the structure of $t$.* $\square$

*We need to prove that if $\widehat{v}, \widehat{v'} \in [\![\widehat{b}]\!]_A$, then $A \models_{[x\mapsto\widehat{v},y\mapsto\widehat{v'}]} INDIST_{\widehat{b}}(x,y)$ iff $\widehat{v} \approx_{OBS,A,\widehat{b}} \widehat{v'}$.*

$\Longleftarrow$: *Suppose that $\widehat{v} \approx_{OBS,A,\widehat{b}} \widehat{v'}$. We claim that $A \models_{[x\mapsto\widehat{v},y\mapsto\widehat{v'}]} INDIST_{\widehat{b}}(x,y)$ with the predicates $P_b = \approx_{OBS,A,b} \in [\![[b,b]]\!]_A$ for all $b \in B$. By the assumption we have that $A \models_{[x\mapsto\widehat{v},y\mapsto\widehat{v'}]} P_{\widehat{b}}(x,y)$; then $A \models CONG(\langle P_b\rangle_{b\in B})$ since $\approx_{OBS,A}$ is a partial congruence on $A$ (Proposition 4.2), $A \models OBSEQ(\langle P_b\rangle_{b\in B})$ by the definition of $\approx_{OBS,A}$, and $A \models_{[x\mapsto\widehat{v},y\mapsto\widehat{v'}]} REACH_{\widehat{b}}(x) \wedge REACH_{\widehat{b}}(y)$ by the above lemma.*

$\Longrightarrow$: *Suppose that $A \models_{[x\mapsto\widehat{v},y\mapsto\widehat{v'}]} INDIST_{\widehat{b}}(x,y)$. Then $\widehat{v}$ and $\widehat{v'}$ are OBS-reachable by the above lemma. It remains to show that if $\Gamma$ is a OBS-context, $z \notin Vars(\Gamma)$, $s$ is a term such that $\Gamma, z : \widehat{b} \vdash s : b'$ for $b' \in OBS$, and $\rho$ is a $\Gamma$-environment, then $[\![s]\!]_{\rho[z\mapsto\widehat{v}],A} = [\![s]\!]_{\rho[z\mapsto\widehat{v'}],A}$. This is a consequence of the following lemma:*

> **Lemma** *For any OBS-context $\Gamma$, variable $z \notin Vars(\Gamma)$, term $s$ with $\Gamma, z : \widehat{b} \vdash s : b'$, and $\Gamma$-environment $\rho$, $([\![s]\!]_{\rho[z\mapsto\widehat{v}],A}, [\![s]\!]_{\rho[z\mapsto\widehat{v'}],A}) \in P_{b'}$.*
>
> Proof: *By induction on the structure of $s$. Suppose that $s$ is a variable but not $z$; then $b' \in OBS$ since $\Gamma$ is an OBS-context, and the required property follows from the fact that $A \models OBSEQ(\langle P_b\rangle_{b\in B})$. Suppose that $s$ is $z$; then the required property follows directly from the fact that $A \models_{[x\mapsto\widehat{v},y\mapsto\widehat{v'}]} P_{\widehat{b}}(x,y)$. Suppose that $s$ is a function application; then the required property follows from the inductive assumption and the fact that $A \models CONG(\langle P_b\rangle_{b\in B})$.* $\square$

*From this, together with the fact that $A \models OBSEQ(\langle P_b\rangle_{b\in B})$ and Proposition 3.7, it follows that $[\![s]\!]_{\rho[z\mapsto\widehat{v}],A} = [\![s]\!]_{\rho[z\mapsto\widehat{v'}],A}$ when $b' \in OBS$.* $\square$

In [Sch94] an analogous expressibility result for the indistinguishability relation used in [Rei85] is given for a language of second-order logic. Detailed comparisons are rendered difficult by the fact that the logic used there is untyped.

Let $\approx = \langle \approx_A \rangle_{A\in Alg(\Sigma)}$ be a family of partial congruences that is expressible by the family of predicates $\sim = \langle \sim_b \rangle_{b\in B}$.

Definition 3.14 showed how to extend a partial congruence to bracket types. We can express exactly the same thing for any expressible congruence.

**Proposition 5.3** *For any type $\tau$ there is a closed predicate $\sim_\tau$ such that $\vdash \sim_\tau : [\tau,\tau]$ and $[\![\sim_\tau]\!]_{[],A} = \approx_{A,\tau}$, given by the following definition:*

*If $\tau = b \in B$ then $\sim_\tau =_{\mathrm{def}} \sim_b$.*

*If $\tau = [\tau_1,\ldots,\tau_n]$ then*

$$\sim_\tau =_{\mathrm{def}}$$
$$\lambda(p{:}[\tau_1,\ldots,\tau_n], p'{:}[\tau_1,\ldots,\tau_n]).$$
$$\forall_{1\le i\le n} x_i, x'_i{:}\tau_i.\ \left(\bigwedge_{1\le j\le n} x_j \sim_{\tau_j} x'_j\right) \Rightarrow (p(x_1,\ldots,x_n) \Leftrightarrow p'(x'_1,\ldots,x'_n))$$

*(The definition of $\sim_{[\tau_1,\ldots,\tau_n]}$ is recursive, but the result is a finite term for any type $[\tau_1,\ldots,\tau_n]$.)*

Proof: *Immediate.* $\square$

This leads directly to a family of predicates characterizing the values that are in the interpretation of types w.r.t. $\approx$.

**Proposition 5.4** *For any type $\tau$ there is a closed predicate $DOM_\tau$ such that $\vdash DOM_\tau : [\tau]$ and $[\![DOM_\tau]\!]_{[],A} = [\![\tau]\!]_A^{\approx_A}$, given by the following definition:*

*If $\tau = b \in B$ then $DOM_\tau =_{\mathrm{def}} \lambda(x{:}b).x \sim_b x$.*

*If $\tau = [\tau_1, \ldots, \tau_n]$ then*

$$DOM_\tau =_{\mathrm{def}}$$
$$\lambda(p{:}[\tau_1,\ldots,\tau_n]).p \sim_{[\tau_1,\ldots,\tau_n]} p \wedge \bigvee_{1\le i\le n} x_i{:}\tau_i.\left(p(x_1,\ldots,x_n) \Rightarrow \bigwedge_{1\le j\le n} DOM_{\tau_j}(x_j)\right)$$

*(Again, this is a recursive definition that gives a finite term for any type.)*

Proof: *By induction on the structure of $\tau$, using Propositions 3.18 and 5.3.* $\square$

We can use the predicates $DOM_\tau$ thus defined to transform any formula $\varphi$ into a formula $\ulcorner\varphi\urcorner$ such that $\ulcorner\varphi\urcorner$ is satisfied exactly when $\varphi$ is satisfied w.r.t. $\approx$. The idea is simply to "relativize" each bound variable by attaching a requirement that the value taken on by the variable is in the interpretation of its type w.r.t. $\approx$.

**Definition 5.5** *Let $t$ be a term in context $\Gamma$. The* $\sim$-relativization *of $t$ is the term $\ulcorner t\urcorner$ (in context $\Gamma$) defined as follows:*

$$\begin{array}{rcl}
\ulcorner x\urcorner & = & x \\
\ulcorner c(t_1,\ldots,t_n)\urcorner & = & c(\ulcorner t_1\urcorner,\ldots,\ulcorner t_n\urcorner) \\
\ulcorner \lambda(x_1{:}\tau_1,\ldots,x_n{:}\tau_n).t\urcorner & = & \lambda(x_1{:}\tau_1,\ldots,x_n{:}\tau_n).DOM_{\tau_1}(x_1)\wedge\cdots\wedge DOM_{\tau_n}(x_n)\wedge\ulcorner t\urcorner \\
\ulcorner t(t_1,\ldots,t_n)\urcorner & = & \ulcorner t\urcorner(\ulcorner t_1\urcorner,\ldots,\ulcorner t_n\urcorner) \\
\ulcorner t\Rightarrow t'\urcorner & = & \ulcorner t\urcorner\Rightarrow\ulcorner t'\urcorner \\
\ulcorner \forall x{:}\tau.t\urcorner & = & \forall x{:}\tau.DOM_\tau(x)\Rightarrow\ulcorner t\urcorner
\end{array}$$

The following results relate satisfaction of a formula to satisfaction of its relativized version.

**Theorem 5.6** *Let $A$ be a $\Sigma$-algebra, $\Gamma \vdash t : \tau$ and let $\rho$ be a $\Gamma$-environment w.r.t. $\approx_A$ (so $\rho$ is also an ordinary $\Gamma$-environment by Proposition 3.18). Then $[\![t]\!]_{\rho,A}^{\approx_A} \approx_{A,\tau} [\![\ulcorner t\urcorner]\!]_{\rho,A}$.*

Proof: *By induction on the structure of the derivation of $\Gamma \vdash t : \tau$. For $\lambda$-abstraction, we use Propositions 3.22 and 5.4. For universal quantification, we use Proposition 5.4.* $\square$

**Corollary 5.7** *Let $A$ be a $\Sigma$-algebra, let $\varphi$ be a formula in context $\Gamma$ and let $\rho$ be a $\Gamma$-environment w.r.t. $\approx_A$. Then $A \models_\rho^{\approx_A} \varphi$ iff $A \models_\rho \ulcorner\varphi\urcorner$.*

Proof: *Immediate from Theorem 5.6.* $\square$

The definition of the $\sim$-relativization of a formula is closely related to the definition of "lifted" formula in [BH95], and Corollary 5.7 is a higher-order version of Theorem 15 there.

**Corollary 5.8** *Let $A, A'$ be $\Sigma$-algebras such that $A/{\approx_A} \cong A'/{\approx_{A'}}$, and let $\varphi$ be a closed formula. Then $A \models \ulcorner\varphi\urcorner$ iff $A' \models \ulcorner\varphi\urcorner$.*

Proof: *$A \models \ulcorner\varphi\urcorner$ iff $A \models^{\approx_A} \varphi$ (by Corollary 5.7, since $\varphi$ is closed) iff $A/{\approx_A} \models \varphi$ (by Theorem 3.28) iff $A'/{\approx_{A'}} \models \varphi$ (by Corollary 3.12) iff $A' \models^{\approx_{A'}} \varphi$ iff $A' \models \ulcorner\varphi\urcorner$.* $\square$

The relativization construction may be used to define another behavioural equivalence relation, in which two algebras are regarded as behaviourally equivalent provided they cannot be distinguished by relativized formulae. The motivation for this apparent departure from our earlier notion of behavioural equivalence is that it is a convenient technical device for proving left-factorizability of $\equiv_{OBS}$ by $\approx_{OBS}$ (Corollary 5.11), since this follows directly from left-factorizability of this new relation by $\approx_{OBS}$ (Theorem 5.10). In fact, it will turn out (Corollary 5.16) that this "new" relation coincides with $\equiv_{OBS}$.

**Definition 5.9** *Let $A, A' \in Alg(\Sigma)$. $A$ is* behaviourally equivalent *to $A'$* via relativized formulae, *written $A \equiv_{RelForm} A'$, if there is an OBS-context $\Gamma$ and $\Gamma$-environments $\rho_A$ on $A$ and $\rho_{A'}$ on $A'$ that are OBS-surjective such that for any formula $\varphi$ in context $\Gamma$, $A \models_{\rho_A} \ulcorner\varphi\urcorner$ iff $A' \models_{\rho_{A'}} \ulcorner\varphi\urcorner$, where $\ulcorner\varphi\urcorner$ is the $\langle INDIST_b\rangle_{b\in B}$-relativization of $\varphi$.*

**Theorem 5.10** *For any $A, A' \in Alg(\Sigma)$, if $A/{\approx_{OBS,A}} \cong A'/{\approx_{OBS,A'}}$ then $A \equiv_{RelForm} A'$.*

Proof: *Let $h : A/{\approx_{OBS,A}} \rightarrow A'/{\approx_{OBS,A'}}$ be an isomorphism. Since $\approx_{OBS}$ is equality on $b \in OBS$, we have a bijection $\widehat{h}_b : [\![b]\!]_A \rightarrow [\![b]\!]_{A'}$ for $b \in OBS$ defined by $[\widehat{h}_b(v)]_{\approx_{OBS,A',b}} = h_b([v]_{\approx_{OBS,A,b}})$ for $v \in [\![b]\!]_A$.*

*Let $\Gamma$ be the OBS-context such that $\Gamma_b = [\![b]\!]_A$ for every $b \in OBS$ (w.l.o.g. we assume that $[\![b]\!]_A \subseteq X$ and that $[\![b]\!]_A$ and $[\![b']\!]_A$ are disjoint for $b \neq b'$). Define an OBS-surjective $\Gamma$-environment $\rho_A$ on $A$ by $\rho_A(x) = x$. Define an OBS-surjective $\Gamma$-environment $\rho_{A'}$ on $A'$ by $\rho_{A'}(x) = \widehat{h}(x)$. Since $\approx_{OBS}$ is equality on $b \in OBS$, $\rho_A$ (resp. $\rho_{A'}$) is also a $\Gamma$-environment w.r.t. $\approx_{OBS}$ on $A$ (resp. $A'$). Let $\varphi$ be a formula in context $\Gamma$. Then $A \models_{\rho_A} \ulcorner\varphi\urcorner$ iff $A \models^{\approx_{OBS,A}}_{\rho_A} \varphi$ (by Corollary 5.7) iff $A/{\approx_{OBS,A}} \models_{\psi\circ\rho_A} \varphi$ (by the lemma in the proof of Theorem 3.28, where $\psi$ is the function from that proof, since $\psi_{[]}$ is the identity) iff $A'/{\approx_{OBS,A'}} \models_{h\circ\psi\circ\rho_A} \varphi$ (by Proposition 3.11, since $h_{[]}$ is the identity) iff $A'/{\approx_{OBS,A'}} \models_{\psi\circ\rho_{A'}} \varphi$ (since $h \circ \psi \circ \rho_A = \psi \circ \widehat{h} \circ \rho_A$) iff $A' \models^{\approx_{OBS,A'}}_{\rho_{A'}} \varphi$ iff $A' \models_{\rho_{A'}} \ulcorner\varphi\urcorner$.* □

**Corollary 5.11** *For any $A, A' \in Alg(\Sigma)$, if $A/{\approx_{OBS,A}} \cong A'/{\approx_{OBS,A'}}$ then $A \equiv_{OBS} A'$.*

Proof: *Observe that for any equation $\varphi \in ObsEq_\Gamma(\Sigma)$, $\varphi \models\!\!\mid \ulcorner\varphi\urcorner$ since the extra premise in $\ulcorner\varphi\urcorner$ always holds, by an easy argument involving Proposition 5.4. Then just apply Theorem 5.10.* □

Yet another definition of behavioural equivalence is obtained by extending the definition of $\equiv_{OBS}$ to take advantage of the availability of higher-order formulae to perform experiments.

**Definition 5.12** *A type $\tau$ is* observable *if either:*

- *$\tau$ is a base type that is in OBS; or*
- *$\tau = [\tau_1, \ldots, \tau_n]$ and $\tau_i$ is observable for all $1 \leq i \leq n$.*

*Let $\Gamma$ be a context. A term $t$ in context $\Gamma$ is* observation-restricted *if all types occurring in $t$ (i.e. as types of bound variables in $\lambda$-abstractions and universal quantifications) are observable. If $t$ is a formula and $\Gamma$ is an OBS-context then $t$ is called* observable*. Let $ObsForm_\Gamma(\Sigma)$ be the set of observable formulae in context $\Gamma$.*

Since predicates in formulae can only arise in two ways — via $\lambda$-abstraction and via quantification — the restrictions imposed on observable formulae ensure that predicates in such formulae always have observable type. Note that $ObsEq_\Gamma(\Sigma) \subset ObsForm_\Gamma(\Sigma)$.

**Definition 5.13** *Let $A, A' \in Alg(\Sigma)$. $A$ is* behaviourally equivalent *to $A'$* via formulae*, written $A \equiv_{OBSForm} A'$, if there is an OBS-context $\Gamma$ and $\Gamma$-environments $\rho_A$ on $A$ and $\rho_{A'}$ on $A'$ that are OBS-surjective such that for any formula $\varphi \in ObsForm_\Gamma(\Sigma)$, $A \models_{\rho_A} \varphi$ iff $A' \models_{\rho_{A'}} \varphi$.*

Left-factorizability of $\equiv_{OBSForm}$ by $\approx_{OBS}$ is another direct consequence of Theorem 5.10.

**Corollary 5.14** *For any $A, A' \in Alg(\Sigma)$, if $A/{\approx_{OBS,A}} \cong A'/{\approx_{OBS,A'}}$ then $A \equiv_{OBSForm} A'$.*

Proof: *For any observable type $\tau$, $v \approx_{OBS,A,\tau} v$ for any $v \in [\![\tau]\!]_A$, by induction on the structure of $\tau$ using Proposition 5.3. From this it follows that for any $\varphi \in ObsForm_\Gamma(\Sigma)$, $\varphi \models\!\!\mid \ulcorner\varphi\urcorner$. Then apply Theorem 5.10.* □

**Theorem 5.15** *$\equiv_{RelForm}$, $\equiv_{OBS}$ and $\equiv_{OBSForm}$ are factorizable by $\approx_{OBS}$.*

Proof:

$\equiv_{RelForm}$: *By Proposition 4.9 (since* $\equiv_{RelForm} \subseteq \equiv_{OBS}$ *by an argument like the one in Corollary 5.11) and Theorem 5.10.*

$\equiv_{OBS}$: *By Proposition 4.9 and Corollary 5.11.*

$\equiv_{OBSForm}$: *By Proposition 4.9 (since* $\equiv_{OBSForm} \subseteq \equiv_{OBS}$*) and Corollary 5.14.* □

It is an easy consequence of the above theorem that all three of our behavioural equivalence relations coincide. This demonstrates that using formulae more complex than equations as experiments does not allow finer distinctions between algebras to be made. This is not necessarily what one would expect: in the case of non-deterministic algebras, the use of more complex formulae does yield a different relation, see [Nip88].

**Corollary 5.16** $\equiv_{RelForm} = \equiv_{OBS} = \equiv_{OBSForm}$.

PROOF: *Immediate from Theorem 5.15 and the definition of factorizability.* □

# 6 Relating abstractor specifications and behavioural specifications

As discussed in the introduction, ordinary specifications consisting of a signature together with a set of axioms are not sufficiently abstract in that they sometimes describe classes of algebras that are not closed under behavioural equivalence. Two approaches to resolving this problem have been proposed. The first, due to [SW83], is to simply close the class of models of a specification under behavioural equivalence using an operation called *behavioural abstraction.* The second, due to [Rei85], is to take as models of a specification all those algebras that behaviourally satisfy the axioms. We provide syntax for all three kinds of specifications here in order to study how they are related.

**Definition 6.1** *A (*flat*)* specification *consists of a signature* $\Sigma$ *and a set* $\Phi$ *of closed* $\Sigma$*-formulae, called* axioms. *The* models *of a specification* $\langle\Sigma, \Phi\rangle$ *are all the algebras in the class*

$$Mod(\langle\Sigma, \Phi\rangle) = \{A \in Alg(\Sigma) \mid A \models \Phi\}.$$

Let $\langle\Sigma, \Phi\rangle$ be a specification. Let $\approx\, = \langle\approx_A\rangle_{A\in Alg(\Sigma)}$ be a family such that each $\approx_A$ is a partial congruence on $A$, and let $\equiv\, \subseteq Alg(\Sigma) \times Alg(\Sigma)$ be an equivalence relation.

**Definition 6.2** *For any class* $\mathcal{A} \subseteq Alg(\Sigma)$*, the* closure *of* $\mathcal{A}$ *under* $\equiv$ *is the class*

$$Abs_{\equiv}(\mathcal{A}) = \{A \in Alg(\Sigma) \mid A \equiv A' \textit{ for some } A' \in \mathcal{A}\}.$$

*When* $\equiv$ *is the relation* $\equiv_{OBS}$ *for some set OBS of base types, the operator* $Abs_{\equiv}$ *is known as* behavioural abstraction.

*A (*flat*)* abstractor specification*, written* abstract $\langle\Sigma, \Phi\rangle$ w.r.t. $\equiv$, *has as models all those* $\Sigma$*-algebras that are equivalent to models of* $\langle\Sigma, \Phi\rangle$*:*

$$Mod(\textsf{abstract } \langle\Sigma, \Phi\rangle \textsf{ w.r.t. } \equiv) = Abs_{\equiv}(Mod(\langle\Sigma, \Phi\rangle)).$$

**Definition 6.3** *A (*flat*)* behavioural specification*, written* behaviour $\langle\Sigma, \Phi\rangle$ w.r.t. $\approx$, *has as models all those* $\Sigma$*-algebras that satisfy the axioms* $\Phi$ *w.r.t.* $\approx$*:*

$$Mod(\textsf{behaviour } \langle\Sigma, \Phi\rangle \textsf{ w.r.t. } \approx) = \{A \in Alg(\Sigma) \mid A \models^{\approx_A} \Phi\}.$$

The notation used for behavioural specifications should not be confused with similar notation used in [SW83] and [ST87] for a particular special case of abstractor specifications.

We have now built up enough machinery to redo the development in [BHW94] in the framework of higher-order logic. Although it is not made explicit there, their results are independent of the logic used in axioms, provided properties corresponding to Corollary 3.12 and Theorem 3.28 hold for the logic of interest. In the remainder of this section we merely state the theorems and indicate dependencies; for proofs and discussion, see [BHW94].

The theorems below hold for arbitrary choices of $\approx$ and $\equiv$ that satisfy the following assumption:

**Assumption** *$\approx$ is regular and $\equiv$ is factorizable by $\approx$.*

The particular case of interest is where $\approx$ and $\equiv$ are $\approx_{OBS}$ and $\equiv_{OBS}$ respectively, for an arbitrary choice $OBS$ of observable base types. These satisfy the assumption by Proposition 4.4 and Theorem 5.15.

**Theorem 6.4 ([BHW94])** $Mod(\textsf{behaviour } \langle\Sigma,\Phi\rangle \textsf{ w.r.t. } \approx) = Abs_{\equiv}(FA_{\approx}(Mod(\langle\Sigma,\Phi\rangle)))$.

PROOF: *See [BHW94]. The proof depends on Corollary 3.12 and Theorem 3.28.* □

**Corollary 6.5 ([BHW94])** $Mod(\textsf{behaviour } \langle\Sigma,\Phi\rangle \textsf{ w.r.t. } \approx) \subseteq Mod(\textsf{abstract } \langle\Sigma,\Phi\rangle \textsf{ w.r.t. } \equiv)$. □

**Definition 6.6 ([BHW94])** *For any class $\mathcal{A} \subseteq Alg(\Sigma)$ of $\Sigma$-algebras, define:*

1. $Beh_{\approx}(\mathcal{A}) = Abs_{\equiv}(FA_{\approx}(\mathcal{A}))$.
2. $\mathcal{A}/\!\approx\, = \{A/\!\approx_A \mid A \in \mathcal{A}\}$.

**Theorem 6.7 ([BHW94])** $Mod(\textsf{abstract } \langle\Sigma,\Phi\rangle \textsf{ w.r.t. } \equiv) = Beh_{\approx}(Mod(\langle\Sigma,\Phi\rangle)/\!\approx)$.

PROOF: *See [BHW94]. The proof uses the fact that $A \equiv A/\!\approx_A$.* □

The main characterization theorem is the following:

**Theorem 6.8 ([BHW94])** *The following conditions are equivalent:*

1. $Mod(\textsf{behaviour } \langle\Sigma,\Phi\rangle \textsf{ w.r.t. } \approx) = Mod(\textsf{abstract } \langle\Sigma,\Phi\rangle \textsf{ w.r.t. } \equiv)$
2. $Mod(\langle\Sigma,\Phi\rangle) \subseteq Mod(\textsf{behaviour } \langle\Sigma,\Phi\rangle \textsf{ w.r.t. } \approx)$
3. $Mod(\langle\Sigma,\Phi\rangle)/\!\approx\, \subseteq Mod(\langle\Sigma,\Phi\rangle)$

PROOF: *See [BHW94]. The proof depends on Theorems 3.28, 6.4 and 6.7.* □

# 7 Reasoning about specifications

The results presented above serve to clarify our understanding of behavioural satisfaction and behavioural equivalence and the relationship between these in the context of higher-order logic. A concrete benefit of this is a number of methods for reasoning about specifications, as will be summarized below. Some of these appear in a different form in [BH95] or elsewhere, while others (Proof Methods 7.14, 7.16 and 7.17) are new.

We begin by introducing some (mostly standard) concepts and notation. Let $\Sigma = \langle B, C\rangle$ be a signature. In this section we restrict attention to closed formulae.

**Definition 7.1** *A closed formula $\varphi$ is a* consequence *of a set $\Phi$ of closed formulae, written $\Phi \models \varphi$, if for any $\Sigma$-algebra $A$, $A \models \Phi$ implies $A \models \varphi$.*

When reasoning about a specification $SP$, our goal is to discover whether or not a given formulae $\varphi$ is satisfied by all models of $SP$. Let $\approx\, = \langle\approx_A\rangle_{A\in Alg(\Sigma)}$ be a family of partial congruences. A related goal is that of discovering whether or not $\varphi$ is satisfied w.r.t. $\approx$ by all models of $SP$. These questions amount to determining whether or not $\varphi$ is in the *theory* (resp. *theory w.r.t.* $\approx$) of $SP$.

**Definition 7.2** *Let $\mathcal{A} \subseteq Alg(\Sigma)$ be a class of $\Sigma$-algebras. The* theory w.r.t. $\approx$ *of $\mathcal{A}$ is the set $Th_{\approx}(\mathcal{A}) = \{\varphi \mid A \models^{\approx_A} \varphi$ for every $A \in \mathcal{A}\}$. The* (ordinary) theory *of $\mathcal{A}$ is the set $Th(\mathcal{A}) = \{\varphi \mid A \models \varphi$ for every $A \in \mathcal{A}\}$; note that $Th(\mathcal{A}) = Th_{=}(\mathcal{A})$. If $SP$ is a specification, we write $Th(SP)$ for $Th(Mod(SP))$ and $Th_{\approx}(SP)$ for $Th_{\approx}(Mod(SP))$.*

The essence of reasoning about specifications is to find a way of reducing the problems of determining $\varphi \in Th(SP)$ and $\varphi \in Th_{\approx}(SP)$ to that of consequence ($\Phi \models \psi$ for appropriate $\Phi$ and $\psi$); then any proof system that is sound for $\models$ may be used to finish the job. For the ordinary theory of a flat specification, the reduction is trivial: $\varphi \in Th(\langle\Sigma, \Phi\rangle)$ iff $\Phi \models \varphi$. For the theory w.r.t. $\approx$ and for behavioural specifications and abstractor specifications, the problem is much more difficult. We consider each case below, giving proof methods that provide such reductions.

Let $\approx$ be expressible by the family of predicates $\sim = \langle\sim_b\rangle_{b\in B}$, and let $\equiv \subseteq Alg(\Sigma) \times Alg(\Sigma)$ be an equivalence relation.

## 7.1 $\varphi \in Th_{\approx}(\langle\Sigma, \Phi\rangle)$

This is the problem that is studied in [BH95], where it is argued that a solution to this problem provides the basis of a strategy for proving correctness of implementation steps in stepwise refinement of specifications (cf. [BH94b] and "abstractor" implementations in [ST88]).

The following proof method follows immediately from Corollary 5.7:

**Proof Method 7.3** *$\varphi \in Th_{\approx}(\langle\Sigma, \Phi\rangle)$ iff $\Phi \models \ulcorner\varphi\urcorner$.* □

This is essentially the same as the solution proposed in [BH95], except that because the analogue of our Corollary 5.7 there involves infinitary formulae, more work is required to reduce the problem to one of consequence for finitary formulae.

Alternatively, if Theorem 6.8 applies, then this problem is equivalent to the problem treated in Section 7.3 below according to the following result:

**Proposition 7.4 ([BHW94])** *If $\equiv$ is factorizable by $\approx$ then $Th_{\approx}(Abs_{\equiv}(\mathcal{A})) = Th_{\approx}(\mathcal{A})$.*

Proof: *See [BHW94]. The proof depends on Corollary 3.12 and Theorem 3.28.* □

In this case, Proof Methods 7.7–7.13 below are also applicable.

## 7.2 $\varphi \in Th(\textsf{behaviour } \langle\Sigma, \Phi\rangle \textsf{ w.r.t. } \approx)$

This problem can be reduced to ordinary consequence by applying the following easy consequence of Corollary 5.7:

**Proposition 7.5** *$Mod(\textsf{behaviour } \langle\Sigma, \Phi\rangle \textsf{ w.r.t. } \approx) = Mod(\langle\Sigma, \ulcorner\Phi\urcorner\rangle)$ where $\ulcorner\Phi\urcorner = \{\ulcorner\varphi\urcorner \mid \varphi \in \Phi\}$.* □

This leads to the following proof method:

**Proof Method 7.6** *$\varphi \in Th(\textsf{behaviour } \langle\Sigma, \Phi\rangle \textsf{ w.r.t. } \approx)$ iff $\ulcorner\Phi\urcorner \models \varphi$.* □

If $\approx$ is regular and $\equiv$ is factorizable by $\approx$, then any behavioural specification is at least as restrictive as the corresponding abstractor specification by Corollary 6.5. Thus, under these conditions the proof methods in Section 7.4 below (i.e. Proof Methods 7.14–7.17) may be soundly applied to this problem.

## 7.3 $\varphi \in Th_{\approx}(\textsf{behaviour } \langle\Sigma, \Phi\rangle \textsf{ w.r.t. } \approx)$

This is the problem that is studied in [BH94a], for a specific indistinguishability relation different from $\approx_{OBS}$.

Corollary 5.7 and Proposition 7.5 yield the following proof method:

**Proof Method 7.7** *$\varphi \in Th_{\approx}(\textsf{behaviour } \langle\Sigma, \Phi\rangle \textsf{ w.r.t. } \approx)$ iff $\ulcorner\Phi\urcorner \models \ulcorner\varphi\urcorner$.* □

Another approach, which appears to be more powerful, is obtained by appealing to the following results:

**Proposition 7.8 ([BHW94])** *If* $\equiv$ *is factorizable by* $\approx$ *then* $Th_{\approx}(Abs_{\equiv}(FA_{\approx}(\mathcal{A}))) = Th(FA_{\approx}(\mathcal{A}))$.

PROOF: *By Proposition 7.4 and the definition of fully abstract algebra.* □

**Proposition 7.9 ([BH95])** $FA_{\approx}(Mod(\langle\Sigma,\Phi\rangle)) = Mod(\langle\Sigma, \Phi \cup \{\forall x, y{:}b.(x \sim_b y \Leftrightarrow x =_b y) \mid b \in B\}\rangle)$.

PROOF: *Directly from the definition of fully abstract algebra.* □

These together with Theorem 6.4 yield the following:

**Proof Method 7.10** *Suppose that* $\approx$ *is regular and* $\equiv$ *is factorizable by* $\approx$*. Then* $\varphi \in Th_{\approx}(\textsf{behaviour}\ \langle\Sigma,\Phi\rangle\ \textsf{w.r.t.}\ \approx)$ *iff* $\Phi \cup \{\forall x, y{:}b.(x \sim_b y \Leftrightarrow x =_b y) \mid b \in B\} \models \varphi$. □

This is essentially the same as the method proposed in [BH95], with the proviso concerning infinitary formulae mentioned earlier.

It is worth pointing out that a weaker but very simple and potentially useful consequence of this is the following:

**Proof Method 7.11** *Suppose that* $\approx$ *is regular and* $\equiv$ *is factorizable by* $\approx$*. Then* $\varphi \in Th_{\approx}(\textsf{behaviour}\ \langle\Sigma,\Phi\rangle\ \textsf{w.r.t.}\ \approx)$ *if* $\Phi \models \varphi$. □

Finally, a more direct approach to this problem is to reduce it trivially to consequence w.r.t. $\approx$:

**Definition 7.12** *A closed formula* $\varphi$ *is a* consequence *of a set* $\Phi$ *of closed formulae* w.r.t. $\approx$*, written* $\Phi \models^{\approx} \varphi$*, if for any* $\Sigma$*-algebra* $A$*,* $A \models^{\approx_A} \Phi$ *implies* $A \models^{\approx_A} \varphi$.

**Proof Method 7.13** $\varphi \in Th_{\approx}(\textsf{behaviour}\ \langle\Sigma,\Phi\rangle\ \textsf{w.r.t.}\ \approx)$ *iff* $\Phi \models^{\approx} \varphi$. □

Then what is required to finish the job is a proof system that is sound for $\models^{\approx}$. See [Rei85], where a proof system for conditional equational logic is given that is sound for an indistinguishability relation different from $\approx_{OBS}$, in the context of partial algebras; see also [HW93].

## 7.4 $\varphi \in Th(\textsf{abstract}\ \langle\Sigma,\Phi\rangle\ \textsf{w.r.t.}\ \equiv)$

This is the problem that is of importance for reasoning about specifications in a language like ASL [SW83] that includes a specification-building operation corresponding to abstract; cf. [Far92].

If Theorem 6.8 applies, this problem can be reduced to the problem treated in Section 7.2 above. Then Proof Method 7.6 is applicable.

Alternatively, if the formula to be proved is a relativized formula or is logically equivalent to such a formula, Corollary 5.8 yields the following reduction.

**Proof Method 7.14** *Suppose that* $\equiv$ *is factorizable by* $\approx$ *and* $\varphi \models\!\mid \ulcorner\psi\urcorner$ *for some closed formula* $\psi$*. Then* $\Phi \models \varphi$ *implies* $\varphi \in Th(\textsf{abstract}\ \langle\Sigma,\Phi\rangle\ \textsf{w.r.t.}\ \equiv)$.

PROOF: *Suppose* $A \in Mod(\textsf{abstract}\ \langle\Sigma,\Phi\rangle\ \textsf{w.r.t.}\ \equiv)$*, i.e. there is some algebra* $A'$ *such that* $A' \in Mod(\langle\Sigma,\Phi\rangle)$ *and* $A \equiv A'$*, and* $\Phi \models \varphi$*. But then* $A' \models \varphi$*, so* $A' \models \ulcorner\psi\urcorner$*, and then* $A \models \ulcorner\psi\urcorner$ *(by factorizability and Corollary 5.8) so* $A \models \varphi$. □

This is a direct extension of the method for reasoning about abstractor specifications presented in Section 4 of [ST87], which applies only to formulae built in certain ways from observable equations. By analogy with an observation there, Proof Method 7.14 is not confined to inferring formulae that are equivalent to relativized formulae. In order to validly conclude that $\varphi \in Th(\textsf{abstract}\ \langle\Sigma,\Phi\rangle\ \textsf{w.r.t.}\ \equiv)$, it is enough to have a proof of $\Phi \models \varphi$ for which there is a "horizontal cut" containing only formulae that are equivalent to relativized formulae. Similar remarks apply to the proof methods presented below. A formula that is equivalent to a relativized formula is called a "$\approx$-invariant" formula in [BH95], but this concept is not used as the basis of a reasoning method there.

A useful special case of Proof Method 7.14 can be obtained by adding "respectful" abstraction $\lambda^{\mathrm{r}}$ and quantification $\forall^{\mathrm{r}}$ to the syntax, where:

$$\begin{array}{rll} \lambda^{\mathrm{r}}(x_1{:}\tau_1,\ldots,x_n{:}\tau_n).t & \text{abbreviates} & \lambda(x_1{:}\tau_1,\ldots,x_n{:}\tau_n).DOM_{\tau_1}(x_1) \wedge \cdots \wedge DOM_{\tau_n}(x_n) \wedge t \\ \forall^{\mathrm{r}} x{:}\tau.t & \text{abbreviates} & \forall x{:}\tau.DOM_{\tau}(x) \Rightarrow t \end{array}$$

**Definition 7.15** *A* respectful formula *is a formula that may contain* $\lambda^{\mathrm{r}}$ *and/or* $\forall^{\mathrm{r}}$ *but does not contain* $\lambda$ *or* $\forall$*.*

It is easy to see that $\varphi \models\!\mid \ulcorner\varphi\urcorner$ for any respectful formula $\varphi$. This gives the following.

**Proof Method 7.16** *Suppose that* $\equiv$ *is factorizable by* $\approx$ *and* $\varphi$ *is a closed respectful formula. Then* $\Phi \models \varphi$ *implies* $\varphi \in Th(\textsf{abstract}\ \langle\Sigma, \Phi\rangle\ \textsf{w.r.t.}\ \equiv)$*.* □

In the case of behavioural abstraction, note that $\forall^{\mathrm{r}}$ on base types corresponds exactly to reachable quantification as in [Sch92]. Also, since every observable formula amounts to a respectful formulae (since respectful abstraction and quantification over observable types is equivalent to ordinary abstraction and quantification), we have the following:

**Proof Method 7.17** *Suppose that* $\varphi$ *is a closed observable formula. Then* $\Phi \models \varphi$ *implies* $\varphi \in Th(\textsf{abstract}\ \langle\Sigma, \Phi\rangle\ \textsf{w.r.t.}\ \equiv_{OBS})$*.* □

In Section 5 of [Sch92], Schoett highlights an inadequacy in the method for reasoning about abstractor specifications presented in [ST87]. He gives a simple abstractor specification with axioms in first-order equational logic and a property that it satisfies, and shows that an infinite number of applications of the proof method in [ST87] would be required in a proof of that property. This particular example is easily dealt with using Proof Method 7.16: the required property can be expressed using higher-order respectful quantifiers and proved in the unabstracted specification, whereupon a single application of the proof method completes the proof.

### 7.5 $\varphi \in Th_{\approx}(\textsf{abstract}\ \langle\Sigma, \Phi\rangle\ \textsf{w.r.t.}\ \equiv)$

If $\equiv$ is factorizable by $\approx$ then this problem is equivalent to the problem treated in Section 7.1 above according to Proposition 7.4. Thus Proof Method 7.3 is applicable. If Theorem 6.8 applies then Proof Methods 7.7–7.13 from Section 7.3 are applicable as well.

## 8 Further work

One of our reasons for studying behavioural semantics of specifications with higher-order formulae as axioms was the desire to apply the results in the Extended ML framework for the formal development of ML programs from specifications [ST89], [KST94]. The characterization results and reasoning methods are of direct relevance in this context: the interpretation of Extended ML interfaces involves abstractor specifications, and the logical system used for writing axioms is (a form of) higher-order logic. However, the framework presented here needs to be extended in two directions to make the match a perfect one.

First, the framework needs to be generalized to allow functions of higher type as in [Mei92], in addition to the predicates of higher type that are already present. The most obvious examples which require the use of behavioural semantics in the context of higher-order logic (e.g. in Extended ML) use such functions. Such a generalization would involve adding constants of higher type to signatures and allowing $\lambda$-abstraction to be used for forming functions as well as predicates. This can be done, as we will describe in a future paper; the generalization is not straightforward because Proposition 3.18 does not hold if we extend Definition 3.17 with obvious choices for $[\![\tau \to \tau']\!]^{\approx}_{A}$, such as $\{f : [\![\tau]\!]^{\approx}_{A} \to [\![\tau']\!]^{\approx}_{A} \mid \forall u, v \in [\![\tau]\!]^{\approx}_{A}. u \approx_{A,\tau} v \Rightarrow f(u) \approx_{A,\tau'} f(v)\}$. Furthermore, for functions to be of real use there must be a way of constructing them from "specifications", e.g. recursive definitions. Therefore, one needs a unique choice operator or (in the absence of a syntax for proofs) a general choice operator $\varepsilon$ as in HOL [GM93]. The presence of such a choice operator again poses non-trivial albeit surmountable problems. But note that $n$-ary functions may already be coded as $(n+1)$-ary predicates in the usual way, and that this coding extends to functions of higher type.

Second, the use of $\textsf{behaviour}$ and $\textsf{abstract}$ in the context of structured specifications built using operations like $\textsf{enrich}$ and $\textsf{derive}$ needs to be studied. An attempt at this appears in [BHW94], where the extension of behaviour to structured specifications is a *post hoc* construction on the class of models of the underlying specification:

$$Mod(\textsf{behaviour } SP \textsf{ w.r.t. } \approx) = Beh_{\approx}(Mod(SP))$$

Unless $SP$ is a flat specification, the result that this produces is different from what is obtained when the specification-building operations in $SP$ are interpreted in the usual way but with axioms in $SP$ satisfied according to $\models^{\approx}$ rather than $\models$. Further work is required to clarify the relationship between abstractor specifications (which generalize easily to structured specifications) and this alternative interpretation of behavioural specifications.

Applying the results and proof methods to concrete examples should shed considerable light. Without having attempted such examples, we are not yet in a position to understand the tradeoffs between the various proof methods that may be applicable in a particular situation. But in view of the size and complexity of the predicates $INDIST_{\widehat{b}}$ in Theorem 5.2, it seems clear that proof methods that involve the direct manipulation of relativized formulae will not be convenient for use in practice when $\approx$ is the indistinguishability relation $\approx_{OBS}$. Here, a promising avenue is the search for more tractable predicates which correctly express $\approx_{OBS}$ in restricted circumstances (cf. the notion of "conditional axiomatization" in [BH95]). Proof methods which make no use of the predicates $INDIST_{\widehat{b}}$ (e.g. Proof Methods 7.13 and 7.17) do not suffer from this problem.

**Acknowledgements:** Thanks to Michel Bidoit and Rolf Hennicker for many very useful comments, including an explanation of how [BH95] relates to concepts and results in Sections 5 and 7. Proof Method 7.10 is due to them, and they pointed out that a previous version of Proof Methods 7.14 and 7.16 were unnecessarily restrictive. Thanks to Andrzej Tarlecki for many discussions on related topics and for drawing our attention to the idea behind the predicate $INDIST_{\widehat{b}}$ in Theorem 5.2. Thanks to David Aspinall for helpful comments on a draft of this paper, and to Wolfgang Degen for providing useful pointers to the literature.

# References

[BH94a] M. Bidoit and R. Hennicker. Proving behavioural theorems with standard first-order logic. *Proc. 4th Intl. Conf. on Algebraic and Logic Programming*, Madrid. Springer LNCS 850 (1994).

[BH94b] M. Bidoit and R. Hennicker. Proving the correctness of behavioural implementations. Draft report, Ecole Normale Supérieure (1994).

[BH95] M. Bidoit and R. Hennicker. Behavioural theories. *Selected Papers from the 10th Workshop on Specification of Abstract Data Types*, Santa Margherita Ligure. Springer LNCS, to appear (1995).

[BHW94] M. Bidoit, R. Hennicker and M. Wirsing. Behavioural and abstractor specifications. Report LIENS-94-10, Ecole Normale Supérieure (1994). To appear in *Science of Computer Programming*. A short version appeared as: Characterizing behavioural semantics and abstractor semantics. *Proc. 5th European Symp. on Programming*, Edinburgh. Springer LNCS 788, 105–119 (1994).

[Far92] J. Farrés-Casals. Verification in ASL and Related Specification Languages. Ph.D. thesis, Report CSR-92-92, Univ. of Edinburgh (1992).

[GGM76] V. Giarratana, F. Gimona and U. Montanari. Observability concepts in abstract data type specification. *Proc. 1976 Symp. on Mathematical Foundations of Computer Science*, Gdansk. Springer LNCS 45, 567–578 (1976).

[GM82] J. Goguen and J. Meseguer. Universal realization, persistent interconnection and implementation of abstract modules. *Proc. 9th Intl. Colloq. on Automata, Languages and Programming*, Aarhus. Springer LNCS 140, 265–281 (1982).

[GM93] M. Gordon and T. Melham. *Introduction to HOL: a theorem proving environment for higher-order logic.* Cambridge Univ. Press (1993).

[Hen50] L. Henkin. Completeness in the theory of types. *Journal of Symbolic Logic* 15:81–91 (1950).

[Hen91] R. Hennicker. Context induction: a proof principle for behavioural abstractions and algebraic implementations. *Formal Aspects of Computing* 4:326–345 (1991).

[HW93] R. Hennicker and M. Wirsing. Behavioural specifications. In: *Proof and Computation*. Marktoberdorf International Summer School, 1993. NATO ASI Series F, Vol. 139. Springer, to appear.

[KST94] S. Kahrs, D. Sannella and A. Tarlecki. The semantics of Extended ML: a gentle introduction. *Proc. Intl. Workshop on Semantics of Specification Languages*, Utrecht, 1993. Springer Workshops in Computing, 186–215 (1994).

[Mei92] K. Meinke. Universal algebra in higher types. *Theoretical Computer Science* 100:385–417 (1992).

[MG85] J. Meseguer and J. Goguen. Initiality, induction and computability. In: *Algebraic Methods in Semantics* (M. Nivat and J. Reynolds, eds.). Cambridge Univ. Press, 459–540 (1985).

[Mit90] J. Mitchell. Type systems for programming languages. In *Handbook of Theoretical Computer Science, Vol. B* (J. van Leeuwen, ed.), 365–458. North Holland (1990).

[Nip88] T. Nipkow. Observing nondeterministic data types. *Selected Papers from the 5th Workshop on Specification of Abstract Data Types*, Gullane. Springer LNCS 332, 170–183 (1988).

[NO88] P. Nivela and F. Orejas. Initial behaviour semantics for algebraic specifications. *Selected Papers from the 5th Workshop on Specification of Abstract Data Types*, Gullane. Springer LNCS 332, 184–207 (1988).

[ONS91] F. Orejas, M. Navarro and A. Sánchez. Implementation and behavioural equivalence: a survey. *Selected Papers from the 8th Workshop on Specification of Abstract Data Types*, Dourdan. Springer LNCS 655, 93–125 (1991).

[Pho92] W. Phoa. An introduction to fibrations, topos theory, the effective topos and modest sets. Report ECS-LFCS-92-208, Univ. of Edinburgh (1992).

[Rei85] H. Reichel. Behavioural validity of conditional equations in abstract data types. *Proc. of the Vienna Conf. on Contributions to General Algebra*, 1984. Teubner-Verlag, 301–324 (1985).

[ST87] D. Sannella and A. Tarlecki. On observational equivalence and algebraic specification. *Journal of Computer and System Sciences* 34:150–178 (1987).

[ST88] D. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: implementations revisited. *Acta Informatica* 25:233–281 (1988).

[ST89] D. Sannella and A. Tarlecki. Toward formal development of ML programs: foundations and methodology. *Joint Conf. on Theory and Practice of Software Development*, Barcelona. Springer LNCS 352, 375–389 (1989).

[ST92] D. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: model-theoretic foundations. *Proc. 19th Intl. Colloq. on Automata, Languages and Programming*, Vienna. Springer LNCS 623, 656–671 (1992).

[SW83] D. Sannella and M. Wirsing. A kernel language for algebraic specification and implementation. *Proc. 1983 Intl. Conf. on Foundations of Computation Theory*, Borgholm. Springer LNCS 158, 413–427 (1983).

[Sch92] O. Schoett. Two impossibility theorems on behavioural specification of abstract data types. *Acta Informatica* 29:595–621 (1992).

[Sch94] P.-Y. Schobbens. Second-order proof systems for algebraic specification languages. *Selected Papers from the 9th Workshop on Specification of Abstract Data Types*, Caldes de Malavella. Springer LNCS 785, 321–336 (1994).

[Sch77] K. Schütte. *Proof Theory*. Springer (1977).

[Sti92] C. Stirling. Modal and temporal logics for processes. Report ECS-LFCS-92-221, Univ. of Edinburgh (1992). To appear in: *Proc. of the VIII Banff Higher Order Workshop*, Springer Workshops in Computing (1995).

[Win93] G. Winskel. *The Formal Semantics of Programming Languages*. MIT Press (1993).