Referential Opacity in Equational Reasoning

Jo Erskine Hannay

Technical Report ECS-LFCS-98-398
Division of Informatics
University of Edinburgh

October 27, 1998

Abstract: Module constructs in programming languages have protection mechanisms hindering unauthorised external access to internal operators of data types. In some cases, granting external access to internal operators would result in serious violation of a data type's specified external properties. In order to reason consistently about specifications of such data types, it is necessary in general to incorporate a notion of protective abstraction barrier in proof strategies as well. We show how this can be done in the equational calculus by simply restricting the congruence axiom, and see how the motivation for this naturally arises from the forget-identify (FI) and forget-restrict-identify (FRI) approaches to specification refinement.

Contents

1 Introduction

Many programming languages have encapsulation mechanisms that hide internal detail of data types. Besides providing abstraction from uninteresting detail, these encapsulation mechanisms also provide vital protection of a data type's internal workings, to which direct access might otherwise enable a user to create havoc. Consider for example a data type implementation of sets in SML by sorted non-repeating lists. If granted access to the set constructor, a user might generate things (s)he thinks represent sets but which do not according to the data type. Then applying operators which assume the correct representation might give wrong answers. The power to enforce a suitable *abstraction barrier* between a module and the surrounding program is thus not just an organisational nicety, but also essential for program soundness. We here address these latter aspects of encapsulation, i.e. those pertaining to its *logical* or *protective*, as opposed to *organisational*, necessity.

Algebraic specification is viewed in this report in a refinement setting as described in e.g. [ST88, ST97] or [Mor94]. In such a setting data types are viewed as algebras, and in several schemes, e.g. [San91], [DO91] specifications and programs are written in a uniform language, so that specifications are abstract multi-modeled descriptions of a data type, while program modules are concrete monomorphic executable descriptions of the same. A refinement process then seeks to develop in a sound methodical way the latter from the former. In this setting, the need for abstraction barriers arises naturally in algebraic specifications as well. The specificational and semantic formalisms of algebraic specification have structural constructs, which if combined in the right order provide protective encapsulation, as for example in the *forget-identify* (FI) and the *forget-restrict-identify* (FRI) approach to refinement [Wir90].

The broad issue of our discussion is that when reasoning about specifications and programs, e.g. when doing refinement proofs, one needs to take into consideration *abstraction barriers in proof methods* as well. This is because information about hidden parts of a data type may have to be used when reasoning about its external properties. In this report we look specifically at proof obligations arising from the FI and FRI implementation schemes, i.e. implementing a data type by hiding details in, and then quotienting, another data type. Moreover, we wish to show how an abstraction barrier can easily be enforced in equational logic, so we look here at equational specifications. This means we will consider the case when the congruence with which the quotienting is done can be expressed by equations. It would then be proof-technically convenient if these latter equations could be used in an equational calculus directly in conjunction with other equations specifying the data type. Our result is that this is indeed possible, provided one incorporates the appropriate abstraction barrier *in the calculus itself*. It suffices to restrict the congruence (monotonicity) axiom to contexts without designated hidden symbols, i.e. imposing *referential opacity*, see [QG93, SS90] for other uses of referential opacity. Without such an abstraction barrier, the resulting set of equations may be inconsistent since (the axioms for) hidden operators might not respect the intended equality predicate.

Several proof system schemata for structured specifications exist, see [Hen97] for an overview, and the standard way by which quotienting is dealt with is by introducing a predicate symbol and explicitly axiomatising the congruence in terms of that symbol [Wir94]. This also goes for the behavioural equalities viz. congruences dealt with in [Hen97], where the axiomatisations are in general infinitary, although in [HS96] this problem is taken to higher order logic and finitary axiomatisation is then possible. Our approach is beneficial to mechanised reasoning because it remains finitary, first-order and purely equational. In some cases it also allows one to do behavioural verification more directly because now we can safely do proofs w.r.t. behavioural quotients instead of having to axiomatise behavioural equalities.

We will assume that the specifications to which the hiding and quotienting operators are applied are basic or "flat". It should be noted that this is not such a great restriction. Any first-order specification built from a basic specification by applying the standard specification building operators **sum**, **derive**, **translate** [Wir94] can be algorithmically normalised to a basic specification with a **derive** operator outermost [Wir93, FC92, Cen94]. The other relevant operators are **abstract**, **behaviour** and **quotient**. In a refinement context the two former, it can be argued, should be seen as meta-operators and should only be applied outermost [BST]. A similar argument can be made for **quotient**.

In Sect. 2 relevant notions are given as well as motivating examples. In Sect. 3 a calculus is presented which is sound and complete w.r.t. the model class of an equational instance of an FI structure. In Sect. 4 we present a calculus with an ω-rule which is shown sound and complete for the semantics of an equational instance of the FRI approach. The FI case is a special case of the FRI case and the completeness proof of the latter immediately gives a completeness proof for the former.

2 Preliminaries and Motivation

A basic knowledge of notions within universal algebra and algebraic specification is assumed, see [Wir90, ST88]. Below we give some notions and simplifying assumptions central to the discussion. We will be dealing with many-sorted algebraic specifications whose semantics will be given as classes of total many-sorted algebras with non-empty carriers. Fix a signature $\Sigma = \langle S, \Omega \rangle$. The class of Σ-algebras is denoted by $\Sigma\mathbf{Alg}$. For Σ-algebras A and B the class of Σ-homomorphisms from A to B is denoted by $\Sigma\mathbf{Alg}(A,\ B)$. We also write $\phi : A \to B$ to indicate that ϕ is a homomorphism from A to B. Throughout this report, fix X as a U-sorted set of variables, where U includes all sorts involved. The Σ-term algebra, i.e. the free Σ-algebra generated by X^S where the S-sorted X^S is given by $X^S_s = X_s$ for $s \in S$, is denoted by $T_\Sigma(X)$. For a Σ-context $c[\Box]$ we write $c \in T_\Sigma(X)$ instead of $c \in T_{\Sigma\cup\{\Box\}}(X)$. All signatures $\Sigma = \langle S, \Omega \rangle$ are assumed to be *sensible* w.r.t. to a designated set $I \subseteq S$, i.e. for the I-sorted X^I given by $X^I_s = X_s$ for $s \in I$, we assume that the free Σ-algebra generated by X^I, denoted $T_\Sigma(X^I)$, is non-empty. If $I = \emptyset$ this amounts to assuming that there is at least one constant in Ω of every sort s in S. We write $T_\Sigma(\emptyset)$ as G_Σ.

For signatures $\Sigma = \langle S, \Omega \rangle, \Sigma' = \langle S', \Omega' \rangle$, a *signature morphism* $\sigma : \Sigma \to \Sigma'$ maps the sorts and operator symbols of Σ to those of Σ' such that sorts are preserved. For a Σ'-algebra A the *σ-reduct* $A|_\sigma$ of A is the Σ-algebra with carriers $(A|_\sigma)_s = A_{\sigma(s)}$ for each sort $s \in S$ and $f^{A|_\sigma} = \sigma(f)^A$ for each $f \in \Omega$. For any Σ'-congruence $\sim^A$ on A, $\sim^A|_\sigma$ is defined as $\sim^A|_{\sigma_s} = \sim^A{}_{\sigma(s)}$ for each sort $s \in S$. For any Σ'-homomorphism $\phi : A \to B$, $\phi|_\sigma : A|_\sigma \to B|_\sigma$ is the Σ-homomorphism defined by $\phi|_{\sigma_s} = \phi_{\sigma(s)}$ for each sort $s \in S$. In case $\Sigma \subseteq \Sigma'$ and $\sigma : \Sigma \hookrightarrow \Sigma'$ is the inclusion morphism, we write $A|_\Sigma$, $\sim^A|_\Sigma$, $\phi^A|_\Sigma$, and we might write $a \sim^A a'$ in place of $a \sim^A|_\Sigma a'$, since $\sim^A|_{\Sigma_s} = \sim^A{}_s$ for $s \in S$.

If σ is not surjective, the effect is that of *hiding*, i.e. removing, those carriers and operators of A which are not interpretations of symbols in $\sigma(\Sigma)$.

The class of all Σ-algebras that are models (in the standard sense) of a set of axioms Φ is denoted $Mod_\Sigma(\Phi)$. In examples, equational propositions will be written without universal quantification, but they should be regarded as implicitly quantified over free variables.

2.1 Congruence Induced by a Set of Equations

The following standard notion is central. For a set of Σ-equations $E \subseteq T_\Sigma(X) \times T_\Sigma(X)$, the *congruence $\sim_E^A$ induced by E* on any Σ-algebra A is defined as the least Σ-congruence containing $\{\langle \phi(l), \phi(r)\rangle \mid \langle l, r\rangle \in E,\ \phi : T_\Sigma(X) \to A\}$. This definition is equivalent to demanding the least *equivalence relation* containing $\{\langle \phi(c[l]), \phi(c[r])\rangle \mid \langle l, r\rangle \in E, c \in T_\Sigma(X),\ \phi : T_\Sigma(X) \to A\}$, i.e. the relation inductively defined by

$$induce : \frac{}{\phi(c[l]) \sim_E^A \phi(c[r])};\quad \langle l, r\rangle \in E, c \in T_\Sigma(X),\ \phi : T_\Sigma(X) \to A$$

$$refl : \frac{}{a \sim_E^A a} \qquad sym : \frac{a \sim_E^A a'}{a' \sim_E^A a} \qquad trans : \frac{a \sim_E^A a'', a'' \sim_E^A a'}{a \sim_E^A a'}$$

The quotient w.r.t. to $\sim_E^A$ is written A/E. Of course, usually $s \sim_E^{T_\Sigma(X)} t$ is written $E \vdash s = t$.

2.2 Abstraction Barriers by Specificational Structure - FI & FRI

Henceforth, we tacitly assume that every class of algebras presented is closed under isomorphism. A *basic specification* is a pair $\langle \Sigma, \Phi\rangle$. Its *loose* semantics $[\![\langle \Sigma, \Phi\rangle]\!]$ is $Mod_\Sigma(\Phi)$. A number of *specification building operators* exist for constructing *structured* specifications. A common one which will be used in examples is **enrich** SP **by sorts** S' **ops** Ω' **axioms** Φ' with semantics $\{A \in \Sigma'\mathbf{Alg} \mid A|_\Sigma \in [\![SP]\!] \ \wedge\ A \models \Phi'\}$, where $\Sigma' = \langle S \cup S', \Omega \cup \Omega'\rangle$. (This **enrich** operator can be expressed by the **sum** operator.) We are here interested in encapsulation and in particular encapsulation out of logical necessity. Our focus is therefore on the two operators **derive** SP **by** σ whose semantics for a signature morphism σ, is $\{A|_\sigma \mid A \in [\![SP]\!]\}$, and **quotient** SP **by** E whose semantics for a set of equations E, is $\{A/E \mid A \in [\![SP]\!]\}$. The particular structure of interest is

$$\textbf{quotient} \quad (\textbf{derive}\ \langle \Sigma, E\rangle\ \textbf{by}\ incl : \Sigma^e \hookrightarrow \Sigma) \quad \textbf{by}\ \ E' \tag{1}$$

with semantics $\{(A|_{\Sigma^e})/E' \mid A \in Mod_\Sigma(E)\}$. If $\Sigma^e \subsetneq \Sigma$ then the signature fragment $\Sigma^h = \Sigma \setminus \Sigma^e$ is outside the image of *incl*, so the reduct construct hides the interpretations of operator symbols and sorts in Σ^h. Structure is the essential abstraction barrier here: It is crucial that the hiding **derive** step is done before quotienting, since quotienting in the presence of hidden operators might give inconsistency in the sense illustrated in the following example.

Example 1 Following [ST88], a specification SP' is a *refinement* of SP, written $SP \rightsquigarrow SP'$ iff $[\![SP']\!] \subseteq [\![SP]\!]$. A nice feature in refinement settings is the provision for using an implementation of one data type to implement another. In the example below from [ST88], the specification Set is refined by using Bag and specification building operators. This reuses any refinement Bag $\overset{*}{\rightsquigarrow} SP''$ previously done for Bag. In particular if Bag has been refined to an executable module, then this code is reused when implementing Set. Specifically we have

spec Set **is**
sorts nat, set
ops empty : set, add : nat $\times$ set $\to$ set
in : nat $\times$ set $\to$ bool
axioms $\mathsf{add}(x, \mathsf{add}(x, s)) = \mathsf{add}(x, s)$
$\mathsf{add}(x, \mathsf{add}(y, s)) = \mathsf{add}(y, \mathsf{add}(x, s))$
$\mathsf{in}(x, \mathsf{empty}) = \mathsf{false}$
$\mathsf{in}(x, \mathsf{add}(y, s)) = \mathsf{if}\ x =_{\mathsf{nat}} y\ \mathsf{then\ true\ else\ in}(x, s)$

spec Bag **is**
sorts nat, bag
ops empty : bag, add : nat $\times$ bag $\to$ bag
count : nat $\times$ bag $\to$ nat
axioms $\mathsf{add}(x, \mathsf{add}(y, b)) = \mathsf{add}(y, \mathsf{add}(x, b))$
$\mathsf{count}(x, \mathsf{empty}) = 0$
$\mathsf{count}(x, \mathsf{add}(y, b)) = \mathsf{if}\ x =_{\mathsf{nat}} y\ \mathsf{then\ succ}(\mathsf{count}(x, b))\ \mathsf{else\ count}(x, b)$

The idea is to put an appropriate interface on bags as specified by Bag, so that they look like sets as specified by Set. This may be done safely by adding in as an interface operator, then hiding its implementation in terms of count and then identifying bags that represent the same set. First in is added:

spec Bag+ **is**
enrich Bag **by**
ops in : nat $\times$ bag $\to$ bool
axioms $\mathsf{in}(x, b) = \mathsf{count}(x, b) > 0$

Then we use an instance of the structure (1):

spec SetbyBag **is**
quotient
derive Bag+ **by** $\sigma = \iota[\mathsf{set} \mapsto \mathsf{bag}]$
by $E' : \{\mathsf{add}(x, \mathsf{add}(x, s)) = \mathsf{add}(x, s)\}$

where σ is the signature morphism from the signature of Set to that of Bag+ which is the identity on everything except the sort set which is renamed to bag. (For simplicity (1) was stated using an inclusion morphism. In this example, the signature morphism is not an inclusion proper. However, the renaming from set to bag is trivial.) The morphism is not surjective thus hiding count. This specification is structured so that count is hidden *before* quotienting. If this were not done, the specification would be inconsistent relative to the intended semantics on nat, since any model $B = A/E'$ would then have to satisfy e.g. $2 = 1$, by

$$2 = \mathsf{count}^B(x, \mathsf{add}^B(x, \mathsf{add}^B(x, \mathsf{empty}^B))) = \mathsf{count}^B(x, \mathsf{add}^B(x, \mathsf{empty}^B)) = 1$$

and now it would be too late for hiding count^B. However, the above structure ensures the appropriate abstraction barrier and the desired semantics.

In an executable implementation of SetbyBag, the derive operator might be implemented by an encapsulation mechanism hindering outside access to count, and the quotient operator might be implemented by an equality predicate.

The task is now to prove Set $\rightsquigarrow$ SetbyBag. We will present a calculus allowing a direct approach to proofs w.r.t. the general structure (1), and hence particularly w.r.t. SetbyBag for this example. ○

The specification structure (1) in general, and the specification SetbyBag of Example 1 in particular, are instances of the common *forget-identify* (FI) implementation strategy of algebraic specification. Even more common is the strategy of *forget-restrict-identify* (FRI), which involves restricting to the unique reachable sub-algebra after reducting and before quotienting.

Let $\Sigma = \langle S, \Omega\rangle$ and let $S' \subseteq S$. The set $I = S \setminus S'$ might be thought of as designated *input* sorts. A Σ-algebra A is *reachable on* S' if there is no proper Σ-subalgebra whose I-sorted carriers are the same as those of A. Equivalently, let $X^I \subseteq X$ denote the I-sorted variables of X. Then A is reachable on S' iff for every $a \in A$ there is a term $t \in T_\Sigma(X^I)$ such that $\phi(t) = a$ for some homomorphism $\phi : T_\Sigma(X^I) \to A$. Any Σ-algebra has a unique Σ-subalgebra which is reachable on S', denoted $R_{S'}(A)$. The *restriction* $R\restriction^{B'}_{A'}$ of a relation $R \subseteq A \times B$ is here taken to be $R \cap A' \times B'$. For any Σ-homomorphism $\phi : A \to B$, the Σ-homomorphism $R_{S'}(\phi) : R_{S'}(A) \to R_{S'}(B)$ (*qua* relation) is defined to be $\phi\restriction^{B}_{R_{S'}(A)}$ (note that the range of $\phi\restriction^{B}_{R_{S'}(A)}$ is $R_{S'}(B)$, so functoriality is secured). For a Σ-congruence $\sim$ on A, the Σ-congruence $R_{S'}(\sim)$ is defined as $\sim\restriction^{R_{S'}(A)}_{R_{S'}(A)}$.

The semantics of the specification **restrict** SP **on** S' is $\{R_{S'}(A) \mid A \in [\![SP]\!]\}$. Specifications with the **restrict** operator are normalisable to the form mentioned in the introduction, but with infinitary axioms. However, in refinement we could again claim **restrict** as a meta-operator to be applied outermost. The FRI approach then, is in our context represented by the specification structure

$$\textbf{quotient} \quad (\textbf{restrict} \quad (\textbf{derive}\ \langle \Sigma, E\rangle\ \textbf{by}\ \mathit{incl} : \Sigma^e \hookrightarrow \Sigma) \quad \textbf{on}\ S') \ \ \textbf{by} \ \ E' \qquad (2)$$

where $\Sigma^e = \langle S^e, \Omega^e\rangle$, $S' \subseteq S^e$. Its semantics is $\{R_{S'}(A|_{\Sigma^e})/E' \mid A \in \mathit{Mod}_\Sigma(E)\}$. Note that the input sorts I are now $S^e \setminus S'$. There is a range of model classes according to the choice of S'. The case $S' = \emptyset$ gives $R_\emptyset(A|_{\Sigma^e}) = A|_{\Sigma^e}$, and corresponds to FI. The case $S' = S^e$ is ground term denotability.

Example 2 Consider the specification

> **spec** SetEnr **is**
> **enrich** Set **by**
> **ops** remove : nat $\times$ set $\to$ set
> **axioms** $\mathsf{in}(x, \mathsf{remove}(x, s)) = \mathsf{false}$

In this example sets as specified by SetEnr are implemented by lists where equal elements occur consecutively. (This means that when inserting an element one has to check for occurrences of that element in the list, and the more obvious way to implement sets, would be by lists where elements occur at most once. However, it is not unthinkable that one at lower levels of implementation would wish to keep a record of how many times the insertion operation on any particular element has been called. Also, formulating the example in this way will nicely illustrate the use of referential opacity.) We do this by putting an appropriate interface on basic lists, i.e. starting from

> **spec** List **is**
> **sorts** nat, list
> **ops** nil : list, _ :: _ : nat $\times$ list $\to$ list

we add interface operators:

> **spec** List+ **is**
> **enrich** List **by**
> **ops** empty : list, add : nat $\times$ list $\to$ list,
> remove : nat $\times$ list $\to$ list, in : nat $\times$ list $\to$ bool
> **axioms** $\mathsf{empty} = \mathsf{nil}$
> $\mathsf{add}(x, \mathsf{nil}) = x :: \mathsf{nil}$
> $\mathsf{add}(x, y :: l) = \mathsf{if}\ x =_{\mathsf{nat}} y\ \mathsf{then}\ x :: y :: l\ \mathsf{else}\ y :: \mathsf{add}(x, l)$
> $\mathsf{in}(x, \mathsf{nil}) = \mathsf{false}$
> $\mathsf{in}(x, y :: l) = \mathsf{if}\ x =_{\mathsf{nat}} y\ \mathsf{then\ true\ else}\ \mathsf{in}(x, l)$
> $\mathsf{remove}(x, \mathsf{nil}) = \mathsf{nil}$
> $\mathsf{remove}(x, y :: \mathsf{nil}) = \mathsf{if}\ x =_{\mathsf{nat}} y\ \mathsf{then\ nil\ else}\ y :: \mathsf{nil}$
> $\mathsf{remove}(x, y :: z :: l) = \mathsf{if}\ x =_{\mathsf{nat}} y\ \mathsf{then}$
> $\qquad \mathsf{if}\ x =_{\mathsf{nat}} z\ \mathsf{then}\ \mathsf{remove}(x, l)\ \mathsf{else}\ z :: l$
> $\quad \mathsf{else}\ y :: \mathsf{remove}(x, z :: l)$

Notice that remove is optimised by using the fact that we intend to represent sets by lists in which equal elements are stored consecutively. However, this representation has to be guaranteed by imposing a suitable abstraction barrier. We use the FRI construct (2):

$$
\begin{array}{l}
\textbf{spec } \mathsf{SetbyCanonicalList} \textbf{ is} \\
\quad \textbf{quotient} \\
\quad\quad \textbf{restrict} \\
\quad\quad\quad \textbf{derive } \mathsf{List+} \textbf{ by } \sigma = \iota[\mathsf{set} \mapsto \mathsf{list}] \\
\quad\quad \textbf{on } \{\mathsf{set}\} \\
\quad \textbf{by } E' : \ \{\mathsf{add}(x, \mathsf{add}(y, s)) = \mathsf{add}(y, \mathsf{add}(x, s)), \\
\quad\quad\quad\quad\ \ \mathsf{add}(x, \mathsf{add}(x, s)) = \mathsf{add}(x, s)\}
\end{array}
$$

where σ is the signature morphism from the signature of Set to that of List+ which is the identity on everything except for the renaming of set to list. It is not surjective thus hiding nil and ::. Semantically, no model of SetbyCanonicalList has interpretations of nil and ::, so the only way of generating lists is by the interpretations of the operator symbols empty and add which in the initial model will generate the canonical lists with which we intend to represent sets. However, we also have to restrict to the least reachable sub-algebra on $\{\mathsf{set}\}$, because the reduct operator only takes away operators and entire carriers, and leaves all carriers which are interpretations of sorts in $\sigma(\Sigma)$ intact. Without the restrict step, models would not necessarily satisfy $\mathsf{in}(x, \mathsf{remove}(x, s)) = \mathsf{false}$, since s would then range over all lists hence also non-canonical lists. Note that we must prove $[\![\mathsf{SetbyCanonicalList}]\!] \models \mathsf{in}(x, \mathsf{remove}(x, s)) = \mathsf{false}$ to verify that SetbyCanonicalList is a refinement of Set. ○

2.3 Overview of Main Results

We will present sound and complete equational calculi for the FI and FRI structured semantics as formulated in schemes (1) and (2). The usefulness of such calculi are apparent in refinement scenarios as those in Examples 1 and 2. The calculi will be generalisations in a certain sense of calculi for the flat basic cases, as explained in the following. For a basic specification $SP = \langle \Sigma, E \rangle$ we have by Birkhoff for the equational calculus $\vdash$

$$[\![SP]\!] \models s = t \ \Leftrightarrow \ T_\Sigma(X)/E \models s = t \ \Leftrightarrow \ E \vdash s = t \qquad *$$

Here $T_\Sigma(X)/E$ is a *classifying* model of $[\![SP]\!]$. Now let K^{FI} be the semantics of the FI structure (1). The first main result will be a calculus $\vdash^{\mathrm{FI}}$ and a classifying model $T_{K^{\mathrm{FI}}}$ such that

$$K^{\mathrm{FI}} \models s = t \ \Leftrightarrow \ T_{K^{\mathrm{FI}}} \models s = t \ \Leftrightarrow \ \vdash^{\mathrm{FI}} s = t$$

Secondly, recall that for the basic specification SP we have for $\vdash^\omega$, i.e. the equational calculus augmented with the ω-rule,

$$\mathit{Reach}([\![SP]\!]) \models s = t \ \Leftrightarrow \ G_\Sigma/E \models s = t \ \Leftrightarrow \ \vdash^\omega s = t \qquad **$$

where $Reach([\![SP]\!])$ is the subclass of $[\![SP]\!]$ consisting of all algebras reachable on the sorts S of Σ, i.e. ground term denotable algebras, or computation structures. Now, in the FRI approach we are interested in classes of reachable sub-algebras, rather than sub-classes of reachable algebras. However in a flat equational setting these two are the same: Let $R_{S'}(Mod_\Sigma(E)) = \{R_{S'}(A) \mid A \in Mod_\Sigma(E)\}$ and $Reach_{S'}(Mod_\Sigma(E)) = \{A \in Mod_\Sigma(E) \mid A \text{ is reachable on } S'\}$.

Fact 2.1 $R_{S'}(Mod_\Sigma(E)) = Reach_{S'}(Mod_\Sigma(E))$

Proof: Consider any $A \in Mod_\Sigma(E)$ that is reachable on S'. But then $R_{S'}(A) = A$, so there is an $A' \in Mod_\Sigma(E)$ s.t. $R_{S'}(A') = A$. Hence $A \in R_{S'}(Mod_\Sigma(E))$. Conversely, consider any $R_{S'}(A) \in R_{S'}(Mod_\Sigma(E))$. By definition, $R_{S'}(A)$ is reachable on S', so it remains to show that $R_{S'}(A)$ is a model of E. This works because we have an equational setting, and is part of the variety theorem: We know that A is a model of E, i.e. for any$\langle l, r\rangle \in E$, $\forall \phi \in \Sigma\mathbf{Alg}(T_\Sigma(X),\ A)\ .\ \phi(l) = \phi(r)$. But this implies in particular $\forall \phi \in \Sigma\mathbf{Alg}(T_\Sigma(X),\ R_{S'}(A))\ .\ \phi(l) = \phi(r)$. Hence $R_{S'}(A)$ is a model of E. $\Box$

This correspondence means that we can utilise the ω-rule also when considering $R_{S'}([\![SP]\!])$. In fact it is follows that for arbitrary $S' \subseteq S$,

$$R_{S'}([\![SP]\!]) \models s = t \;\Leftrightarrow\; R_{S'}(T_\Sigma(X)/E) \models s = t \;\Leftrightarrow\; \vdash^{\omega}_{S'} s = t$$

where $\vdash^{\omega}_{S'}$ denotes the standard equational calculus augmented by the following parametrised ω-rule.

$$\frac{\forall \tau : T_\Sigma(X) \to R_{S'}(T_\Sigma(X))\ .\ \vdash \tau(s) = \tau(t)}{\vdash s = t}$$

This is not hard to show. In any case, this result also follows from our results later on. The special case $S' = \emptyset$ is simply $*$. The case $S' = S$ is $**$, in which case $R_{S'}(T_\Sigma(X)/E) \cong G_\Sigma/E$ is the initial object of $Mod_\Sigma(E)$.

Now let $K^{\mathrm{FRI}}_{S'}$ be the semantics the FRI structure (2). By analogy to the basic case we will as a second main result devise a calculus $\vdash^{\mathrm{FRI}}_{S'}$ with a parametrised ω rule and classifying model $T_{K^{\mathrm{FRI}}_{S'}}$ such that

$$K^{\mathrm{FRI}}_{S'} \models s = t \;\Leftrightarrow\; T_{K^{\mathrm{FRI}}_{S'}} \models s = t \;\Leftrightarrow\; \vdash^{\mathrm{FRI}}_{S'} s = t$$

Analogously to the basic case above, the classifying model $T_{K^{\mathrm{FRI}}_{S'}}$ will in the case $S' = S$ be the initial object of K^{FI}.

As a curio, there is an aspect in which the analogy does not hold. In the basic case $R_{S'}(T_\Sigma(X)/E)$ is free on $X_{S \setminus S'}$ in $[\![SP]\!]$, assuming that E does not identify any variables. However, in general $T_{K^{\mathrm{FRI}}_{S'}}$ is not free on $X_{S \setminus S'}$ in or for K^{FI}, except in the case $S' = S$.

3 FI Approach — A Referentially Opaque Calculus

In the following we shall develop a calculus for structured specifications of the form (1), for $\Sigma^e = \langle S^e, \Omega^e\rangle \subseteq \Sigma = \langle S, \Omega\rangle$, E a set of Σ-equations and E' a set of Σ^e-equations. The calculus implements a protective abstraction barrier in the form of referential opacity. The model class of (1) is given by $\{(A|_{\Sigma^e})/E' \mid A \in Mod_\Sigma(E)\}$ and will be denoted by K^{FI} throughout. We will give a calculus that is equationally sound and complete for K^{FI}.

Algebras in K^{FI} are of the form $(A|_{\Sigma^e})/E'$ where A is a model for E. The classifying model is $(T_\Sigma(X)/E|_{\Sigma^e})/E'$ (Theorem 3.1). Viewing for the moment $T_\Sigma(X)/E|_{\Sigma^e}$ as a "term-algebra" T, we directly get an "abstract" calculus for K^{FI} by considering $\sim^T_{E'}$ on T and the classifying model T/E'. This is a generalisation of the basic case $*$ in Sect. 2.3 where $E \vdash$ is given directly by $\sim_E^{T_\Sigma(X)}$. The abstract calculus thus operates on elements of T, i.e. congruence classes q of $T_\Sigma(X)/E|_{\Sigma^e}$. Notice that each q has the form $[t]_E$ for $t \in T_\Sigma(X)|_{\Sigma^e}$, and recall that in general $T_{\Sigma^e}(X) \not\supseteq T_\Sigma(X)|_{\Sigma^e}$, because for any $s \in S^e$, $T_\Sigma(X)|_{\Sigma^e}{}_s = T_\Sigma(X)_s$.

Of course, instead of this abstract calculus we would rather have a calculus operating on terms. We obtain this by "opening up" the congruence classes q and then building a calculus over E' on $T_\Sigma(X)|_{\Sigma^e}$. Opening up the congruence classes necessitates importing the calculus $E \vdash$.

Definition 1 (Calculus $\vdash^{\mathrm{FI}}$) *For all $u, v \in T_\Sigma(X)|_{\Sigma^e}$,*

$$importE : \frac{E \vdash u = v}{\vdash^{\mathrm{FI}} u = v}$$

$$induceE' : \frac{}{\vdash^{\mathrm{FI}} \phi(c[l]) = \phi(c[r])}; \langle l, r\rangle \in E', c \in T_{\Sigma^e}(X),\ \phi : T_{\Sigma^e}(X) \to T_\Sigma(X)|_{\Sigma^e}$$

$$refl : \frac{}{\vdash^{\mathrm{FI}} u = u} \qquad sym : \frac{\vdash^{\mathrm{FI}} u = v}{\vdash^{\mathrm{FI}} v = u} \qquad trans : \frac{\vdash^{\mathrm{FI}} u = w, \vdash^{\mathrm{FI}} w = v}{\vdash^{\mathrm{FI}} u = v}$$

In rule *induceE'* in Definition 1, the contexts are $T_{\Sigma^e}(X)$-contexts, giving referential opacity w.r.t. $T_\Sigma(X)|_{\Sigma^e}$. This is a direct consequence of the definition of congruence on a Σ^e-algebra ($T_\Sigma(X)|_{\Sigma^e}$) induced by a set of Σ^e-equations (E') (Sect. 2.1). In this way the abstraction barrier provided by the reduct construct in the semantics K^{FI}, gets its counterpart in the calculus in the form of referential opacity. Notice that in fact $\phi(c[\Box]) \in T_\Sigma(X)|_{\Sigma^e}$. However, the fact that $c[\Box]$ is a $T_{\Sigma^e}(X)$-context, ensures the essential property that all operator symbols in the path from $\Box$ to the root of $\phi(c[\Box])$ (seen as a tree of sub-terms) are from Ω^e.

Note that the calculus in Definition 1 is given by the Σ^e-congruence on $T_\Sigma(X)|_{\Sigma^e}$ induced by the set of Σ^e-equations

$$E^{\mathrm{FI}} = (\sim_E^{T_\Sigma(X)}) \restriction_{T_\Sigma(X)|_{\Sigma^e}} \ \cup \ E' \tag{3}$$

We shall make use of this observation later.

All algebras in K^{FI} are Σ^e-algebras, so satisfaction by K^{FI} only has meaning for Σ^e-equations. However, it is necessary for completeness that the calculus considers substitutions into $T_\Sigma(X)|_{\Sigma^e}$ of Σ^e-equations, since $(T_\Sigma(X)/E|_{\Sigma^e})/E'$ is in K^{FI}. This is just a manifestation of the above discussion, where it was motivated from considering what will turn out to be the classifying model $(T_\Sigma(X)/E|_{\Sigma^e})/E'$ that $\vdash^{\mathrm{FI}}$ must be defined over $T_\Sigma(X)|_{\Sigma^e}$.

Theorem 3.1 (Soundness and completeness) *Let* K^{FI} *be the semantics* $\{(A|_{\Sigma^e})/E' \mid A \in Mod_\Sigma(E)\}$ *of (1). For all* $u, v \in T_{\Sigma^e}(X)$*,*

$$K^{\mathrm{FI}} \models u = v \;\Leftrightarrow\; T_\Sigma(X)/E|_{\Sigma^e}/E' \models u = v \;\Leftrightarrow\; \vdash^{\mathrm{FI}} u = v$$

Proof: This follows from Theorem 4.3 by observations 4.1 and 4.2. □

Example 3 By Theorem 3.1, the calculus $\vdash^{\mathrm{FI}}$ can be used in verifying the refinement postulated in Example 1, namely $\mathsf{Set} \rightsquigarrow \mathsf{SetbyBag}$. The calculus ensures the safe interaction between the set E of equations associated with $\mathsf{Bag}+$ and the set E' of equations introduced in the quotienting step forming $\mathsf{SetbyBag}$. For instance, although we can derive $\vdash^{\mathrm{FI}} \mathsf{add}(x, \mathsf{add}(x, \mathsf{empty})) = \mathsf{add}(x, \mathsf{empty})$, referential opacity prevents us from inferring $\vdash^{\mathrm{FI}} \mathsf{count}(x, \mathsf{add}(x, \mathsf{add}(x, \mathsf{empty}))) = \mathsf{count}(x, \mathsf{add}(x, \mathsf{empty}))$, which would then have given $\vdash^{\mathrm{FI}} 2 = 1$. The inference is illegal because count is a hidden operator symbol. Referential opacity ensures soundness and is an appropriate abstraction barrier in the calculus. ○

4 FRI Approach

We now address the FRI approach in which reducts are restricted to reachable subalgebras on certain sorts. We consider the FRI specification structure (2). Again, for $\Sigma^e = \langle S^e, \Omega^e \rangle \subseteq \Sigma = \langle S, \Omega \rangle$, let E be a set of equations over $T_\Sigma(X)$ and let E' be a set of equations over $T_{\Sigma^e}(X)$. Let $S' \subseteq S^e$. The model class of (2) is given by $\{R_{S'}(A|_{\Sigma^e})/E' \mid A \in Mod_\Sigma(E)\}$ and will be denoted by $K^{\mathrm{FRI}}_{S'}$ throughout.

Observation 4.1 *For* $S' = \emptyset$*,* $K^{\mathrm{FRI}}_{S'} = K^{\mathrm{FI}}$.

4.1 A Restricted Calculus with ω-rule

Algebras in $K^{\mathrm{FRI}}_{S'}$ are of the form $R_{S'}(A|_{\Sigma^e})/E'$ where A is a model for E. The classifying model is $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E'$ (Theorem 4.3). As we did for the FI case, viewing for the moment $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})$ as a "term-algebra" T', we directly get an "abstract" calculus for K^{FRI} by considering $\sim^{T'}_{E'}$ on T' and the classifying model T'/E'. The abstract calculus thus operates on elements of T', i.e. congruence classes q of $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})$, and each q has the form $[t]_E$ for $t \in R_{S'}(T_\Sigma(X)|_{\Sigma^e})$. Again we obtain a term calculus by opening up the congruence classes and importing $E \vdash$. This calculus is defined over $R_{S'}(T_\Sigma(X)|_{\Sigma^e})$ and is given by the congruence on $R_{S'}(T_\Sigma(X)|_{\Sigma^e})$ induced by the set of Σ^e-equations

$$E^{\mathrm{FRI}} = \; (\sim^{T_\Sigma(X)}_{E}) \restriction_{R_{S'}(T_\Sigma(X)|_{\Sigma^e})} \;\cup\; E'$$

Remember now that as K^{FRI} consists of Σ^e-algebras, we are interested in satisfiability of Σ^e statements, i.e. Σ^e-equations. However, depending on S' it may be the case that $T_{\Sigma^e}(X) \not\subseteq R_{S'}(T_\Sigma(X)|_{\Sigma^e})$, in which case the calculus will not respond to all Σ^e-equations. Hence, we supply an ω-rule dependent on S'.

Definition 2 (Calculus $\vdash^{\mathrm{FRI}}_{S'}$) *The calculus $\vdash^{\mathrm{FRI}}_{S'}$ is given by the following single rule. For all $u, v \in T_{\Sigma^e}(X)$,*

$$\omega_{S'} : \frac{\forall \tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \ . \ \tau(u) \sim^{R_{S'}(T_\Sigma(X)|_{\Sigma^e})}_{E^{\mathrm{FRI}}} \tau(v)}{\vdash^{\mathrm{FRI}}_{S'} u = v}$$

To spell that out, let $\vdash^{\mathrm{FI}}_{S'}$ be the following calculus. For all $u, v \in R_{S'}(T_\Sigma(X)|_{\Sigma^e})$,

$$importE : \frac{E \vdash u = v}{\vdash^{\mathrm{FI}}_{S'} u = v}$$

$$induceE' : \frac{}{\vdash^{\mathrm{FI}}_{S'} \phi(c[l]) = \phi(c[r])};$$

$$\langle l, r\rangle \in E', c \in T_{\Sigma^e}(X), \ \phi : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e})$$

$$refl : \frac{}{\vdash^{\mathrm{FI}}_{S'} u = u} \qquad sym : \frac{\vdash^{\mathrm{FI}}_{S'} u = v}{\vdash^{\mathrm{FI}}_{S'} v = u} \qquad trans : \frac{\vdash^{\mathrm{FI}}_{S'} u = w, \vdash^{\mathrm{FI}}_{S'} w = v}{\vdash^{\mathrm{FI}}_{S'} u = v}$$

Now $\vdash^{\mathrm{FRI}}_{S'}$ is given by the following rule. For all $u, v \in T_{\Sigma^e}(X)$,

$$\omega_{S'} : \frac{\forall \tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \ . \ \vdash^{\mathrm{FI}}_{S'} \tau(u) = \tau(v)}{\vdash^{\mathrm{FRI}}_{S'} u = v}$$

Observation 4.2 *$\vdash^{\mathrm{FRI}}_{S'}$ subsumes $\vdash^{\mathrm{FI}}$: If $S' = \emptyset$ then the rule $\omega_{S'}$ adds nothing to $\vdash^{\mathrm{FI}}_{S'}$, so for $u, v \in T_{\Sigma^e}(X)$, $\vdash^{\mathrm{FRI}}_{\emptyset} u = v \Leftrightarrow \vdash^{\mathrm{FI}}_{\emptyset} u = v \Leftrightarrow \vdash^{\mathrm{FI}} u = v$.*

Theorem 4.3 (Soundness and completeness) *Let $K^{\mathrm{FRI}}_{S'}$ be the semantics $\{R_{S'}(A|_{\Sigma^e})/E' \mid A \in Mod_\Sigma(E)\}$ of (2). For any $u, v \in T_{\Sigma^e}(X)$,*

$$K^{\mathrm{FRI}}_{S'} \models u = v \ \Leftrightarrow \ R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E' \models u = v \ \Leftrightarrow \ \vdash^{\mathrm{FRI}}_{S'} u = v$$

Proof: The proof is split into Lemmas 4.4 and 4.7 below. □

Lemma 4.4 (Completeness)

$$K^{\mathrm{FRI}}_{S'} \models u = v \ \Rightarrow \ R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E' \models u = v \ \Rightarrow \ \vdash^{\mathrm{FRI}}_{S'} u = v$$

Proof: Suppose $K^{\mathrm{FRI}}_{S'} \models u = v$. By definition $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E' \models u = v$. Lemma 4.5 gives $\forall \tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \ . \ [[\tau(u)]_E]_{E'} = [[\tau(v)]_E]_{E'}$. Lemma 4.6 then gives $\forall \tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \ . \ \vdash^{\mathrm{FI}}_{S'} \tau(u) = \tau(v)$. Finally, the rule $\omega_{S'}$ gives $\vdash^{\mathrm{FRI}}_{S'} u = v$. □

Lemma 4.5 *For $u, v \in T_{\Sigma^e}(X)$,*

$$R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E' \models u = v$$
$$\Downarrow$$
$$\forall \tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \ . \ [[\tau(u)]_E]_{E'} = [[\tau(v)]_E]_{E'}$$

where $[w]_E$ denotes the congruence class of w in $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})$, and $[q]_{E'}$ denotes the congruence class of q in $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E'$.

Proof: Suppose $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E' \models u = v$, i.e.

$$\forall \varphi : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E' \ . \ \varphi(u) = \varphi(v)$$

Define $\psi_E : T_\Sigma(X) \to T_\Sigma(X)/E$ as $\psi_E(u) = [u]_E$. For any $\tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e})$, we get $R_{S'}(\psi_E|_{\Sigma^e}) \circ \tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})$.

Let $\psi_{E'} : R_{S'}(T_\Sigma(X)/E|_{\Sigma^e}) \to R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E'$ be defined as $\psi_{E'}(q) = [q]_{E'}$. Then $\psi_{E'} \circ (R_{S'}(\psi_E|_{\Sigma^e}) \circ \tau) : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E'$, and so

$$[[\tau(u)]_E]_{E'} = \psi_{E'} \circ (R_{S'}(\psi_E|_{\Sigma^e}) \circ \tau)(u) = \psi_{E'} \circ (R_{S'}(\psi_E|_{\Sigma^e}) \circ \tau)(v) = [[\tau(v)]_E]_{E'}.$$

□

Lemma 4.6 *Let $[w]_E$ denote the congruence class of w in $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})$, and $[q]_{E'}$ denote the congruence class of q in $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E'$. For $u, v \in R_{S'}(T_\Sigma(X)|_{\Sigma^e})$,*

$$[[u]_E]_{E'} = [[v]_E]_{E'} \ \Rightarrow \ \vdash^{\mathrm{FI}}_{S'} u = v$$

Proof: Suppose $[[u]_E]_{E'} = [[v]_E]_{E'}$, that is, $[u]_E \sim_{E'}^{R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})} [v]_E$. Induction on the construction of $\sim_{E'}^{R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})}$.

induce : For some $\langle l, r\rangle \in E'$, $c \in T_{\Sigma^e}(X)$, $\varphi : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})$, we have $\varphi(c[l]) \sim_{E'}^{R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})} \varphi(c[r])$ and $u \in \varphi(c[l])$ and $v \in \varphi(c[r])$. It is a fact that $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e}) \cong R_{S'}(T_\Sigma(X)|_{\Sigma^e})/R_{S'}(\sim_E^{T_\Sigma(X)}|_{\Sigma^e})$. So by Fact 4.9 φ can be factored into $\psi_E \circ \tau$ for some $\tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e})$, and we have $\varphi(c[l]) = [\tau(c[l])]_E$ and $\varphi(c[r]) = [\tau(c[r])]_E$. So then $E \vdash u = \tau(c[l])$ and $E \vdash \tau(c[r]) = v$, which by *importE* gives $\vdash^{\mathrm{FI}}_{S'} u = \tau(c[l])$ and $\vdash^{\mathrm{FI}}_{S'} \tau(c[r]) = v$. By *induceE'* we have $\vdash^{\mathrm{FI}}_{S'} \tau(c[l]) = \tau(c[r])$, and *trans* then gives $\vdash^{\mathrm{FI}}_{S'} u = v$.

refl : Then $[u]_E = [v]_E$. So $E \vdash u = v$, which by *importE* gives $\vdash^{\mathrm{FI}}_{S'} u = v$.

sym and *trans*: These are dealt with by the i.h. and *sym* and *trans* of $\vdash^{\mathrm{FI}}_{S'}$. □

Lemma 4.7 (Soundness) $\vdash^{\mathrm{FRI}}_{S'} u = v \ \Rightarrow \ K^{\mathrm{FRI}}_{S'} \models u = v$

Proof: Fix $A \in Mod_\Sigma(E)$ arbitrarily. Suppose $\vdash^{\mathrm{FRI}}_{S'} u = v$, for $u, v \in T_{\Sigma^e}(X)$. The way this is possible is via the only rule $\omega_{S'}$, and so we must have $\forall \tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \ . \ \vdash^{\mathrm{FI}}_{S'} \tau(u) = \tau(v)$. By Lemma 4.8 we then get for any $\tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e})$ and $\psi : R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \to R_{S'}(A|_{\Sigma^e})$ that

$$\psi(\tau(u)) \sim_{E'}^{R_{S'}(A|_{\Sigma^e})} \psi(\tau(v)). \qquad *$$

Fix $\phi' : T_{\Sigma^e}(X) \to R_{S'}(A|_{\Sigma^e})$ arbitrarily. We now show that there exist $\tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e})$ and $\psi : R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \to R_{S'}(A|_{\Sigma^e})$ such that $\phi' = \psi \circ \tau$, i.e. making the left-hand part of following diagram commute:

$$\begin{tikzcd}
T_{\Sigma^e}(X) \arrow[r, "\exists\tau"] \arrow[dr, "\forall\phi'"'] & R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \arrow[d, "\exists\psi"] & \\
& R_{S'}(A|_{\Sigma^e}) & T_{\Sigma^e}(X^I) \arrow[l, dashed, "\rho"'] \arrow[ul, dashed, hook]
\end{tikzcd}$$

For any $x \in X_s$, $s \in S^e$, let $a = \phi'(x)$. There is some $t_a \in T_{\Sigma^e}(X^I)$ and $\rho : T_{\Sigma^e}(X^I) \to A|_{\Sigma^e}$ such that $\rho(t_a) = a$. We determine τ by defining $\tau(x) = t_a$ ($T_{\Sigma^e}(X^I) \subseteq R_{S'}(T_\Sigma(X)|_{\Sigma^e})$ because $X^I \subseteq R_{S'}(T_\Sigma(X)|_{\Sigma^e})$ and $R_{S'}(T_\Sigma(X)|_{\Sigma^e})$ is a Σ^e-algebra).

Determine ψ as follows. Let $\psi_\rho : T_\Sigma(X) \to A$ be determined by

$$\psi_\rho(x) = \begin{cases} \rho(x), & x \in X_s,\ s \in I = S^e \setminus S' \\ a_\perp, & x \in X_s,\ s \in S \setminus I, \text{ for some choice } a_\perp \end{cases}$$

Define $\psi = R_{S'}(\psi_\rho|_{\Sigma^e})$. Then for any $t \in T_{\Sigma^e}(X^I)$, $\psi(t) = R_{S'}(\psi_\rho|_{\Sigma^e})(t) = \psi_\rho(t) = \rho(t)$.

So for any $x \in X_s$, $s \in S^e$, $\psi(\tau(x)) = \psi(t_a) = \rho(t_a) = a = \phi'(x)$, and so $\phi' = \psi \circ \tau$.

Together with $*$ this gives $\phi'(u) = \psi(\tau(u)) \sim_{E'}^{R_{S'}(A|_{\Sigma^e})} \psi(\tau(v)) = \phi'(v)$. Now, ϕ' was arbitrary so we get $\forall \phi' : T_{\Sigma^e}(X) \to R_{S'}(A|_{\Sigma^e})\ .\ \phi'(u) \sim_{E'}^{R_{S'}(A|_{\Sigma^e})} \phi'(v)$. Consider any $\phi : T_{\Sigma^e}(X) \to R_{S'}(A|_{\Sigma^e})/E'$. Fact 4.9 gives a $\phi' : T_{\Sigma^e}(X) \to R_{S'}(A|_{\Sigma^e})$ such that $\phi = \psi_{E'} \circ \phi'$. So for all $\phi : T_{\Sigma^e}(X) \to R_{S'}(A|_{\Sigma^e})/E'$ we have $\phi(u) = [\phi'(u)]_{E'} = [\phi'(v)]_{E'} = \phi(v)$, i.e. $R_{S'}(A|_{\Sigma^e})/E' \models u = v$. $\square$

Lemma 4.8 *Let $A \in Mod_\Sigma(E)$. For $u, v \in R_{S'}(T_\Sigma(X)|_{\Sigma^e})$,*

$$\vdash_{S'}^{\mathrm{FI}} u = v \ \Rightarrow\ \forall \psi : R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \to R_{S'}(A|_{\Sigma^e})\ .\ \psi(u) \sim_{E'}^{R_{S'}(A|_{\Sigma^e})} \psi(v)$$

Proof: Suppose $\vdash_{S'}^{\mathrm{FI}} u = v$, for $u, v \in R_{S'}(T_\Sigma(X)|_{\Sigma^e})$. Induction on the construction of $\vdash_{S'}^{\mathrm{FI}}$. Fix $\psi : R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \to R_{S'}(A|_{\Sigma^e})$ arbitrarily.

importE: Since $A \in Mod_\Sigma(E)$ we have for any $\varphi : T_\Sigma(X) \to A$ that $\varphi(u) = \varphi(v)$. Determine $\varphi_\psi : T_\Sigma(X) \to A$ by

$$\varphi_\psi(x) = \begin{cases} \psi(x), & x \text{ a variable in } R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \\ a_\perp, & x \text{ a variable otherwise in } X^S, \text{ for some choice } a_\perp \end{cases}$$

Then for $w \in R_{S'}(T_\Sigma(X)|_{\Sigma^e})$, we have $\varphi_\psi(w) = \psi(w)$, and so $\psi(u) = \varphi_\psi(u) = \varphi_\psi(v) = \psi(v)$. By *refl* of $\sim_{E'}^{R_{S'}(A|_{\Sigma^e})}$, $\psi(u) \sim_{E'}^{R_{S'}(A|_{\Sigma^e})} \psi(v)$.

induceE': Then $u = \tau(c[l])$, $\tau(c[r]) = v$ for some $\langle l, r \rangle \in E', c \in T_{\Sigma^e}(X)$ and $\tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e})$. By *induce* of $\sim_{E'}^{R_{S'}(A|_{\Sigma^e})}$ we have $\rho(c[l]) \sim_{E'}^{R_{S'}(A|_{\Sigma^e})} \rho(c[r])$ for $\rho : T_{\Sigma^e}(X) \to R_{S'}(A|_{\Sigma^e})$. Now $\psi \circ \tau : T_{\Sigma^e}(X) \to R_{S'}(A|_{\Sigma^e})$, and so $\psi(u) = \psi(\tau(c[l])) \sim_{E'}^{R_{S'}(A|_{\Sigma^e})} \psi(\tau(c[r])) = \psi(v)$.

refl, *sym* and *trans*: These are dealt with by the i.h. and *refl*, *sym* and *trans* of $\sim_{E'}^{R_{S'}(A|_{\Sigma^e})}$. $\square$

Fact 4.9 *Let Σ be arbitrary. For any Σ-algebras A and B, let $\sim$ be any Σ-congruence on B, and let $\phi : A \to B/\!\sim$ be a Σ-homomorphism. Then there exists a Σ-homomorphism $\phi' : A \to B$, such that $\phi = \psi_\sim \circ \phi'$ where $\psi_\sim$ is the Σ-homomorphism taking any b in B to its equivalence class $[b]_\sim$ in $B/\!\sim$.*

Proof: Let $\nu : B/\!\sim \to B$ be a choice function taking any equivalence class q in $B/\!\sim$ to a choice $b \in q$. Define $\phi' : A \to B$ as follows. Let A° be the free elements of A, i.e. those elements that are not in the range of the unique homomorphism from G_Σ to A. For every $a \in A^\circ$, define $\phi'(a) = \nu(\phi(a))$. This uniquely determines ϕ' as a Σ-homomorphism. Moreover, $\phi = \psi_\sim \circ \phi'$. $\square$

Example 4 The calculus $\vdash^{\mathrm{FRI}}_{S'}$ can thus be used in verifying the refinement postulated in Example 2, namely $\mathsf{Set} \rightsquigarrow \mathsf{SetbyCanonicalList}$. Referential opacity ensures the safe and sound interaction between the set E of equations associated with $\mathsf{List+}$ and the set E' of equations introduced in the quotienting step forming $\mathsf{SetbyCanonicalList}$. For instance, although $\vdash^{\mathrm{FRI}}_{S'} \mathsf{add}(x, \mathsf{add}(x, \mathsf{add}(y, \mathsf{empty}))) = \mathsf{add}(y, \mathsf{add}(x, \mathsf{add}(x, \mathsf{empty})))$, referential opacity hinders the inference

$$\vdash^{\mathrm{FRI}}_{S'} \mathsf{in}(y, \mathsf{remove}(y, y :: \mathsf{add}(x, \mathsf{add}(x, \mathsf{add}(y, \mathsf{empty}))))) = \\ \mathsf{in}(y, \mathsf{remove}(y, y :: \mathsf{add}(y, \mathsf{add}(x, \mathsf{add}(x, \mathsf{empty})))))$$

which would have given $\vdash^{\mathrm{FRI}}_{S'} \mathsf{true} = \mathsf{false}$. The inference is illegal because :: is a hidden operator symbol. Also, completeness is secured by the $\omega_{S'}$-rule for $S' = \{\mathrm{set}\}$. Thus we have $\vdash^{\mathrm{FRI}}_{S'} \mathsf{in}(x, \mathsf{remove}(x, s)) = \mathsf{false}$ because for this to hold it is only required that $\vdash^{\mathrm{FRI}}_{S'} \mathsf{in}(x, \mathsf{remove}(x, s\tau)) = \mathsf{false}$ holds for all instances $s\tau$ generated by empty and add. $\circ$

4.2 Coincidence of Initial Models

Fact 4.10 *If E is* sufficiently complete *w.r.t. Σ^e, i.e. for every ground term $g \in G_\Sigma$ there is a ground term $g^e \in G_{\Sigma^e}$ such that $E \vdash g = g^e$ then the class K^{FI} has an initial object, namely $G_\Sigma/E|_{\Sigma^e}/E'$.*

Proof: Suppose E is sufficiently complete w.r.t. Σ^e. Then for any $[g]_E \in G_\Sigma/E|_{\Sigma^e}$ there is a $g^e \in G_{\Sigma^e}$ s.t. for the unique Σ^e-homomorphism from G_{Σ^e} to $G_\Sigma/E|_{\Sigma^e}$ we have $\phi(g^e) = [g]_E$. Hence $G_\Sigma/E|_{\Sigma^e}$ is a computation structure and then so is $G_\Sigma/E|_{\Sigma^e}/E'$. Then any Σ^e-homomorphism from $G_\Sigma/E|_{\Sigma^e}/E'$ to any Σ^e-algebra must be unique. Clearly $G_\Sigma/E|_{\Sigma^e}/E'$ is in K^{FI}, so it remains to show for an arbitrary $B \in K^{\mathrm{FI}}$ that there exists a Σ^e-homomorphism from $G_\Sigma/E|_{\Sigma^e}/E'$ to B. Now, in general a Σ-computation structure I is initial in a class C of Σ-algebras iff for all $g, g' \in G_\Sigma$, $I \models g = g' \;\Leftrightarrow\; \forall A \in C \,.\, A \models g = g'$. By this, it suffices to show for all $g, g' \in G_{\Sigma^e}$ that $G_\Sigma/E|_{\Sigma^e}/E' \models g = g' \;\Rightarrow\; T_\Sigma(X)/E|_{\Sigma^e}/E' \models g = g'$, which is easy. $\square$

If E is not sufficiently complete in the sense above, then K^{FI} may or may not have an initial object.

Example 5 Let $\Sigma = \langle \{s\}, \{a : s, b : s, h : s \to s, h' : s \to s\} \rangle$ and $\Sigma^e = \langle \{s\}, \{a : s, b : s\} \rangle$. Let $E = \{h'(a) = b, h'(b) = b, h'(h(x)) = a\}$. Then E is not sufficiently complete w.r.t. Σ^e. Let $E' = \emptyset$. In this case K^{FI} has Σ^e algebras with carriers of arbitrary finite cardinality (among others). An initial algebra of K^{FI} must have separate interpretations for a and b for there to be a Σ^e-homomorphism to every other algebra in K^{FI}. But any algebra with two elements must have at least three, because E then forces $h(a)$ and $h(b)$ to denote elements different from a and b. Hence, any candidate for the initial algebra has junk w.r.t. Σ^e, which can be interpreted in more than one way by Σ^e-homomorphisms. ○

Example 6 Let $\Sigma = \langle \{s\}, \{a : s, b : s, h : s \to s, h' : s \to s\} \rangle$ and $\Sigma^e = \langle \{s\}, \{a : s, b : s\} \rangle$. Let $E = \{h'(a) = b, h'(b) = b\}$. Let $E' = \emptyset$. Although E is not sufficiently complete w.r.t. Σ^e, K^{FI} has in this case an initial algebra, namely G_{Σ^e}/E. This is possible because there is nothing stopping $h(a)$ and $h(b)$ being interpreted to one of a and b in a two element carrier. ○

Theorem 4.11 *Suppose E is* sufficiently complete *w.r.t. Σ^e, i.e. for every ground term $g \in G_\Sigma$ there is a ground term $g^e \in G_{\Sigma^e}$ such that $E \vdash g = g^e$. By Fact 4.10 the initial object $G_\Sigma/\!\approx\!|_{\Sigma^e}$ of K^{FI} exists. Then for $S' = S$,*

$$R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E' \cong G_\Sigma/E|_{\Sigma^e}/E'$$

Proof: It suffices to show for $S' = S$ that $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e}) \cong G_{\Sigma^e}/(\sim_E^{T_\Sigma(X)}|_{\Sigma^e}) \cong G_\Sigma/E|_{\Sigma^e}$.

Any $q \in R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})$ has the form $[t]_{\sim_E^{T_\Sigma(X)}|_{\Sigma^e}}$ where t can be chosen such that t is denotable by a term in $T_{\Sigma^e}(X^I) = G_{\Sigma^e}$, i.e. $t \in G_{\Sigma^e}$. Clearly then, $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})$ is isomorphic to $G_{\Sigma^e}/\!\sim_E^{T_\Sigma(X)}|_{\Sigma^e}$.

On the other hand, any congruence class q' of the initial model $G_\Sigma/E|_{\Sigma^e}$ has the form $[g]_{\sim_E^{G_\Sigma}}$ where g can be chosen in G_{Σ^e} by the assumption of sufficient completeness. Then ι given by $\iota([g]_{\sim_E^{T_\Sigma(X)}|_{\Sigma^e}}) = [g]_{\sim_E^{G_\Sigma}}$ is a Σ^e-isomorphism if for $g, g' \in G_{\Sigma^e}$, $g \sim_E^{T_\Sigma(X)} g' \;\Leftrightarrow\; g \sim_E^{G_\Sigma} g'$, which is easy to show. □

Note for $S' = S$ that $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E'$ is initial in $K_{S'}^{\mathrm{FRI}}$ by virtue of it being the classifying model, and a Σ^e-computation structure (meaning there is a unique homomorphism from G_{Σ^e} to $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E'$. Generally, a Σ-computation structure I is initial in a class C of Σ-algebras iff for all $g, g' \in G_\Sigma$, $I \models g = g' \;\Leftrightarrow\; \forall A \in C \,.\, A \models g = g'$).

As noted in section 2.3 the more general result that $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E'$ is free on X^I for K^{FI}, does not hold in general. This is due to the reduct construct rendering carrier elements as junk that might not be removed by restricting on S'. This then leaves junk in $R_{S'}(T_\Sigma(X)/E|_{\Sigma^e})/E'$ w.r.t. $T_{\Sigma^e}(X^I)$ that might be mapped to several values in some algebra in K^{FI}, giving rise to several homomorphisms.

4.3 Forget-Identify-Restrict (FIR)

The FRI structure (2) is not equivalent to the structure in which the restrict step is done outermost, i.e. FRI is not equivalent to forget-identify-restrict (FIR), for a counter-example see [EKMP82]. Let $K^{\mathrm{FIR}}_{S'} = \{R_{S'}(A|_{\Sigma^e}/E') \mid A \in Mod_\Sigma(E)\}$. A sound and complete calculus $\vdash^{\mathrm{FIR}}_{S'}$ for $K^{\mathrm{FIR}}_{S'}$ is given by the following rule. For all $u, v \in T_{\Sigma^e}(X)$,

$$\omega_{S'} : \frac{\forall \tau : T_{\Sigma^e}(X) \to R_{S'}(T_\Sigma(X)|_{\Sigma^e}) \ . \ \vdash^{\mathrm{FI}}_{\emptyset} \tau(u) = \tau(v)}{\vdash^{\mathrm{FIR}}_{S'} u = v}$$

Again, $\vdash^{\mathrm{FI}}$ is subsumed by $\vdash^{\mathrm{FIR}}_{S'}$. Also, for $S' = S$, we have that the classifying model of $K^{\mathrm{FIR}}_{S'}$ is isomorphic to the initial model of K^{FI}.

5 Discussion and Conclusions

We are concerned here with the idea of enforcing protective abstraction barriers in proof methods that reflect abstraction barriers in (the semantics of) programs and more generally in program specifications. In this report we have shown that for equational forms of the FI and FRI approaches to refinement, it suffices to use primitive equational logic with an abstraction barrier in the form of referential opacity. Although we have only discussed the flat case, we claim generality in light of the normal form result for specifications [Wir93, FC92, Cen94] and the argument that **quotient** should only be employed outermost [BST].

The calculi exhibited in this report could form a basis upon which adaptations of semi-automated proof systems could be done. For instance, the referential opacity present in the calculi suggests altering the rewrite and completion-based method of *proof by consistency* by constraining the generation of critical pairs according to the context in the overlap of rewrite rules. Note that the results concerning the coincidence of initial models is then relevant. In particular, the $\omega_{S'}$ rule of the FRI calculus is relevant for the FI case at initiality. Note also, that the $\omega_{S'}$ rule is particularly interesting when constructors are hidden as in Example 2. Under certain sufficient completeness conditions, the rule states that it is enough to do induction over "abstract constructors", like empty and add, even when functions are defined over "concrete" constructors, as is remove over nil and :: in Example 2.

From this, it would seem prudent to not only do proofs by induction over abstract constructors, but to also do definitions by induction over abstract constructors. This is indeed a valid argument, but there is an important change of perspective involved. If we were to specify by induction on abstract constructors, this would not use implementation dependent information and would hence be equivalent to specifying outside the abstraction barrier. For example, specifying remove over empty and add would be equivalent to doing

$$
\begin{array}{l}
\textbf{spec}\ \mathsf{List{+}'}\ \textbf{is}\\
\quad\textbf{enrich}\\
\quad\quad\textbf{derive}\ (\textit{Specification enriching}\ \mathsf{List}\ \textit{by}\ \mathsf{empty}, \mathsf{add}, \mathsf{in})\ \textbf{by}\ \iota[\mathsf{set} \mapsto \mathsf{list}]\\
\quad\textbf{by}\\
\quad\quad\textbf{ops}\ \mathsf{remove} : \mathsf{nat} \times \mathsf{set} \to \mathsf{set}\\
\quad\quad\textbf{axioms}\ \ \mathsf{remove}(x, \mathsf{empty}) = \mathsf{empty}\\
\quad\quad\quad\quad\quad\quad\mathsf{remove}(x, \mathsf{add}(y, s)) = \mathsf{if}\ x =_{\mathsf{nat}} y\ \mathsf{then}\ \mathsf{remove}(x, s)\\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\mathsf{else}\ \mathsf{add}(y, \mathsf{remove}(x, s))
\end{array}
$$

This does not allow implementation details to be used in optimising the implementation of remove as before, which is fine if that is what we want. This is merely the decision of when during a refinement process the implementation of a function should be fixed.

Behavioural proofs are often simplified greatly by introducing behavioral lemmas, i.e. propositions that are true according to behavioural equality but not true literally. The referentially opaque calculus ensures an appropriate abstraction barrier so that such lemmas may be inserted soundly into the proof environment. The assumption of *stability* introduced in [Sch86] ensures the safe insertion of such lemmas too. Although the notion of stability applies at a different level of the refinement process, there seems to be a relationship worthwhile looking at between the abstraction barriers provided by stability and referential opacity.

In the setting of behavioural refinement, semantically one does not need quotienting, and the restrict operator is also superfluous since one can speak in terms of partial behavioural congruences. However, for proving behavioural refinement steps, the calculi developed here are useful. One way of proving behavioural refinement steps is to consider the *behaviour* of algebras [Hen97]. The behaviour of an algebra is its quotient by a behavioural congruence, and in the case where this congruence is partial, a restrict step has to be done. Hence we regain the FI and FRI situation.

We remarked in Sect. 3 that the calculus in Definition 1 is given by the Σ^e-congruence on $T_\Sigma(X)|_{\Sigma^e}$ induced by the set of Σ^e-equations E^{FI} given by (3). We could therefore have defined K^{FI} as $\{C|_{\Sigma^e}/E^{\mathrm{FI}} \mid C \in \Sigma\mathbf{Alg}\}$, and expressed the classifying model as $T_\Sigma(X)|_{\Sigma^e}/E^{\mathrm{FI}}$. In a sense this flattens the structured view of the semantics we had in the former characterisation of K^{FI}. In fact we can flatten things even more by considering the following relation.

Definition 3 (Referentially Opaque Congruence) *For any set E of Σ-equations and any set E' of Σ^e-equations, define $\approx^A$ on any Σ-algebra A as the least equivalence containing*

$$
\begin{array}{l}
\{\langle \phi(c[l]), \phi(c[r])\rangle \mid \langle l, r\rangle \in E, c \in T_\Sigma(X),\ \phi : T_\Sigma(X) \to A\} \ \bigcup\\
\{\langle \phi(c[l]), \phi(c[r])\rangle \mid \langle l, r\rangle \in E', c \in T_{\Sigma^e}(X),\ \phi : T_{\Sigma^e}(X) \to A|_{\Sigma^e}\}
\end{array}
$$

i.e. the relation inductively defined by

$$induceE : \frac{}{\phi(c[l]) \approx^A \phi(c[r])}; \quad \langle l, r\rangle \in E, c \in T_\Sigma(X),\ \phi : T_\Sigma(X) \to A$$

$$induceE' : \frac{}{\phi(c[l]) \approx^A \phi(c[r])}; \quad \langle l, r\rangle \in E', c \in T_{\Sigma^e}(X),\ \phi : T_{\Sigma^e}(X) \to A|_{\Sigma^e}$$

$$refl : \frac{}{a \approx^A a} \qquad sym : \frac{a \approx^A a'}{a' \approx^A a} \qquad trans : \frac{a \approx^A a'', a'' \approx^A a'}{a \approx^A a'}$$

Note that $\approx^A$ is not (in general) a Σ-congruence on A. However, if we define the reduct $\approx^A|_\sigma$ of $\approx^A$ w.r.t. a signature morphism $\sigma : \Sigma' \to \Sigma$ as for congruences, i.e. $(\approx^A|_\sigma)_s = \approx^A{}_{\sigma(s)}$, then $\approx^A|_{\Sigma^e}$ is a Σ^e-congruence on $A|_{\Sigma^e}$. It is easy to show:

Fact 5.1 *Let* $\overline{K}^{\mathrm{FI}} = \{A|_{\Sigma^e}/(\approx^A|_{\Sigma^e}) \mid A \in \Sigma\mathbf{Alg}\}$. *Then* $\overline{K}^{\mathrm{FI}} = K^{\mathrm{FI}}$

In a behavioural context, we can in fact use congruences of the form $\approx^A|_{\Sigma^e}$ to give behaviours of algebras. Here we manage to generate the behavioural congruence in the manner of a congruence induced by equations, even in the presence of problematic hidden operators. By Fact 5.1 the calculus $\vdash^{\mathrm{FI}}$ lets us do proofs accordingly.

Now having considered $A|_{\Sigma^e}/(\approx^A|_{\Sigma^e})$, one might ask what $(A/\!\approx^A)|_{\Sigma^e}$ is, and in particular what $A/\!\approx^A$ is. For a Σ-algebra A and a congruence $\sim$ on A, $(A/\!\sim)|_{\Sigma^e}$ is of course equal to $A|_{\Sigma^e}/(\sim|_{\Sigma^e})$. Since $\approx$ is not in general a congruence on A, considering an obvious naive definition of $A/\!\approx^A$ wouldn't necessarily give a Σ-algebra. There are however, reasons to consider various other definitions of $A/\!\approx^A$. Let us give the as yet tentative structure $A/\!\approx^A$ the name Q_A.

For example, we could define Q_A as the Σ^e-algebra, having as carriers $A_s/\!\approx^A_s$ for each $s \in S$, and standard interpretations of each $f \in \Omega^e$. Note that Q_A then has carriers for all sorts in S. If we want a direct proof of the soundness and completeness of the calculus $\vdash^{\mathrm{FI}}$ rather than deriving it is a sub-case of the FRI result, this definition of Q_A enables an easy direct proof. This is due to the flat structure, but also to the ability of $Q_{T_\Sigma(X)} = T_\Sigma(X)/\!\approx^{T_\Sigma(X)}$ to characterise all derivations involved in $\vdash^{\mathrm{FI}}$, also the ones giving the equations (theorems) imported by *importE*.

Hidden operator symbols, i.e. those in $\Omega \setminus \Omega^e$ have no interpretation in the above tentative definition of Q_A. Viewing Q_A as a Σ-structure would demand that hidden operator symbols get an interpretation. These interpretations could be intensional operators, i.e. operators not respecting the equality predicate of the data type. For instance, looking to Example 1, count^{Q_A} would not respect the equality predicate given by the idempotency and associativity axioms for add^{Q_A}. Considering intensional operators in data types is not an alien concept. Hiding and quotienting do not only occur in software development, but abound elsewhere in mathematics too. At the very foundations of real analysis, the reals are defined as a quotient of a set of Cauchy-sequences. The n'th approximant function is then intensional, and from a constructivist point of view, so is every discontinuous function [TvD88]. Indeed in a constructive setting it might be prudent to add intensional operators to a data type [Hof95] (a choice operator for quotient types). Note then that in $(A/\!\approx^A)|_{\Sigma^e}$

the reduct operator is now applied outermost, in contrast to the FI- and FRI-models we considered earlier. One could speculate if there might be an adjunction between the appropriate reduct functor and some free functor. The free functor would then add intensional operators, and it would seem imperative to find a definition of structures with intensional operators, and in particular a definition of Q_A as a Σ-structure, such that intensional operators are added in a manner in which they bring with them the appropriate abstraction barrier.

Acknowledgments

Thanks are due to Don Sannella, Martin Hofmann for valuable suggestions and comments. This research has been supported by NFR (Norwegian Research Council) grant 110904/41.

References

[BST] M. Bidoit, D. Sannella, and A. Tarlecki. Behavioural encapsulation. CoFI Language Design Study Note, 1996. Available at `ftp://ftp.brics.dk/Projects/CoFI/StudyNotes/Lang/MB+DTS+AT-1.ps.Z`.

[Cen94] M.V. Cengarle. *Formal Specification with Higher-Order Parameterization.* PhD thesis, Fakultät für Mathematik, LMU, München, 1994.

[DO91] O.-J. Dahl and O. Owe. Formal development with ABEL. Forskningsraport 552, Institutt for informatikk, Universitetet i Oslo, 1991.

[EKMP82] H. Ehrig, H.-J. Kreowski, B. Mahr, and P. Padawitz. Algebraic implementation of abstract data types. *Theoretical Computer Science*, 20:209–263, 1982.

[FC92] J. Farrés-Casals. *Verification in ASL and Related Specification Languages, Report CST-92-92.* PhD thesis, Dept. of Computer Science, University of Edinburgh, 1992.

[Hen97] R. Hennicker. Structured specifications with behavioural operators: Semantics, proof methods and applications. Habilitationsschrift, Inst. für Informatik, LMU, München, 1997.

[Hof95] M. Hofmann. *Extensional Concepts in Intensional Type Theory, Report CST-117-95 and Tech. Report ECS-LFCS-95-327.* PhD thesis, Dept. of Computer Science, University of Edinburgh, 1995.

[HS96] M. Hofmann and D. Sannella. On behavioural abstraction and behavioural satisfaction in higher-order logic. *Theoretical Computer Science*, 167:3–45, 1996.

[Mor94] C. Morgan. *Programming from Specifications, 2nd ed.* Prentice Hall International Series in Computer Science; C.A.R. Hoare, Series Editor. Prentice-Hall, UK, 1994.

[QG93] X. Qian and A. Goldberg. Referential opacity in nondeterministic data refinement. *ACM LoPLaS*, 2(1–4):233–241, 1993.

[San91] D. Sannella. Formal development in extended ML. In *Proc. 3rd BCS/FACS Workshop on Refinement*, pages 99–130, Hursley Park, 1991. Springer Workshops in Computing.

[Sch86] O. Schoett. *Data Abstraction and the Correctness of Modular Programming*. PhD thesis, University of Edinburgh, 1986.

[SS90] H. Søndergaard and P. Sestoft. Referential transparency, definiteness and unfoldability. *Acta Inform.*, 27(6):505–517, 1990.

[ST88] D. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: implementations revisited. *Acta Inform.*, 25(3):233–281, 1988.

[ST97] D. Sannella and A. Tarlecki. Essential concepts of algebraic specification and program development. *Formal Aspects of Computing*, 9:229–269, 1997.

[TvD88] A.S. Troelstra and D. van Dalen. *Constructivism in Mathematics, An Introduction*, volume 121 of *Studies in Logic and The Foundations of Mathematics*. North Holland, 1988.

[Wir90] M. Wirsing. Algebraic specification. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 13, pages 675–788. Elsevier, 1990.

[Wir93] M. Wirsing. Structured specifications: Syntax, semantics and proof calculus. In F.L. Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification, Intl. Summer School Marktoberdorf*, NATO ASI Series F, pages 411–442. Springer, 1993.

[Wir94] M. Wirsing. Algebraic Specification Languages: An Overview. In E. Astesiano, G. Reggio, and A. Tarlecki, editors, *Recent Trends in Data Type Specification*, LNCS, pages 81–115. Springer, 1994.