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

Martin Hofmann** and Donald Sannella***

Laboratory for Foundations of Computer Science
University of Edinburgh, Edinburgh EH9 3JZ

Abstract. The behavioural semantics of specifications with higher-order formulae as axioms is analyzed. A characterization of behavioural abstraction via behavioural satisfaction of formulae in which the equality symbol is interpreted as indistinguishability, due to Reichel and 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*, which seems to appropriately capture the "black box" character of data abstractions, see e.g. [GM82], [ST87]. Roughly speaking, two Σ-algebras A, B are behaviourally equivalent with respect to a set OBS of *observable types* if all computations that can be expressed in Σ and that yield a result in OBS produce the same result in both A and B. 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 means of a specification-building operation known as *behavioural abstraction* [SW83], [ST87]. The term "behavioural semantics" is used to characterize approaches that take the need for behavioural closure into account.

One issue in behavioural semantics is the relationship between the class of algebras produced by applying behavioural abstraction to a specification $\langle \Sigma, \Phi \rangle$, and that obtained by simply interpreting equality in Φ as *indistinguishability* rather than as identity. The latter approach, sometimes known as *behavioural satisfaction*, is due to Reichel [Rei85] who showed that these two classes coincide when the axioms involved are conditional equations, provided that the

* This is a condensed version of [HS95].

** 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.

conditions used are equations between terms of types in *OBS*. Bidoit *et al* have recently generalized this 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 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, provided that indistinguishability is *regular* and that behavioural equivalence is *factorizable* by indistinguishability.

We examine these issues for the case of (flat) specifications with higher-order logical formulae as axioms, generalizing 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. We give syntax and semantics for higher-order formulae and show that they have the required properties. We then define behavioural equivalence and indistinguishability and prove regularity and factorizability, which leads directly to a characterization result analogous to the one in [BHW94].

Higher-order logic provides sufficient power to express the indistinguishability relation as a predicate. We apply this fact to develop methods for reasoning about specifications under behavioural semantics. We characterize behavioural satisfaction in terms of ordinary satisfaction, by giving a translation that takes any formula φ to a "relativized" formula $\ulcorner\varphi\urcorner$ such that $\ulcorner\varphi\urcorner$ is satisfied exactly when φ is behaviourally satisfied. This, together with the characterization theorem, leads directly to various proof methods.

For reasons of space, proofs are omitted in this paper. See [HS95] for these and for additional material.

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.

Definition 2.1 *A* signature Σ *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* $c : b_1 \times \cdots \times b_n \to b$.

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

Definition 2.2 *The* types *over* Σ *are given by the grammar* $\tau ::= b \mid [\tau_1, \ldots, \tau_n]$*, where* $b \in B$ *and* $n \geq 0$*.* $Types(\Sigma)$ *denotes the set of all types over* Σ.

A type $[\tau_1, \ldots, \tau_n]$ is the type of n-ary predicates taking arguments of types $\tau_1, \ldots, \tau_n$. The type $[\,]$ may be thought of as **Prop**, the type of propositions. Let X be a fixed infinite set of variables, ranged over by x.

Definition 2.3 *The* terms *over* Σ *are given by the following grammar:*

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

where $c \in C$ and $n \geq 0$. As usual, we regard α-convertible terms as equal.

Function application $c(t_1, \ldots, t_n)$ is distinguished from predicate application $t(t_1, \ldots, t_n)$. λ-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 are not distinguished syntactically from terms denoting predicates or propositions. 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 Γ *is a sequence* $x_1 : \tau_1, \ldots, x_n : \tau_n$ *where* $x_i \neq x_j$ *for all* $i \neq j$. *We write* $\Gamma(x_j)$ *for* τ_j *and* $Vars(\Gamma)$ *for* $\{x_1, \ldots, x_n\}$, *and we identify* Γ *with the* $Types(\Sigma)$*-sorted set of variables such that* $\Gamma_\tau = \{x \in Vars(\Gamma) \mid \Gamma(x) = \tau\}$ *for all* $\tau \in Types(\Sigma)$. *Let* $T \subseteq Types(\Sigma)$ *be a subset of the set of types over* Σ; *then* Γ *is called a* T-context *if* $\Gamma(x) \in T$ *for all* $x \in Vars(\Gamma)$.

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

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

$$\frac{\Gamma, x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : [\,]}{\Gamma \vdash \lambda(x_1{:}\tau_1, \ldots, x_n{:}\tau_n).t : [\tau_1, \ldots, \tau_n]} \qquad \frac{\Gamma \vdash t : [\,] \quad \Gamma \vdash t' : [\,]}{\Gamma \vdash t \Rightarrow t' : [\,]}$$

$$\frac{c : b_1 \times \cdots \times b_n \to b \quad \Gamma \vdash t_1 : b_1 \quad \cdots \quad \Gamma \vdash t_n : b_n}{\Gamma \vdash c(t_1, \ldots, t_n) : b} \qquad \frac{\Gamma, x : \tau \vdash t : [\,]}{\Gamma \vdash \forall x{:}\tau.t : [\,]}$$

Equality is expressible using higher-order quantification. That is, suppose $\Gamma \vdash t : \tau$ and $\Gamma \vdash t' : \tau$; then

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

where $P \notin Vars(\Gamma)$. Existential quantification and the missing connectives are expressible as usual in terms of $\forall$ and $\Rightarrow$:

true abbreviates $\forall P{:}[\,].P \Rightarrow P$	$\varphi \vee \varphi'$ abbreviates $(\neg\varphi) \Rightarrow \varphi'$
false abbreviates $\forall P{:}[\,].P$	$\varphi \wedge \varphi'$ abbreviates $\neg(\neg\varphi \vee \neg\varphi')$
$\neg\varphi$ abbreviates $\varphi \Rightarrow$ *false*	$\exists x{:}\tau.\varphi$ abbreviates $\neg\forall x{:}\tau.\neg\varphi$

Finally, there is no need to treat reachability constraints as a special case, since induction principles are expressible. For example, the following formula (call it *GENNAT*) asserts that *nat* is generated by 0 and *succ*:

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

See [HS95] for an example which gives a taste of the expressive power of the language thus defined.

3 Semantics of higher-order logic

Let $\Sigma = \langle B, C \rangle$ be a signature. Terms over Σ are interpreted in the context of a Σ-algebra which gives meaning to the base types and the constants in Σ.

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

Let A be a Σ-algebra. We define two interpretations for terms. The first is the obvious "standard" interpretation and the second is modulo a partial congruence relation on A. In the latter interpretation, quantification 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 to be defined in Sect. 4. Theorem 3.19 demonstrates a relationship between the two interpretations that will be crucial in the sequel. Our use of *partial* congruences stems from the need to establish an appropriate relationship between indistinguishability and behavioural equivalence, see Theorem 5.12, in order to apply the characterization theorems in Sect. 6.

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 Γ be a context.

Definition 3.3 *A* Γ-environment *(on* A*) is a* $Types(\Sigma)$*-sorted function* $\rho = \langle \rho_\tau : \Gamma_\tau \to [\![\tau]\!]_A \rangle_{\tau \in Types(\Sigma)}$. *The notation* $\rho[x_1 \mapsto v_1, \ldots, x_n \mapsto v_n]$ *denotes the environment* ρ *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* Γ*-environment* ρ *is* T-surjective *if* $\rho_\tau : \Gamma_\tau \to [\![\tau]\!]_A$ *is surjective for each* $\tau \in T$.

Definition 3.4 *Let* ρ *be a* Γ*-environment. The interpretation of constants is extended to terms in context* Γ *as follows:*

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

Proposition 3.5 *If $\Gamma \vdash t : \tau$ and ρ is a Γ-environment then $[\![t]\!]_{\rho,A} \in [\![\tau]\!]_A$.* □

The following proposition demonstrates that $=_\tau$ really is equality.

Proposition 3.6 *Suppose $v, v' \in [\![\tau]\!]_A$ for some type τ. Then for any environment ρ, $[\![x =_\tau y]\!]_{\rho[x\mapsto v, y\mapsto v'],A} = tt$ iff $v = v'$.* □

Definition 3.7 *Let $B' \subseteq B$. A value $v \in [\![b]\!]_A$ is* B'-reachable *if there is a B'-context Γ, a term t with $\Gamma \vdash t : b$, and a Γ-environment ρ, 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 that the formula *GENNAT* asserts that *nat* is generated by 0 and *succ*. Indeed, for any algebra A over the relevant signature, $[\![GENNAT]\!]_{[],A} = tt$ iff every value of type *nat* in A is $\emptyset$-reachable.

Definition 3.8 *Let φ, φ' be formulae in context Γ. Let ρ be a Γ-environment; we write $A \models_\rho \varphi$ if $[\![\varphi]\!]_{\rho,A} = tt$. We write $A \models \varphi$ (A* satisfies *φ) if $A \models_\rho \varphi$ for all Γ-environments ρ. We write $\varphi \models\!\models \varphi'$ (φ is* equivalent to *φ') if for all $A \in Alg(\Sigma)$ and all Γ-environments ρ, $A \models_\rho \varphi$ iff $A \models_\rho \varphi'$. Finally, if Φ is a set of formulae in context Γ then we write $A \models \Phi$ if $A \models \varphi$ for all $\varphi \in \Phi$.*

Proposition 3.9 *If $A \cong A'$ then $A \models \varphi$ iff $A' \models \varphi$.* □

3.2 Interpretation w.r.t. a partial congruence

Definition 3.10 *A* partial congruence *$\approx$ on A is a family of* partial equivalence 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 $v_j, v_j' \in [\![b_j]\!]_A$ such that $v_j \approx_{b_j} v_j'$ for $1 \le j \le n$, $[\![c]\!]_A(v_1, \ldots, v_n) \approx_b [\![c]\!]_A(v_1', \ldots, v_n')$. A* (total) congruence *is a reflexive partial congruence.*

Let $\approx$ be a partial congruence on A. We now generalise the definition of satisfaction up to $\approx$ in first-order equational logic to higher-order logic. Whereas in the first-order case it is enough to interpret the equality symbol as the partial congruence and to restrict all quantifiers to 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" $\approx$. That our definition is the right generalisation is shown by Prop. 3.16 and Theorem 3.19.

The following definition extends the partial congruence $\approx$ to a so-called *logical relation* over all types. The resulting relation will be used below to interpret bracket types.

Definition 3.11 *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 $v_j, v_j' \in [\![\tau_j]\!]_A$ such that $v_j \approx_{\tau_j} v_j'$ for $1 \le j \le n$, $(v_1, \ldots, v_n) \in p$ 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 $v \approx_{[]} v'$ iff $v = v'$.

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

Definition 3.12 *Interpretation of types w.r.t. $\approx$ is defined as follows:*

$$[\![b]\!]^{\approx}_A = \{v \in [\![b]\!]_A \mid v \text{ respects } \approx\}$$
$$[\![[\tau_1, \ldots, \tau_n]]\!]^{\approx}_A = \{p \in Pow([\![\tau_1]\!]^{\approx}_A \times \cdots \times [\![\tau_n]\!]^{\approx}_A) \mid p \text{ respects } \approx\}$$

We have $[\![[\,]]\!]^{\approx}_A = [\![[\,]]\!]_A = \{ff, tt\}$. The second clause of the above definition is well-formed because $[\![\tau]\!]^{\approx}_A \subseteq [\![\tau]\!]_A$ for any type τ. Let Γ be a context.

Definition 3.13 *A Γ-environment* (w.r.t. $\approx$, on A) *is a $Types(\Sigma)$-sorted function $\rho = \langle \rho_\tau : \Gamma_\tau \to [\![\tau]\!]^{\approx}_A \rangle_{\tau \in Types(\Sigma)}$.*

Definition 3.14 *Let ρ be a Γ-environment w.r.t. $\approx$. The interpretation w.r.t. $\approx$ of terms in context Γ is defined as follows:*

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

Proposition 3.15 *If $\Gamma \vdash t : \tau$ and ρ is a Γ-environment w.r.t. $\approx$ then $[\![t]\!]^{\approx}_{\rho,A} \in [\![\tau]\!]^{\approx}_A$.* □

The following shows that $=_\tau$ refers to $\approx$ under the interpretation w.r.t. $\approx$. This is because P in $\forall P{:}[\tau].P(t) \Rightarrow P(t')$ ranges over predicates that respect $\approx$.

Proposition 3.16 *Suppose $v, v' \in [\![\tau]\!]^{\approx}_A$ for some type τ. Then for any environment ρ w.r.t. $\approx$, $[\![x =_\tau y]\!]^{\approx}_{\rho[x \mapsto v, y \mapsto v'],A} = tt$ iff $v \approx_\tau v'$.* □

The interpretation of *GENNAT* with respect to $\approx$ is also different from what it was under the standard interpretation. For any algebra A, $[\![GENNAT]\!]^{\approx}_{[],A} = tt$ iff every value in $[\![nat]\!]^{\approx}_A$ is congruent to a $\emptyset$-reachable value.

Definition 3.17 *Let φ be a formula in context Γ. Suppose ρ is a Γ-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 *φ* w.r.t. $\approx$*) if $A \models^{\approx}_\rho \varphi$ for all Γ-environments ρ w.r.t. $\approx$. If Φ is a set of formulae in context Γ 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 3.18 *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:*

$$[\![b]\!]_{A/\approx} = \{[v]_{\approx_b} \mid v \in [\![b]\!]_A \text{ and } v \approx_b v\}$$

$$[\![c]\!]_{A/\approx}([v_1]_{\approx_{b_1}}, \ldots, [v_n]_{\approx_{b_n}}) = [[\![c]\!]_A(v_1, \ldots, v_n)]_{\approx_b}$$

The following demonstrates a fundamental relationship between $\models$ and $\models^{\approx}$. In the first-order case, it says that standard satisfaction of a formula φ in $A/\approx$ is equivalent to satisfaction of φ, with the symbol $=$ interpreted as $\approx$, in A itself.

Theorem 3.19 *$A/\approx \models \varphi$ iff $A \models^{\approx} \varphi$.* □

A trivial consequence of this is the fact that $\models^{=}$ coincides with $\models$.

4 Behavioural equivalence and indistinguishability

Let $\Sigma = \langle B, C \rangle$ be a signature, and let OBS, the *observable base types* of Σ, be a subset of B. The intention is that OBS includes just those base types that are directly visible to clients. All other types, including 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 indistinguishability as in [NO88]; $v \approx_{OBS} v'$ if no experiment of observable type with observable inputs can 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 $v \approx_{OBS,A,b} v'$ ($v, v' \in [\![b]\!]_A$ are* indistinguishable*) iff v and v' are OBS-reachable, and for any OBS-context Γ, variable $x \notin Vars(\Gamma)$, term t with $\Gamma, x : b \vdash t : b'$ for $b' \in OBS$, and Γ-environment ρ, $[\![t]\!]_{\rho[x \mapsto v],A} = [\![t]\!]_{\rho[x \mapsto v'],A}$.*

By analogy with the terminology of denotational semantics, a Σ-algebra A is called *fully abstract* when the indistinguishability relation on A is equality. Such an A is called an *algebra of minimal redundancy* in [Rei85].

Definition 4.2 ([BHW94]) *Let $\approx = \langle \approx_A \rangle_{A \in Alg(\Sigma)}$ be a family of partial congruences. A Σ-algebra A is* $\approx$-fully abstract *when for all $v, v' \in [\![b]\!]_A$, $v \approx_A v'$ iff $v = v'$. For any $\mathcal{A} \subseteq Alg(\Sigma)$, define*

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

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

Proposition 4.3 *$\approx_{OBS}$ is regular.* □

We now define what it means for two Σ-algebras to be behaviourally equivalent. The definition resembles that of indistinguishability in that it is based on the idea of experiments. But in this case performing an experiment means testing satisfaction of a formula rather than evaluating a term of base type.

Definition 4.4 *An* observable equation *is a formula $t =_b t'$ in OBS-context Γ where $b \in OBS$. $ObsEq_\Gamma(\Sigma)$ is the set of observable equations in Γ.*

Definition 4.5 *A is* behaviourally equivalent *to A', written $A \equiv_{OBS} A'$, if there is an OBS-context Γ and OBS-surjective Γ-environments ρ_A on A and $\rho_{A'}$ on A' such that for any $\varphi \in ObsEq_\Gamma(\Sigma)$, $A \models_{\rho_A} \varphi$ iff $A' \models_{\rho_{A'}} \varphi$.*

Note that the definition of $\equiv_{OBS}$ does not make use of higher-order features, 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]. But the natural modification of the definition of $\equiv_{OBS}$ to make use of higher-order formulae gives exactly the same relation, see Corollary 5.13.

The following definition is the key to understanding the relationship between indistinguishability and behavioural equivalence. The idea is that a family of partial congruences naturally induces an equivalence on $Alg(\Sigma)$. Since behavioural equivalence is the relation induced by indistinguishability (Theorem 5.12), we can 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 4.6 ([BHW94]) *Let $\approx = \langle \approx_A \rangle_{A \in Alg(\Sigma)}$ be a family of partial congruences, 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 $A/\approx_A \cong A'/\approx_{A'}$.*

The following proposition gives half of factorizability of $\equiv_{OBS}$ by $\approx_{OBS}$. The other half is a consequence of a more general result, see Corollary 5.8 below.

Proposition 4.7 *If $A \equiv_{OBS} A'$ then $A/\approx_{OBS,A} \cong A'/\approx_{OBS,A'}$.* □

In this paper, we consider only the definitions of indistinguishability and behavioural equivalence given above. There are two other candidates for each of these, as described in [BHW94]. These alternatives are not studied here, although the results given here should hold for them as well.

5 Expressible congruences and relativization

Higher-order logic is powerful enough to express directly the indistinguishability relation $\approx_{OBS}$. Let $\Sigma = \langle B, C \rangle$ be such that B and C are finite.

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 a family of predicates $\langle INDIST_b \rangle_{b \in B}$.* □

In [Sch94] an analogous expressibility result for the indistinguishability relation used in [Rei85] is given for a language of second-order logic.

Let $\approx = \langle \approx_A \rangle_{A \in Alg(\Sigma)}$ be expressible by $\sim = \langle \sim_b \rangle_{b \in B}$. We can use $\sim$ to give predicates characterizing the values in the interpretation of types w.r.t. $\approx$.

Proposition 5.3 *For any type τ there is a closed predicate DOM_τ such that $\vdash DOM_\tau : [\tau]$ and $[\![DOM_\tau]\!]_{[],A} = [\![\tau]\!]_A^{\approx_A}$.* □

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

Definition 5.4 *Let t be a term in context Γ. The* $\sim$-relativization *of t is the term $\ulcorner t \urcorner$ (in context Γ) defined as follows:*

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

Theorem 5.5 *Let A be a Σ-algebra, let φ be a formula in context Γ and let ρ be a Γ-environment w.r.t. $\approx_A$. Then $A \models_\rho^{\approx_A} \varphi$ iff $A \models_\rho \ulcorner \varphi \urcorner$.* □

The $\sim$-relativization of a formula is similar to the notion of "lifted" formula in [BH95], and Theorem 5.5 is a higher-order version of Theorem 15 there.

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. It will turn out (Corollary 5.13) that this "new" relation coincides with $\equiv_{OBS}$.

Definition 5.6 *A is* behaviourally equivalent *to A'* via relativized formulae, *written $A \equiv_{RelForm} A'$, if there is an OBS-context Γ and OBS-surjective Γ-environments ρ_A on A and $\rho_{A'}$ on A' such that for any formula φ in context Γ, $A \models_{\rho_A} \ulcorner \varphi \urcorner$ iff $A' \models_{\rho_{A'}} \ulcorner \varphi \urcorner$, where $\ulcorner \varphi \urcorner$ is the $\langle INDIST_b \rangle_{b \in B}$-relativization of φ.*

Theorem 5.7 *If $A/\approx_{OBS,A} \cong A'/\approx_{OBS,A'}$ then $A \equiv_{RelForm} A'$.* □

Corollary 5.8 *If $A/\approx_{OBS,A} \cong A'/\approx_{OBS,A'}$ then $A \equiv_{OBS} A'$.* □

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

Definition 5.9 *A type τ is* observable *if either:*

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

A formula φ *in OBS-context* Γ *is* observable *if all types occurring in* φ *(i.e. as types of bound variables) are observable. Let* $ObsForm_\Gamma(\Sigma)$ *be the set of observable formulae in context* Γ.

The restrictions on observable formulae ensure that predicates in such formulae always have observable type. Note that $ObsEq_\Gamma(\Sigma) \subset ObsForm_\Gamma(\Sigma)$.

Definition 5.10 *A is* behaviourally equivalent *to* A' via formulae, *written* $A \equiv_{OBSForm} A'$, *if there is an OBS-context* Γ *and OBS-surjective* Γ*-environments* ρ_A *on* A *and* $\rho_{A'}$ *on* A' *such that for any* $\varphi \in ObsForm_\Gamma(\Sigma)$, $A \models_{\rho_A} \varphi$ *iff* $A' \models_{\rho_{A'}} \varphi$.

Corollary 5.11 *If* $A/{\approx_{OBS,A}} \cong A'/{\approx_{OBS,A'}}$ *then* $A \equiv_{OBSForm} A'$. □

Theorem 5.12 $\equiv_{RelForm}$, $\equiv_{OBS}$ *and* $\equiv_{OBSForm}$ *are factorizable by* $\approx_{OBS}$. □

Corollary 5.13 $\equiv_{RelForm} = \equiv_{OBS} = \equiv_{OBSForm}$. □

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].

6 Relating abstractor and behavioural specifications

Definition 6.1 *A (*flat*)* specification *consists of a signature* Σ *and a set* Φ *of closed* Σ*-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 of partial congruences, and let $\equiv \subseteq Alg(\Sigma) \times Alg(\Sigma)$ be an equivalence relation.

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

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

When $\equiv$ *is the relation* $\equiv_{OBS}$ *for some OBS,* $Abs_\equiv$ *is known as* behavioural abstraction. *An* abstractor specification, *written* abstract $\langle \Sigma, \Phi \rangle$ w.r.t. $\equiv$, *has as models all those* Σ*-algebras that are equivalent to models of* $\langle \Sigma, \Phi \rangle$*:*

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

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

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

We have now built up enough machinery to redo the development in [BHW94] for higher-order logic. Although it is not explicit there, their results are independent of the logic used in axioms, provided properties corresponding to Prop. 3.9 and Theorem 3.19 hold. We merely state the theorems; for proofs and discussion, see [BHW94].

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 Prop. 4.3 and Theorem 5.12.

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

Definition 6.5 ([BHW94]) *For any $\mathcal{A} \subseteq Alg(\Sigma)$, define the classes $Beh_{\approx}(\mathcal{A}) = Abs_{\equiv}(FA_{\approx}(\mathcal{A}))$ and $\mathcal{A}/\approx = \{A/\approx_A \mid A \in \mathcal{A}\}$.*

Theorem 6.6 ([BHW94])
$Mod(\text{abstract } \langle \Sigma, \Phi \rangle \text{ w.r.t. } \equiv) = Beh_{\approx}(Mod(\langle \Sigma, \Phi \rangle)/\approx)$. □

The main characterization theorem is the following:

Theorem 6.7 ([BHW94]) *The following conditions are equivalent:*

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

7 Reasoning about specifications

A concrete benefit of the results above is a number of methods for reasoning about specifications. Let $\Sigma = \langle B, C \rangle$ be a signature. Let $\approx = \langle \approx_A \rangle_{A \in Alg(\Sigma)}$ be a family of partial congruences that is expressible by $\sim = \langle \sim_b \rangle_{b \in B}$, and let $\equiv \subseteq Alg(\Sigma) \times Alg(\Sigma)$ be an equivalence relation. We restrict attention to closed formulae.

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

Definition 7.2 *Let $\mathcal{A} \subseteq Alg(\Sigma)$. The* theory w.r.t. $\approx$ *of $\mathcal{A}$ is $Th_{\approx}(\mathcal{A}) = \{\varphi \mid A \models^{\approx_A} \varphi$ for every $A \in \mathcal{A}\}$. The* (ordinary) theory *of $\mathcal{A}$ is $Th(\mathcal{A}) = Th_{=}(\mathcal{A})$. 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 $\varphi \in Th(SP)$ and $\varphi \in Th_{\approx}(SP)$ to that of consequence ($\Phi \models \psi$ for appropriate Φ and ψ); then any proof system that is sound for $\models$ may be used to finish the job. We consider the most important cases below, giving proof methods that provide such reductions. See [HS95] for the remaining cases.

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

It is argued in [BH95] that a solution to this problem can be used to prove correctness of implementation steps in stepwise refinement.

The following proof method follows immediately from Theorem 5.5:

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

This is essentially the solution proposed in [BH95], except that because the analogue of our Corollary 5.5 there involves infinitary formulae, more work is required to reduce the problem to one of consequence for finitary formulae.

Alternatively, if Theorem 6.7 applies, then this problem is equivalent to the problem treated in Sect. 7.2 below according to the following result:

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

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

This problem is studied in [BH94], for the indistinguishability relation of [Rei85].

Theorem 5.5 yields the following proof method:

Proof Method 7.5 $\varphi \in Th_{\approx}$(behaviour $\langle \Sigma, \Phi \rangle$ w.r.t. $\approx$) *iff* $\ulcorner \Phi \urcorner \models \ulcorner \varphi \urcorner$. □

A more powerful approach is obtained via the following results:

Proposition 7.6 ([BHW94]) $Th_{\approx}(Abs_{\equiv}(FA_{\approx}(\mathcal{A}))) = Th(FA_{\approx}(\mathcal{A}))$ *if* $\equiv$ *is factorizable by* $\approx$. □

Proposition 7.7 ([BH95]) $Mod(\langle \Sigma, \Phi \cup \{\forall x, y{:}b.(x \sim_b y \Leftrightarrow x =_b y) \mid b \in B\}\rangle) = FA_{\approx}(Mod(\langle \Sigma, \Phi \rangle))$. □

These together with Theorem 6.4 yield the following:

Proof Method 7.8 *Suppose that* $\approx$ *is regular and* $\equiv$ *is factorizable by* $\approx$. *Then* $\varphi \in Th_{\approx}$(behaviour $\langle \Sigma, \Phi \rangle$ 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 method proposed in [BH95], with the proviso concerning infinitary formulae mentioned earlier.

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

Definition 7.9 *A formula* φ *is a* consequence *of a set* Φ *of formulae* w.r.t. $\approx$, *written* $\Phi \models^{\approx} \varphi$, *if for any* Σ*-algebra* A, $A \models^{\approx_A} \Phi$ *implies* $A \models^{\approx_A} \varphi$.

Proof Method 7.10 $\varphi \in Th_{\approx}$(behaviour $\langle \Sigma, \Phi \rangle$ 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 $\models^{\approx}$.

7.3 $\varphi \in Th(\textsf{abstract } \langle \Sigma, \Phi \rangle \textsf{ w.r.t. } \equiv)$

This problem arises in reasoning about specifications in ASL [SW83] which includes a specification-building operation corresponding to abstract; cf. [Far92].

Theorems 5.5 and 6.7 yield the following:

Proof Method 7.11 *Suppose that* $\approx$ *is regular and* $\equiv$ *is factorizable by* $\approx$*, and the conditions in Theorem 6.7 hold for the specification* $\langle \Sigma, \Phi \rangle$*. Then* $\varphi \in Th(\textsf{abstract } \langle \Sigma, \Phi \rangle \textsf{ w.r.t. } \equiv)$ *iff* $\ulcorner \Phi \urcorner \models \varphi$*.* □

Alternatively, if the formula to be proved is a relativized formula or is logically equivalent to such a formula, we obtain the following reduction.

Proof Method 7.12 *Suppose that* $\equiv$ *is factorizable by* $\approx$ *and* $\varphi \models\!\models \ulcorner \psi \urcorner$ *for some* ψ*. Then* $\Phi \models \varphi$ *implies* $\varphi \in Th(\textsf{abstract } \langle \Sigma, \Phi \rangle \textsf{ w.r.t. } \equiv)$*.* □

This is a direct extension of the method for reasoning about abstractor specifications presented in Sect. 4 of [ST87], which applies only to formulae built in certain ways from observable equations. 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 7.12 can be obtained by adding "respectful" abstraction λ^{r} and quantification $\forall^{\mathrm{r}}$ to the syntax, where:

$\lambda^{\mathrm{r}}(x_1{:}\tau_1, \ldots, x_n{:}\tau_n).t$ abbreviates $\lambda(x_1{:}\tau_1, \ldots, x_n{:}\tau_n).$
$DOM_{\tau_1}(x_1) \wedge \cdots \wedge DOM_{\tau_n}(x_n) \wedge t$

$\forall^{\mathrm{r}} x{:}\tau.t$ abbreviates $\forall x{:}\tau.DOM_{\tau}(x) \Rightarrow t$

Definition 7.13 *A* respectful formula *is a formula that may contain* λ^{r} *and/or* $\forall^{\mathrm{r}}$ *but does not contain* λ *or* $\forall$*.*

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

Proof Method 7.14 *Suppose that* $\equiv$ *is factorizable by* $\approx$ *and* φ *is a 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, $\forall^{\mathrm{r}}$ on base types corresponds exactly to reachable quantification as in [Sch92]. Also, since every observable formula is equivalent to a respectful formulae, we have:

Proof Method 7.15 *Suppose that* φ *is an 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 example is easily dealt with using Proof Method 7.14: 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.

8 Further work

One reason for studying behavioural semantics in higher-order logic was the desire to apply the results in the Extended ML framework for the formal development of ML programs [KST94]. The results are of direct relevance in this context: the interpretation of interfaces involves abstractor specifications, and axioms are written in (a form of) higher-order logic. However, the framework here needs to be extended in two ways to make the match perfect.

First, the framework needs to be generalized to allow functions of higher type, in addition to the predicates of higher type that are already present. This would involve adding constants of higher type to signatures and allowing λ-abstraction to be used for forming functions. This can be done, as we will describe in a future paper; it is not straightforward because $[\![\tau]\!]_A^{\approx} \not\subseteq [\![\tau]\!]_A$ if we extend Definition 3.12 with obvious choices for function types. But note that n-ary functions may already be coded as $(n+1)$-ary predicates in the usual way.

Second, the use of behaviour and abstract in the context of structured specifications needs to be studied. An attempt appears in [BHW94], where they define $Mod(\textsf{behaviour } SP \textsf{ w.r.t. } \approx) = Beh_{\approx}(Mod(SP))$. Unless SP is a flat specification, the result is different from what is obtained when the specification-building operations in SP are interpreted as usual 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. But in view of the complexity of the predicates $INDIST_b$ (see [HS95]), it seems clear that proof methods involving the manipulation of relativized formulae will not be convenient for use in practice when $\approx$ is $\approx_{OBS}$. 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 $INDIST_b$ (e.g. Proof Methods 7.10 and 7.15) do not suffer from this problem.

Acknowledgements: Thanks to Michel Bidoit and Rolf Hennicker for many very useful comments. Proof Method 7.8 is due to them, and they pointed out that a previous version of Proof Methods 7.12 and 7.14 were unnecessarily restrictive. Thanks to Andrzej Tarlecki for many discussions on related topics and for drawing our attention to the idea behind $INDIST_b$ in Theorem 5.2. Thanks to David Aspinall and Wolfgang Degen for helpful suggestions.

References

[BH94] 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).

[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).

[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).

[HS95] M. Hofmann and D. Sannella. On behavioural abstraction and behavioural satisfaction in higher-order logic. Report ECS-LFCS-95-318, Univ. of Edinburgh (1995). Available on WWW in `http://www.dcs.ed.ac.uk/lfcsreps/EXPORT/95/ECS-LFCS-95-318`.

[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).

[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).

[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).

[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).

[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).