Equational Reasoning with Subtypes

Gary T. Leavens and Don Pigozzi

TR #02-07

July 2002

Keywords: behavioral subtype, subtyping, equational logic, proof theory.
2001 CR Categories: D.3.3 [*Programming Languages*] Language Constructs and Features — Abstract data types, classes and objects; F.3.1 [*Logics and Meanings of Programs*] Specifying and Verifying and Reasoning about Programs — Logics of programs, specification techniques; F.3.2 [*Logics and Meanings of Programs*] Semantics of Programming Languages — algebraic approaches to semantics; F.3.3 [*Logics and Meanings of Programs*] Studies of Program Constructs — object-oriented constructs; F.4.1 [*Mathematical Logic and Formal Languages*] Mathematical Logic — proof theory, model theory.
1991 Mathematics Subject Classification. Primary: 68Q65 Secondary: 68N05, 68N15, 68Q60.

Submitted for publication.

Department of Computer Science
226 Atanasoff Hall
Iowa State University
Ames, Iowa 50011-1041, USA

EQUATIONAL REASONING WITH SUBTYPES

GARY T. LEAVENS AND DON PIGOZZI

Abstract. Using equational logic as a specification language, we investigate the proof theory of behavioral subtyping for object-oriented abstract data types with immutable objects and deterministic methods that can use multiple dispatch. In particular, we investigate a proof technique for correct behavioral subtyping in which each subtype's specification includes terms that can be used to coerce its objects to objects of each of its supertypes. We show that this technique is sound, using our previous work on the model theory of such abstract data types. We also give an example to show that the technique is not complete, even if the methods do not use multiple dispatch, and even if types specified are term-generated. In preparation for the results on equational subtyping we develop the proof theory of a richer form of equational logic that is suitable for dealing with subtyping and behavioral equivalence. This gives some insight into question of when our proof techniques can be make effectively computable, but in general behavioral consequence is not effectively computable.

1. Introduction

The concept of behavioral subtyping plays a foundational role in the specification and verification of object-oriented programs. While we and others have investigated model-theoretic notions of behavioral subtyping [5, 21, 23, 28], model-theoretic techniques do not lead directly to effectively computable ways to prove correct behavioral subtyping. For this purpose one must look to proof-theoretic techniques. The main motivation for our study is to bridge the gap between more practical proof-theoretic work and our sound and complete model-theoretic characterization of behavioral subtyping (for immutable types with deterministic methods that may use multiple dispatch) [23].

In this paper we work with equational specifications and a particular technique for verifying correct behavioral subtyping. Equational logic [11] is at the heart of many proof-theoretic techniques for reasoning about programs. Even in a Hoare logic, one often relies on equational reasoning to prove various facts about expressions. However we must deal

Date: June 7, 2002.

1991 *Mathematics Subject Classification.* Primary: 68Q65 Secondary: 68N05, 68N15, 68Q60.

2001 *CR Categories.* D.3.3 [*Programming Languages*] Language Constructs and Features — Abstract data types, classes and objects; F.3.1 [*Logics and Meanings of Programs*] Specifying and Verifying and Reasoning about Programs — Logics of programs, specification techniques; F.3.2 [*Logics and Meanings of Programs*] Semantics of Programming Languages — algebraic approaches to semantics; F.3.3 [*Logics and Meanings of Programs*] Studies of Program Constructs — object-oriented constructs; F.4.1 [*Mathematical Logic and Formal Languages*] Mathematical Logic — proof theory, model theory.

Key words and phrases. Behavioral subtyping, equational logic, proof theory.

The work of both authors is supported in part by the National Science Foundation under grants CCR-9803843 and CCR-0097907. The work of Leavens was also supported in part by NSF grant CCR-0113181.

with a more complex form of equational logic that is suitable for reasoning about both subtyping and behavioral equivalence.

We consider equational specifications that are augmented, for each type, by a specification of how to coerce an object of that type to each of its direct supertypes. Such a formal system of coercion terms is similar to the system of implicit coercion functions considered in Reynolds's category-sorted algebras [32]. These coercions are also present in the specifications that other authors have used in their work on behavioral subtyping [2, 5, 10, 26].

In our work, the coercions take the from of equations, and we show that if all the coercion equations are behavioral consequences of the given equational specification, then the given specification is necessarily correctly behaviorally subtyped. This is done by applying the algebraic soundness theorem from our work on the model theory of behavioral subtyping [23].

We also investigate the limitations of this technique. In particular, we give an equational specification of an ADT that is correctly behaviorally subtyped, but for which no system of coercion terms exists. Since we allow multiple dispatch and models that are not term-generated, we also discuss how our results are affected by more restrictive conditions on our specification and its models.

The basis of the theory of correct behavioral subtyping is behavioral equivalence. A systematic development of a part of the theory of behavioral equivalence, in the context of correct subtyping, is given in [22]. A more extensive development that does not involve subtyping can be found in a series of papers by Goguen and Malcolm on hidden algebras; see [15, 16] and the many papers referenced there.

To reason effectively about behavioral equivalence and subtyping, we introduce the concept of behavioral consequence. Coercion allows us to reduce the problem of correct behavioral subtyping to one of verifying that certain equations are behavioral consequences of the given specification. While the completeness and soundness theorems for equational logic show that ordinary equational consequence is effectively computable, behavioral consequence is in general not [6]. The first part of the paper is devoted to developing a proof theory for behavioral consequence that in many practical situations does give an effective method for verifying behavioral consequence, and hence that a given specification is correctly behaviorally subtyped.

The proof theory of behavioral consequence is based on coinduction. Various methods of applying coinduction in a reasonably efficient way have been investigated in the literature (see for example [15, 33, 34]). We focus in this paper on one such method that uses the notion of a local context. One of the main results is the closure of behavioral consequence under equational consequence, i.e., that any equation that is provable in ordinary equational logic from a set of behavioral consequences of an equational specification is itself a behavioral consequence. This turns out to be an effective method for proving correct behavioral subtyping in many cases where all the equations associated with a given set of coercion functions can be proved by the methods of equational logic from just a few judiciously chosen ones which have been shown to be behavioral consequences directly by coinduction. An example of this kind is given in some detail. It is part of a running example that includes the equational specification of an object-oriented type of geometric points with its subtype, colored points. We return to this example several times throughout the paper to illustrate the more important definitions and results as they are obtained. In particular we verify by means of coercions that the specification is correctly behaviorally subtyped.

In the last part of the paper we discuss some further aspects of this work. For example we show why, for the study of correct behavioral subtyping, order-sorted algebras do not form an appropriate basis for the model theory. There is also a discussion of related work and conclusions. The paper ends with an appendix on the model theory of correct behavioral subtyping.

2. Equational Logic with Subtyping

In this section, we specify the equational logic that forms the basis of the logic of behavioral equivalence developed in the next section. The novel feature is the way subtyping is incorporated in the logic.

2.1. **Signatures with Subtyping.** We use essentially the same notion of a signature with subtyping from our earlier work [21, 23], which is itself adapted from the signatures Reynolds uses [32]. However under the influence of the "hidden algebra school" [15, 16] the separation into visible and hidden parts is made more explicit.

Definition 2.1. A *visible data signature* $\varPsi = \langle \mathrm{VIS}, \mathrm{Op}_{\mathrm{VIS}}, \mathrm{AdmisType}_{\mathrm{VIS}} \rangle$ consists of:

(i) A nonempty set VIS of *visible types.*

(ii) A nonempty set $\mathrm{Op}_{\mathrm{VIS}}$ of *visible operation symbols.*

(iii) A function $\mathrm{AdmisType}_{\mathrm{VIS}}$ from $\mathrm{Op}_{\mathrm{VIS}}$ to VIS^+, the set of finite nonempty sequences of element of VIS. For each $g \in \mathrm{Op}_{\mathrm{VIS}}$, $\mathrm{AdmisType}(g) = (V_1, \dots, V_n, V)$, which is normally written in the form $V_1, \dots, V_n \to V$, is called the *admissible type of g*. Moreover, $(V_1, \dots, V_n)$ is called the *admissible type domain* and V the *admissible result type* of g. □

Definition 2.2. A *hidden signature with subtyping* $\varSigma = \langle \varPsi, \mathrm{HID}, \leq, \mathrm{Op}_{\mathrm{HID}}, \mathrm{ResType}_{\mathrm{HID}} \rangle$ consists of:

(i) A visible signature $\varPsi = \langle \mathrm{VIS}, \mathrm{Op}_{\mathrm{VIS}}, \mathrm{AdmisType}_{\mathrm{VIS}} \rangle$.

(ii) A nonempty set HID of *hidden types* disjoint from VIS. By a *type* we mean either a visible or hidden type, and the set of all types is denoted by TYPE; thus $\mathrm{TYPE} = \mathrm{VIS} \cup \mathrm{HID}$.

(iii) A *subtype relation* $\leq$, which is a preorder on HID. If $S \leq T$, then S is a *subtype* of T, and T is a *supertype* of S. $\leq$ is extended to a preordering of all of TYPE by setting $V \leq V$ for each $V \in \mathrm{VIS}$; thus $\leq$ is the discrete ordering on VIS.

(iv) An $\mathbb{N}$-indexed family $\mathrm{Op}_{\mathrm{HID}} = \langle \mathrm{Op}_{\mathrm{HID},n} : n \in \mathbb{N} \rangle$ of *hidden operation symbols*, where $\mathbb{N}$ is the set of natural numbers $\{0, 1, 2, \dots\}$. $\mathrm{Op}_{\mathrm{HID},n}$ is the set of hidden operation symbols of rank n; it is assumed to be disjoint from $\mathrm{Op}_{\mathrm{VIS}}$ for each n.

(v) A $\mathbb{N}$-indexed family $\mathrm{ResType}_{\mathrm{HID}} = \langle \mathrm{ResType}_{\mathrm{HID},n} : n \in \mathbb{N} \rangle$ of partial functions $\mathrm{ResType}_{\mathrm{HID},n} : \mathrm{Op}_n \times \mathrm{TYPE}^n \to \mathrm{TYPE}$ for each $n \in \mathbb{N}$ with the following two properties.

  (a) If $\mathrm{ResType}_{\mathrm{HID},n}(g, S_1, \dots, S_n) = S$, then at least one of $S_1, \dots, S_n, S$ is a hidden type.

(b) Each of the functions $\text{ResType}_{\text{HID},n} : \text{Op}_{\text{HID},n} \times \text{TYPE}^n \to \text{TYPE}$ is monotonic in its second argument in the sense of the following *subtype condition*: $(S_1, \dots, S_n) \leq (T_1, \dots, T_n)$ implies

$$\text{ResType}(g, S_1, \dots, S_n) \leq \text{ResType}(g, T_1, \dots, T_n).$$

It is implicit in the subtype condition that if $\text{ResType}_{\text{HID},n}(g, T_1, \dots, T_n)$ is defined, then so is $\text{ResType}_{\text{HID},n}(g, S_1, \dots, S_n)$ if $(S_1, \dots, S_n) \leq (T_1, \dots, T_n)$. □

$\text{ResType}_{\text{HID}}$ gives the *nominal* result type for a given hidden operation and sequence of argument types.

The hidden operations of a signature are polymorphic in the sense that each may have multiple admissible types. A type such as $T_1, \dots, T_n \to T$ is called an *admissible type of* g if $\text{ResType}_{\text{HID},n}(g, T_1, \dots, T_n) = T$; in this case $(T_1, \dots, T_n)$ is said to be an *admissible type domain*, and T an *admissible result type* of g. The defining properties of $\text{ResType}_{\text{HID}}$ and the subtyping relation guarantee that, if $T_1, \dots, T_n \to T$ and $S_1, \dots, S_n \to S$ are two admissible types of a hidden operation g and T is visible, then so is S and in fact they are equal. Similarly for argument types: for each $i \leq n$, if T_i is visible, then so is S_i and they are equal.

The combined set of visible and hidden operation symbols is denoted by Op and the range of $\text{ResType}_{\text{HID}}$ is extended to include visible operations. Thus, for each visible $g \in \text{Op}_n$, $\mathit{ResSort}(g, T_1, \dots, T_n)$ is defined and equal to T if and only if $T_1, \dots, T_n \to T$ is the (unique) admissible type of g.

The hidden operations are also called *methods*; these are partitioned into *attributes*, or *deconstructors*, whose admissible result type is visible (and hence unique), and *constructors*, whose admissible result types are all hidden. The methods are also partitioned into *unimethods*, those whose admissible type domains contain only one hidden type, and the *multimethods*, whose admissible type domains contain two or more hidden types. (As we have observed, the definition of a signature with subtyping guarantees all admissible type domains contain the same number of hidden types).

We assume a fixed countably infinite universe Var of (untyped) variable symbols. The set Tm_Σ of all (*untyped*) *Σ-terms* are defined inductively in the usual way: $\text{Var} \subseteq \text{Tm}_\Sigma$ and, for every $n \in \mathbb{N}$, $g \in \text{Op}_n$, and $t_1, \dots, t_n \in \text{Tm}_\Sigma$, $g(t_1, \dots, t_n) \in \text{Tm}_\Sigma$; in particular, $g \in \text{Tm}_\Sigma$ for every $g \in \text{Op}_0$. A term that is not a variable is *completely visible* if every operation occurring in it is visible. It is *partially hidden* otherwise, i.e., if it contains an occurrence of at least one hidden operation.

Example 2.3. As an example of a signature with subtyping, let Σ_{pt} be a hidden signature with the three types: `int`, `Pt`, and `CPt`. The idea behind this example is that the type `Pt` represents two-dimensional points, and the type `CPt` represents two-dimensional points that also have a color. In this example, the only visible type is `int`, and the subtype relation is such that $T \leq T$, for each type T, and also `CPt` $\leq$ `Pt`. The operations and their admissible types for Σ_{pt} are given in Figure 1. The operations on points also work on colored points, as they must according to the subtype condition. For example, the operation `x` can be used to extract the x coordinate from a point or a colored point. Note that when one of

Visible operations

$$\begin{array}{ll}
\mathtt{0} : \mathtt{int}, & \mathtt{1} : \mathtt{int}, \\
\mathtt{succ} : \mathtt{int} \to \mathtt{int}, & \mathtt{pred} : \mathtt{int} \to \mathtt{int}, \\
\mathtt{add} : \mathtt{int}, \mathtt{int} \to \mathtt{int}, & \mathtt{sub} : \mathtt{int}, \mathtt{int} \to \mathtt{int}.
\end{array}$$

Unimethod attributes

$$\begin{array}{ll}
\mathtt{x} : \mathtt{Pt} \to \mathtt{int}, & \mathtt{y} : \mathtt{Pt} \to \mathtt{int}, \\
\mathtt{x} : \mathtt{CPt} \to \mathtt{int}, & \mathtt{y} : \mathtt{CPt} \to \mathtt{int}, \\
\mathtt{color} : \mathtt{CPt} \to \mathtt{int}. &
\end{array}$$

Multimethod attributes

$$\begin{array}{ll}
\mathtt{xdiff} : \mathtt{Pt}, \mathtt{Pt} \to \mathtt{int}, & \mathtt{xdiff} : \mathtt{CPt}, \mathtt{Pt} \to \mathtt{int}, \\
\mathtt{xdiff} : \mathtt{Pt}, \mathtt{CPt} \to \mathtt{int}, & \mathtt{xdiff} : \mathtt{CPt}, \mathtt{CPt} \to \mathtt{int}.
\end{array}$$

Unimethod constructors

$$\begin{array}{ll}
\mathtt{PtOrigin} :\to \mathtt{Pt}, & \mathtt{CPtOrigin} :\to \mathtt{Cpt}, \\
\mathtt{movex} : \mathtt{Pt}, \mathtt{int} \to \mathtt{Pt}, & \mathtt{movey} : \mathtt{Pt}, \mathtt{int} \to \mathtt{Pt}, \\
\mathtt{movex} : \mathtt{CPt}, \mathtt{int} \to \mathtt{CPt}, & \mathtt{movey} : \mathtt{CPt}, \mathtt{int} \to \mathtt{CPt}, \\
\mathtt{changec} : \mathtt{CPt}, \mathtt{int} \to \mathtt{CPt}, &
\end{array}$$

FIGURE 1. Operations and their admissible types for Σ_{pt}.

the operations `movex` or `movey` is applied to an object of type `CPt`, it is required by the signature to return a `CPt` object. □

A signature with subtyping is capable of modeling "multimethods." In an object-oriented (OO) programming language with multimethods, a call to an operation can be dispatched to a method body based on the types of more than one of its arguments. An example is the operation `xdiff` in the signature Σ_{pt} above. Many OO languages do not support multimethods, however, so it is interesting to study more restricted forms of signatures. Many of these lie within the scope of the following definition. A hidden signature with subtyping, Σ, *has only unary methods*, or is a *unimethod signature*, if every every method is a unimethod, i.e., if, for every $n \in \mathbb{N}$ and every $g \in \mathrm{Op}_n$, if $T_1, \ldots, T_n$ is an admissible type domain of g, then $T_i \notin \mathrm{VIS}$ for at most one $1 \leq i \leq n$.

The signature Σ_{pt} above is not a unimethod signature. However, if one deletes the multimethod `xdiff`, then it becomes one. Note that `movex`, `movey`, `add`, and `sub` are unary methods, because `int` is a visible type.

In the general development below we assume a fixed signature with subtyping, Σ.

2.2. **Typing.** An element (t, T) of $\mathrm{Tm}_\Sigma \times \mathrm{TYPE}$ is called a *typing expression* or simply a *typing* and is written $t{:}T$. A finite sequence of typing expressions $\langle x_1{:}T_1, \ldots, x_n{:}T_n \rangle$ such that $x_1, \ldots, x_n$ are distinct variables is called a *type context*. H can also be regarded as a finite function, where $H(x_i) = T_i$, for $1 \leq i \leq n$, and $\{x_1, \ldots, x_n\}$ is the domain of H. The set of all type contexts is denoted by TCON.

H is a *type subcontext* of another type context K, and K is a *type supercontext* of H, in symbols $H \subseteq K$, if H is a subsequence of K, equivalently, if H as a function is the restriction of K to some subset of its domain. Type contexts H and K are *disjoint* if their domains are disjoint and they are *consistent* if $H \cup K$ is a type context (i.e., $H(x) = K(x)$ for each variable x common to the domains of H and K.

The following are the typing rules for terms [1, 21, 23].

$$(\mathit{ident}) \qquad \Sigma; H \vdash x{:}\, T, \quad \text{if } x{:}\, T \in H$$

$$(\mathit{op\text{-}call}) \qquad \frac{\Sigma; H \vdash t_1{:}\, T_1, \ldots, \Sigma; H \vdash t_n{:}\, T_n}{\Sigma; H \vdash g(t_1, \ldots, t_n){:}\, T}, \quad \text{if } \mathrm{ResType}(g, T_1, \ldots, T_n) = T,$$

A term t is of *nominal H-type* (or simply of *H-type*) S in Σ, in symbols $\Sigma; H \vdash_{\mathrm{NOM}} t{:}\, S$, if the typing expression $t{:}\, S$ is derivable from Σ and H using the two typing rules: (*ident*) and (*op-call*). If the H-type of a term is defined we say that it is *well H-typed.* For example, if $H_{p,x}$ is $\langle p{:}\, \mathtt{CPt},\ x{:}\, \mathtt{int}\rangle$, then $\mathtt{movex}(p, \mathtt{1})$ is well $H_{p,x}$-typed, in fact of $H_{p,x}$-type $\mathtt{CPt}$ because $\Sigma_{pt}; H_{p,x} \vdash_{\mathrm{NOM}} \mathtt{movex}(p, \mathtt{1}) : \mathtt{CPt}$. When we use the expression "$t[x_1{:}\, T_1, \ldots, x_n{:}\, T_n]$" synonymously for "$t$" it is understood that we are assuming that t is well H-typed with $H = \langle x_1{:}\, T_1, \ldots, x_n{:}\, T_n\rangle$. If the nominal H-type of a term is defined, it is easy to see that it is unique. Moreover, if t is well H-typed, it is well H'-typed for every type supercontext H' of H, and its nominal H- and H'-types are the same.

When the type context is irrelevant or understood from context we often say simply that a term is *well typed* and speak of its *nominal type* or even just of its *type.* Clearly each subterm of a well typed term is well typed. The typing expression "$t{:}\, T$" is often used synonymously for "t" under the implicit assumption that t is well typed and that T is its type. For example, $\mathtt{movex}(p, \mathtt{1})$ is well typed, because $\Sigma_{pt}; \langle p : \mathtt{CPt}\rangle \vdash_{\mathrm{NOM}} \mathtt{movex}(p, \mathtt{1}) : \mathtt{CPt}$.

If $t = t[x_1{:}\, T_1, \ldots, x_n{:}\, T_n]$, $(S_1, \ldots, S_n) \le (T_1, \ldots, T_n)$, and $t_1{:}\, S_1, \ldots, t_n{:}\, S_n$, then the result of simultaneously substituting $t_i{:}\, S_i$ for $x_i{:}\, T_i$ in t, for each $1 \le i \le n$, is denoted by $t[t_1, \ldots, t_n]$.

Lemma 2.4 (Subtype Condition for Terms)**.** *Suppose $\Sigma; H \vdash_{\mathrm{NOM}} t[x_1{:}\, T_1, \ldots, x_n{:}\, T_n] : T$, H' is a type context, and $\Sigma; H' \vdash_{\mathrm{NOM}} t_i{:}\, S_i$ for $i \le n$. If $(S_1, \ldots, S_n) \le (T_1, \ldots, T_n)$, then $t[t_1, \ldots, t_n]$ is well H'-typed, and if it is of nominal H'-type S, then $S \le T$.*

Proof. By structural induction on t using the subtype condition for operations. □

A term t is said to be *of subsumptive H-type S in Σ*, in symbols $\Sigma; H \vdash_{\mathrm{SUB}} t{:}\, S$, if the typing expression $t{:}\, S$ is derivable from Σ and H using the following rule of (*subsumption*), in addition to the rules (*ident*) and (*op-call*).

$$(\mathit{subsumption}) \qquad \frac{\Sigma; H \vdash t{:}\, S}{\Sigma; H \vdash t{:}\, T}, \quad \text{if } S \le T.$$

For example, besides its nominal type, $\mathtt{movex}(\mathtt{CPtOrigin}(\mathtt{0}), \mathtt{1}) : \mathtt{CPt}$ also has subsumptive type $\mathtt{Pt}$. According to the following lemma, whose proof is straightforward, a term has a subsumptive type if and only if it is well typed, and, moreover, its subsumptive types are exactly the supertypes of its nominal type.

Lemma 2.5 (Subsumption Lemma)**.** *If $\Sigma; H \vdash_{\mathrm{SUB}} t{:}\, S$, then t is well H-typed, and if T is its nominal H-type, then $T \le S$.* □

Another way of formulating this lemma is that, in the derivation of a subsumptive H-type of t, the subsumption rule need only be applied once, and as the last step of the derivation.

2.3. **Algebras.**

Definition 2.6. A *Σ-algebra* $\boldsymbol{A} = \langle A, \{ g^{\boldsymbol{A}} : g \in \mathrm{Op} \} \rangle$ consists of the following:

- A TYPE-indexed family of sets $A = \langle A_T : T \in \mathrm{TYPE} \rangle$ called the *carrier* of $\boldsymbol{A}$.
- A partial function $g^{\boldsymbol{A}}$: $\left(\bigcup_{S \in \mathrm{TYPE}} A_S\right)^n \to \bigcup_{S \in \mathrm{TYPE}} A_S$ for each $n \in \mathbb{N}$ and $g \in \mathrm{Op}_n$, called the *interpretation* of g, with the property that, for every admissible type $T_1, \dots, T_n \to S$ of g and every $(a_1, \dots, a_n) \in A_{T_1} \times \cdots \times A_{T_n}$, $g^{\boldsymbol{A}}(a_1, \dots, a_n)$ is defined and contained in $\bigcup_{U \leq S} A_U$. □

We follow Reynolds [32], in contrast to Goguen and Meseguer [13, 17], in not requiring $A_S \subseteq A_T$ when $S \leq T$. However Reynolds handles subtyping by means of an implicit coercion mapping between the domains A_S and A_T when $S \leq T$, and this has essentially the effect of requiring A_S and A_T to be disjoint when $S \neq T$. We take a middle ground. The domains of distinct types S and T need not be disjoint, nor do they have to be comparable when $S \leq T$, and we do not formally require coercion functions. Note that, if $(a_1, \dots, a_n) \in (A_{T_1} \times \cdots \times A_{T_n}) \cap (A_{S_1} \times \cdots \times A_{S_n})$, then there is no ambiguity in the value of $g^{\boldsymbol{A}}(a_1, \dots, a_n)$ when $T_1, \dots, T_n$ and $S_1, \dots, S_n$ are both admissible type domains of an operation symbol g, since the interpretation of g is global and not localized to the admissible type domains.

Example 2.7. A Σ_{pt}-algebra, $\boldsymbol{PT}$, is formed as follows. The carrier of the sort `int`, $PT_{\mathtt{int}}$, is $\mathbb{Z}$, the set $\{\dots, -2, -1, 0, 1, 2, \dots\}$ of integers. For the other sorts, the carrier sets are $PT_{\mathtt{Pt}} = \{\langle x, y\rangle : x \in \mathbb{Z}, y \in \mathbb{Z}\}$ and $PT_{\mathtt{CPt}} = \{\langle x, y, c\rangle : x \in \mathbb{Z}, y \in \mathbb{Z}, c \in \mathbb{Z}\}$. The interpretations of the operations are given in Figure 2. Note that the operations `movex` and `movey` change the color when their argument is of type `CPt`. □

It is convenient, in some contexts, to consider the more restricted notion of Goguen and Meseguer's order-sorted algebras [13, 17]. A Σ-algebra $\boldsymbol{A}$ is called *order-sorted* if $A_S \subseteq A_T$ when $S \leq T$. Thus, in order-sorted algebras, carrier sets satisfy the following property: $\bigcup_{S \leq T} A_S = A_T$.

The algebra $\boldsymbol{PT}$ is not order-sorted, since $A_{\mathtt{CPt}} \not\subseteq A_{\mathtt{Pt}}$. However, it is possible to transform $\boldsymbol{PT}$, into an order-sorted algebra by means of a general construction, described below.

Let $\boldsymbol{A} = \langle \{A_T : T \in \mathrm{TYPE}\}, \{ g^{\boldsymbol{A}} : g \in \mathrm{Op} \} \rangle$ be an arbitrary Σ-algebra. For each $T \in \mathrm{TYPE}$ let $A^\circ_T = \bigcup_{S \leq T} A_S$. Note that, because of the subtype condition, if $T_1, \dots, T_n$ is an admissible type domain of $g \in \mathrm{Op}_n$, then $g^{\boldsymbol{A}}$ is defined on $A^\circ_{T_1} \times \cdots \times A^\circ_{T_n}$. Thus $\langle \{A^\circ_T : T \in \mathrm{TYPE}\}, \{ g^{\boldsymbol{A}} : g \in \mathrm{Op} \} \rangle$ is a Σ-algebra that we denote by $\boldsymbol{A}^\circ$.[1] Clearly $\boldsymbol{A}^\circ$ is order-sorted; it is called the *order-sorted transform* of $\boldsymbol{A}$. For example, by the above construction the Σ_{pt}-algebra $\boldsymbol{PT}^\circ$ is order-sorted.

Definition 2.8 (dynamic type, virtual type)**.** An element of A_T is said to be of *dynamic type* T. An element of A°_T is said to be of *virtual type* T; note that each element of dynamic type S is of virtual type T for every $T \geq S$. □

[1]In our earlier work [23], we used the notation $\widehat{A}_T$ for what is here defined as A°_T. The reason for the change of notation is that the corresponding operation on algebras is different from our earlier work. In [23], $\widehat{\boldsymbol{A}}$ is not a Σ-algebra, but a $\widehat{\Sigma}$-algebra, i.e., an algebra whose signature is discrete in that $S \leq T$ only if $S = T$. However, when forming an order-sorted algebra, we do not wish to eliminate subtyping from the signature.

$$\begin{aligned}
\mathtt{0}^{\boldsymbol{PT}} &= 0\\
\mathtt{1}^{\boldsymbol{PT}} &= 1\\
\mathtt{succ}^{\boldsymbol{PT}}(a) &= a+1\\
\mathtt{pred}^{\boldsymbol{PT}}(a) &= a-1\\
\mathtt{add}^{\boldsymbol{PT}}(a,b) &= a+b\\
\mathtt{sub}^{\boldsymbol{PT}}(a,b) &= a-b\\[1em]
\mathtt{PtOrigin}^{\boldsymbol{PT}} &= \langle 0,0\rangle\\
\mathtt{CPtOrigin}^{\boldsymbol{PT}} &= \langle 0,0,0\rangle\\
\mathtt{x}^{\boldsymbol{PT}}(\langle x,y\rangle) &= x\\
\mathtt{x}^{\boldsymbol{PT}}(\langle x,y,c\rangle) &= x\\
\mathtt{y}^{\boldsymbol{PT}}(\langle x,y\rangle) &= y\\
\mathtt{y}^{\boldsymbol{PT}}(\langle x,y,c\rangle) &= y\\
\mathtt{color}^{\boldsymbol{PT}}(\langle x,y,c\rangle) &= c\\
\mathtt{xdiff}^{\boldsymbol{PT}}(\langle x_1,y_1\rangle,\langle x_2,y_2\rangle) &= x_1-x_2\\
\mathtt{xdiff}^{\boldsymbol{PT}}(\langle x_1,y_1,c\rangle,\langle x_2,y_2\rangle) &= x_1-x_2\\
\mathtt{xdiff}^{\boldsymbol{PT}}(\langle x_1,y_1\rangle,\langle x_2,y_2,c\rangle) &= x_1-x_2\\
\mathtt{xdiff}^{\boldsymbol{PT}}(\langle x_1,y_1,c\rangle,\langle x_2,y_2,c\rangle) &= x_1-x_2\\[1em]
\mathtt{movex}^{\boldsymbol{PT}}(\langle x,y\rangle,a) &= \langle x+a,y\rangle\\
\mathtt{movex}^{\boldsymbol{PT}}(\langle x,y,c\rangle,a) &= \langle x+a,y,x\rangle\\
\mathtt{movey}^{\boldsymbol{PT}}(\langle x,y\rangle,a) &= \langle x,y+a\rangle\\
\mathtt{movey}^{\boldsymbol{PT}}(\langle x,y,c\rangle,a) &= \langle x,y+a,y\rangle\\
\mathtt{changec}^{\boldsymbol{PT}}(\langle x,y,c\rangle,d) &= \langle x,y,d\rangle
\end{aligned}$$

FIGURE 2. Definitions of $\boldsymbol{PT}$'s operation interpretations.

An algebra $\boldsymbol{A}$ is *nominal* if, for each $n \in \mathbb{N}$, each $g \in \mathrm{Op}_n$, each admissible type $T_1, \dots, T_n \to S$ of g, and all $a_1, \dots, a_n \in A_{T_1} \times \cdots \times A_{T_n}$, $g^{\boldsymbol{A}}(a_1, \dots, a_n)$ is of dynamic type S, i.e., $g^{\boldsymbol{A}}(a_1, \dots, a_n) \in A_S$.

If $\boldsymbol{A}$ is order-sorted, then every element of virtual type T is also of dynamic type T; in fact this property characterizes order-sorted algebras. Thus every order-sorted Σ-algebra is nominal. However, there are algebras that are nominal but not order-sorted; $\boldsymbol{PT}$ is one example. It is also easy to construct algebras that are not nominal (and hence also not order-sorted); for example, if one changes the algebra $\boldsymbol{PT}$, by making `PtOrigin` return $\langle 0,0,0\rangle$, which is an element of the carrier set of `CPt`, then this modified algebra would not be nominal.

Definition 2.9 (dynamic and virtual environments)**.** Let $\boldsymbol{A}$ be a Σ-algebra, and also let $H = \langle x_1{:}\,T_1, \dots, x_n{:}\,T_n\rangle$ be a type context. A *dynamic* (*resp. virtual*) H*-environment in* $\boldsymbol{A}$ is a sequence $\bar{a} = \langle a_1, \dots, a_n\rangle$ of elements of $\boldsymbol{A}$ such that a_i is of dynamic (resp. virtual) type T_i, for each $i \le n$.[2] □

[2] In our earlier work [23], we called dynamic environments "nominal."

Note that every dynamic H-environment is virtual and that the two kinds of environments coincide just when the algebra is order-sorted. Although virtual environments are more general, we will be dealing mainly with dynamic environments, and hence the latter is taken to be the default kind of environment. Thus when we say "H-environment" without qualification we mean a dynamic H-environment.

A dynamic or virtual H-environment can also be regarded as a finite function from the domain of H to the corresponding elements in the environment. For example, if $H = \langle x_1{:}\,T_1, \dots, x_n{:}\,T_n \rangle$ and $\bar{a} = \langle a_1, \dots, a_n \rangle$ is a dynamic or virtual H-environment, then $\bar{a}(x_i) = a_i$.

For each type context H we define the *H-nominal Σ-term algebra*

$$\mathbf{Tm}_\Sigma(H) = \big\langle \mathrm{Tm}_\Sigma(H), \{\, g^{\mathbf{Tm}_\Sigma(H)} : g \in \mathrm{Op} \,\} \big\rangle$$

as follows. For each $T \in \mathrm{TYPE}$, $\mathbf{Tm}_\Sigma(H)_T$ is the set of all well H-typed terms of nominal H-type T. For each $g \in \mathrm{Op}_n$, each admissible type domain $(T_1, \dots, T_n)$ of g, and each sequence of well H-typed terms $t_1{:}\,T_1, \dots, t_n{:}\,T_n$, $g^{\mathbf{Tm}_\Sigma(H)}(t_1, \dots, t_n) = g(t_1, \dots, t_n)$. Thus the typing rule (*op-call*) guarantees that $\mathbf{Tm}_\Sigma(H)$ is a nominal Σ-algebra. By definition the elements of $\mathbf{Tm}_\Sigma(H)$ of dynamic type T are exactly the terms of nominal type T. So the type context H is itself a dynamic H-environment of $\mathbf{Tm}_\Sigma(H)$, called the *canonical dynamic environment*. Again if, H is irrelevant or understood from context, we refer simply to the *nominal Σ-term algebra*, or even more simply the *Σ-term algebra*.

By the subsumption lemma the elements of $\mathbf{Tm}_\Sigma(H)$ of virtual type S are exactly the terms of subsumptive type S. Thus $\mathbf{Tm}_\Sigma(H)^\circ$ is called the *subsumptive H-term Σ-algebra*.

If $H = \langle x_1{:}\,T_1, \dots, x_n{:}\,T_n \rangle$ is a type context, $\bar{a} = \langle a_1, \dots, a_n \rangle$ is an H-environment, and $t[x_1{:}\,T_1, \dots, x_n{:}\,T_n]{:}\,S$ is a well H-typed term, we write $t^{\boldsymbol{A}}[\bar{a}]$ for the usual meaning of t in the environment $\bar{a}$ of $\boldsymbol{A}$, i.e., the value it takes in $\boldsymbol{A}$ when each variable x_i takes the value a_i. It is inductively defined by $x_i^{\boldsymbol{A}}[\bar{a}] = a_i$ and $g(t_1, \dots, t_k)^{\boldsymbol{A}}[\bar{a}] = g^{\boldsymbol{A}}(t_1^{\boldsymbol{A}}[\bar{a}], \dots, t_k^{\boldsymbol{A}}[\bar{a}])$.

Definition 2.10 (satisfaction and validity of typing expressions)**.** Let $\boldsymbol{A}$ be a Σ-algebra and $H = \langle x_1{:}\,T_1, \dots, x_n{:}\,T_n \rangle$ a type context. A typing expression $t{:}\,T$ is *satisfied* in a dynamic or virtual H-environment $\bar{a}$ of $\boldsymbol{A}$ if $t^{\boldsymbol{A}}[\bar{a}]$ is defined (i.e., t is well H-typed) and $t^{\boldsymbol{A}}[\bar{a}] \in A_T$. It is *$H$-valid* in $\boldsymbol{A}$ if it is satisfied in every dynamic H-environment of $\boldsymbol{A}$. In this case we say that $\boldsymbol{A}$ is an *H-model* of $t{:}\,T$. □

Theorem 2.11 (Type Soundness and Completeness Theorem)**.** *Let Σ be a hidden signature with subtyping and $\boldsymbol{A}$ a Σ-algebra.*

(i) *$\boldsymbol{A}$ is nominal if and only if, for every type context H, every type T, and every well typed term t of nominal H-type T, $\boldsymbol{A}$ is an H-model of $t{:}\,T$.*

(ii) *$\boldsymbol{A}$ is order-sorted if and only if, for every type context H, every type S, and every well typed term t of subsumptive H-type S, $\boldsymbol{A}$ is an H-model of $t{:}\,S$.*

Proof. In both parts the implications from left to right are proved by induction on the length of type deductions. The implications in the opposite direction are immediate. □

2.4. **Equations and Specifications.** An *equation over Σ* is an ordered triple (H, t, t') such that H is a type context and t and t' are well H-typed terms with the same nominal

H-type, which we refer to as the *nominal type*, or simply the *type*, of (H, t, t').[3] We normally write equations in the form $\forall H(t = t')$. If the type context is irrelevant or understood from context we write $t = t'$ in place of $\forall H(t = t')$. Since well typed terms have a unique nominal type in a given signature and type context, the nominal type of each side of the equation can be derived from the typing rules. By the subsumption lemma, if t and t' have the same nominal type then they have the same subsumptive types, and we take these to be the *subsumptive types* of the equation $\forall H(t = t')$.

Definition 2.12 (equational specification)**.**

(i) A *visible equational specification* is a pair $\langle \Psi, E_{\mathrm{VIS}} \rangle$ where Ψ is a visible data signature and E_{VIS} is a set of equations over Ψ.

(ii) A *hidden equational specification* is a pair $\langle \Sigma, E \rangle$ where Σ is a hidden signature over the visible data signature Ψ and and E is the disjoint union of a set E_{VIS} of equations over Ψ and a set E_{HID} of equations over Σ such that each equation of E_{HID} is of visible type but is not a Ψ-equation, i.e., it contains at least one hidden operation symbol. We refer to $\langle \Sigma, E \rangle$ a *hidden specification* over the visible specification $\langle \Psi, E_{\mathrm{VIS}} \rangle$, which is called the *visible part of* $\langle \Sigma, E \rangle$. □

Example 2.13. For example, let E_{pt} consist of the equations over Σ_{pt} presented in Figure 3; then $\langle \Sigma_{pt}, E_{pt} \rangle$ is a hidden equational specification whose visible part is given by the first group of equations in the figure. □

Example 2.14. The equations in E_{pt} do not require that the operations `movex` and `movey` preserve the color of a `CPt` argument. However, this can be ensured by adding to E_{pt} the first two additional equations given in Figure 4. This gives the set of equations E_{cpt}. □

The restriction to equations of visible type in a hidden equational specification is nonstandard but is natural from the perspective of operational semantics. That is, in operational terms, one views the equations as specifying the output of programs, which indirectly determine the behavior of the hidden data objects the programs manipulate. One can imagine that the program executes by taking visible data as input, forming hidden data objects by means of the constructors, and finally outputting visible data by means of the attributes. To reflect the fact that one can only observe the visible outputs of program executions, we restrict the underlying equational logic so that equality comparisons between hidden data is not allowed. The only notion of equality appropriate from this perspective is behavioral equivalence.

The restriction that only equations of visible type are allowed does not guarantee that an equational specification is consistent, and it does not guarantee that it is a conservative extension of its visible part. A stronger condition, however, can be used to guarantee consistency. One such condition is embodied in Goguen and Malcolm's notion of a *local equation* [15]; they prove that if the hidden equations of a specification are local then the specification is consistent. It can also be shown that it is a conservative extension of its visible part.

[3]In the standard formalization of multi-sorted equational logic each variable symbol is assumed to have an a priori type that never varies, and thus a type context is completely determined by its domain. In the standard formalization the equation (H, t, t') would be denoted by $\forall X(t = t')$ where X is the domain of H.

$$
\begin{array}{ll}
\multicolumn{2}{l}{\text{for all } a : \texttt{int},\, b : \texttt{int},\, c : \texttt{int},\, i : \texttt{int},\, p : \texttt{Pt},\, p_1 : \texttt{Pt},\, p_2 : \texttt{Pt},} \\
& \texttt{succ}(\texttt{0}) = \texttt{1} \\
& \texttt{pred}(\texttt{1}) = \texttt{0} \\
& \texttt{pred}(\texttt{succ}(a)) = a \\
& \texttt{succ}(\texttt{pred}(a)) = a \\
& \texttt{add}(\texttt{0}, a) = a \\
& \texttt{add}(\texttt{succ}(a), b) = \texttt{succ}(\texttt{add}(a, b)) \\
& \texttt{add}(\texttt{pred}(a), b) = \texttt{pred}(\texttt{add}(a, b)) \\
& \texttt{sub}(a, \texttt{0}) = a \\
& \texttt{sub}(a, \texttt{succ}(b)) = \texttt{pred}(\texttt{sub}(a, b)) \\
& \texttt{sub}(a, \texttt{pred}(b)) = \texttt{succ}(\texttt{sub}(a, b)) \\
\\
(\textit{x-PtOrigin}) & \texttt{x}(\texttt{PtOrigin}) = \texttt{0} \\
(\textit{y-PtOrigin}) & \texttt{y}(\texttt{PtOrigin}) = \texttt{0} \\
(\textit{x-CPtOrigin}) & \texttt{x}(\texttt{CPtOrigin}) = \texttt{0} \\
(\textit{y-CPtOrigin}) & \texttt{y}(\texttt{CPtOrigin}) = \texttt{0} \\
(\textit{color-CPtOrigin}) & \texttt{color}(\texttt{CPtOrigin}) = \texttt{0} \\
\\
(\textit{x-movex}) & \texttt{x}(\texttt{movex}(p, i)) = \texttt{add}(\texttt{x}(p), i) \\
(\textit{y-movey}) & \texttt{y}(\texttt{movey}(p, i)) = \texttt{add}(\texttt{y}(p), i) \\
(\textit{y-movex}) & \texttt{y}(\texttt{movex}(p, i)) = \texttt{y}(p) \\
(\textit{x-movey}) & \texttt{x}(\texttt{movey}(p, i)) = \texttt{x}(p) \\
(\textit{xdiff}) & \texttt{xdiff}(p_1, p_2) = \texttt{sub}(\texttt{x}(p_1), \texttt{x}(p_2))
\end{array}
$$

FIGURE 3. The set of equations E_{pt}. Each of these is understood to be an H-equation where H is the type subcontext obtained from the one given at top of the figure by restricting to the variables actually occurring in the equation. For example, the type context of the two equations is empty, and the type context of the sixth equation is $\{a{:}\,\texttt{int}, b{:}\,\texttt{int}\}$. We label the equations in the non-visible part for later use.

$$
\begin{array}{ll}
\multicolumn{2}{c}{\text{for all } i : \texttt{int},\, cp : \texttt{CPt},} \\
(\textit{color-movex}) & \texttt{color}(\texttt{movex}(cp, i)) = \texttt{color}(cp) \\
(\textit{color-movey}) & \texttt{color}(\texttt{movey}(cp, i)) = \texttt{color}(cp)
\end{array}
$$

FIGURE 4. Equations in E_{cpt} added to those in E_{pt}. The type context of each equation is to be understood to be the one given at top of the figure.

2.5. Model Theory.

We now relate equations and their algebraic models.

Definition 2.15 (satisfaction and validity of equations)**.** Let $\boldsymbol{A}$ be a Σ-algebra. An equation $\forall H(t = t')$ is *satisfied* in a dynamic or virtual H-environment $\bar{a}$ of $\boldsymbol{A}$ if $t^{\boldsymbol{A}}[\bar{a}] = t'^{\boldsymbol{A}}[\bar{a}]$. It is *valid* in $\boldsymbol{A}$ if it is satisfied in every dynamic H-environment of $\boldsymbol{A}$; we say that $\boldsymbol{A}$ is a *model* of $\forall H(t = t')$ in this event. □

The virtual environments of a Σ-algebra $\boldsymbol{A}$ are exactly the dynamic environments of its order-sorted transform $\boldsymbol{A}^\circ$. Thus an H-equation is satisfied in every virtual H-environment

$$\begin{array}{lcl}
\mathtt{movex}^{\boldsymbol{CPT}}(\langle x,y\rangle,a) & = & \langle x+a,y\rangle \\
\mathtt{movex}^{\boldsymbol{CPT}}(\langle x,y,c\rangle,a) & = & \langle x+a,y,c\rangle \\
\mathtt{movey}^{\boldsymbol{CPT}}(\langle x,y\rangle,a) & = & \langle x,y+a\rangle \\
\mathtt{movey}^{\boldsymbol{CPT}}(\langle x,y,c\rangle,a) & = & \langle x,y+a,c\rangle
\end{array}$$

for all other operations, $g \notin \{\mathtt{movex}, \mathtt{movey}\}$, $g^{\boldsymbol{CPT}} = g^{\boldsymbol{PT}}$.

FIGURE 5. Definitions of $\boldsymbol{CPT}$'s operation interpretations.

of $\boldsymbol{A}$ if it is satisfied by every dynamic H-environment of $\boldsymbol{A}^\circ$. Consequently the notion of the validity of an equation based on virtual environments is subsumed under that of validity based on dynamic environments only.

For example, $\boldsymbol{PT}$ is a model of the $\langle p : \mathtt{Pt}, i : \mathtt{int}\rangle$-equation, $\mathtt{x}(\mathtt{movex}(p,i)) = \mathtt{add}(\mathtt{x}(p),i)$, since it is true in every $\langle p : \mathtt{Pt}, i : \mathtt{int}\rangle$-environment, in $\boldsymbol{PT}$, including non-dynamic ones such as $\langle\langle 18,47,342\rangle,54\rangle$. Thus this equation is valid in both $\boldsymbol{PT}$ and in $\boldsymbol{PT}^\circ$. However, equations are not, in general, true in virtual environments even when they hold in all dynamic environments. That is, an algebra may be a model of an equation without its order-sorted transform being a model. For example, while $\boldsymbol{PT}$ is a model of the $\langle p : \mathtt{Pt}\rangle$-equation $\mathtt{movex}(p,\mathtt{0}) = p$, $\boldsymbol{PT}^\circ$ is not a model of this equation, due to the (intentionally strange) definition of $\mathtt{movex}$. To see this, consider the non-dynamic environment $\langle\langle 18,47,342\rangle\rangle$, which binds the variable $p : \mathtt{Pt}$ to the colored point $\langle 18,47,342\rangle$; in this environment, $\mathtt{movex}(p,\mathtt{0})^{\boldsymbol{PT}}[\langle 18,47,342\rangle] = \langle 18,47,18\rangle$, which is not the same as $p^{\boldsymbol{PT}}[\langle 18,47,342\rangle] = \langle 18,47,342\rangle$. On the other hand, both the algebra $\boldsymbol{CPT}$, whose carrier sets are the same as those in $\boldsymbol{PT}$ and whose operations are described in Figure 5, and its order-sorted transform are models of this equation.

Definition 2.16 (models; nominal and order-sorted models)**.** Let $\langle \Sigma, E\rangle$ be an equational specification. The class of models of $\langle \Sigma, E\rangle$, i.e., the class of all Σ-algebras that are models of every equation in E is denoted by $\mathsf{Mod}(E)$. The class of all nominal (resp. order-sorted) models of E, i.e., nominal (resp. order-sorted) algebras that are models of $\langle \Sigma, E\rangle$, is denoted by $\mathsf{Mod}_{\mathrm{NOM}}(E)$ (resp. $\mathsf{Mod}_{\mathrm{ORD}}(E)$). □

The class $\mathsf{Mod}(E)$ of all models of $\langle \Sigma, E\rangle$ is called the *loose semantics* of $\langle \Sigma, E\rangle$.

We note that, in general, $\mathsf{Mod}_{\mathrm{ORD}}(E) \subseteq \mathsf{Mod}_{\mathrm{NOM}}(E)$. For example, $\boldsymbol{PT}$ is a model of $\langle \Sigma_{pt}, E_{pt}\rangle$; hence $\boldsymbol{PT}$ is in $\mathsf{Mod}(E_{pt})$. Since $\boldsymbol{PT}$ is both a nominal Σ_{pt}-algebra and a model of each equation in E_{pt}, $\boldsymbol{PT}$ is a nominal model of $\langle \Sigma_{pt}, E_{pt}\rangle$, i.e., $\boldsymbol{PT} \in \mathsf{Mod}_{\mathrm{NOM}}(E_{pt})$. But $\boldsymbol{PT}$ is not a model of the additional equations in E_{cpt}; indeed both of the equations in Figure 4 are invalid in $\boldsymbol{PT}$. Therefore $\boldsymbol{PT}$ is not a model of $\langle \Sigma_{pt}, E_{cpt}\rangle$. On the other hand, $\boldsymbol{CPT}$ is a model of $\langle \Sigma_{pt}, E_{cpt}\rangle$; since $\boldsymbol{CPT}$ is also nominal it is also a nominal model of $\langle \Sigma_{pt}, E_{cpt}\rangle$; similarly, $\boldsymbol{CPT}^\circ$ is also a nominal model of $\langle \Sigma_{pt}, E_{cpt}\rangle$.

We could have considered also a notion of *virtual model*, where the equations of E are required to be satisfied in all virtual environments as well as all dynamic ones. But it is easy to see that an algebra is a virtual model in this sense if and only if its order-sorted transform is a model in the sense of Def. 2.16 so the notion is not needed; cf. the remarks following Def. 2.15.

Definition 2.17 (nominal and order-sorted consequence)**.** An H-equation $\forall H(t = t')$ over Σ is a *nominal* (*resp. order-sorted*) *consequence* of $\langle \Sigma, E\rangle$, in symbols $E \models_{\text{NOM}} \forall H(t = t')$ (resp. $E \models_{\text{ORD}} \forall H(t = t')$), if it is valid in every nominal (resp. order-sorted) model of $\langle \Sigma, E\rangle$. □

In the theory of specifications additional extra-equational conditions may be imposed in order to restrict the class of models. One natural restriction is to require all models to be term-generated; that is one could require that every element of the carrier of a model can be obtained by applying operations, starting from constants. If in addition the model is required to be initial in the category of all models, then the class of restricted models would contain only one algebra up to isomorphism, making it "categorical" [27, 29].

Because OO programming languages contain "abstract" types that are not term-generated, we do not restrict our study to term-generated models. However, a programming language will have a fixed set of built-in types, which correspond to our notion of visible types, and the class of reducts of the restricted models to these types, the *visible reducts*, will be categorical; this implies that the class of restricted models of the visible part of a hidden specification will be categorical. We refer to the class of restricted models of a hidden equational specification as VIS-*categorical* when we impose this restriction that the class of reducts of the visible part of the signature be term-generated and initial. Without any real loss of generality we will assume that, if the class of restricted models is VIS-categorical, then the visible reduct is fixed; that is, for every pair of restricted models, the visible reducts are the same. This so-called *fixed data assumption* is incorporated into the definition of a hidden specification in [16, 15], but is relaxed in some of the subsequent work on hidden algebras, e.g., [6, 34].

2.6. **Proof Theory.** The formal proof system given in Figure 6 is essentially the one given in Ehrig and Mahr's book [11, p. 111] appropriately modified to handle subtyping. In their book every variable has a preassigned type that is fixed at the beginning and consequently sets of variables play the role of type contexts; this accounts for most of the difference between our equational logic and that of Ehrig and Mahr.

The following two type-context-modifying rules, which are often taken as part of the formalism, are derivable from (*invar*) and will be used in the sequel.

$$\text{(abstr)} \quad \frac{E \vdash \forall H'(t = t')}{E \vdash \forall H(t = t')}, \quad \text{where } H' \subseteq H$$

$$\text{(concr)} \quad \frac{E \vdash \forall H'(t = t')}{E \vdash \forall H(t = t')}, \quad \begin{array}{l}\text{where } H = H' \setminus \{y{:}T\},\ y \text{ does not occur in } t \text{ or } t',\\ \text{and } T \text{ is nominally nonvacuous in } H'\end{array}$$

A type T is said to be *nominally* (*resp. subsumptively*) *vacuous* in a type context H if $\mathbf{Tm}_\Sigma(H)_T = \emptyset$ (resp. $\mathbf{Tm}_\Sigma(H)^\circ_T = \emptyset$).

Definition 2.18. Let $\langle \Sigma, E\rangle$ be an equational specification. An equation over Σ, $\forall H(t = t')$, is *nominally* (*resp. subsumptively*) *equationally provable from* E, in symbols $E \vdash_{\text{NOM}} \forall H(t = t')$ (resp. $E \vdash_{\text{SUB}} \forall H(t = t')$), if there is a finite sequence of equations over Σ terminating in $\forall H(t = t')$ such that each equation is either in E or is directly derivable from preceding equations in the sequence by one of the inference rules in Figure 6 using the nominal (resp. subsumptive) forms of (*cong*) and (*invar*). □

$(\textit{axiom})\qquad E \vdash \forall H(t = t'), \quad \text{where } \forall H(t = t') \in E$

$(\textit{refl})\qquad E \vdash \forall H(t = t)$

$$(\textit{sym})\qquad \frac{E \vdash \forall H(t = t')}{E \vdash \forall H(t' = t)}$$

$$(\textit{trans})\qquad \frac{E \vdash \forall H(t = t'),\ E \vdash \forall H(t' = t'')}{E \vdash \forall H(t = t'')}$$

$$(\textit{cong})_{\text{NOM}}\qquad \frac{E \vdash \forall H(t_1 = t_1'), \ldots, E \vdash \forall H(t_n = t_n')}{E \vdash \forall H(g(t_1, \ldots, t_n) = g(t_1', \ldots, t_n'))},$$

where $\forall H(t_1 = t_1'), \ldots, \forall H(t_n = t_n')$ are of **nominal** types $T_1, \ldots, T_n$, and $T_1, \ldots, T_n$ is an admissible domain of g

$$(\textit{cong})_{\text{SUB}}\qquad \frac{E \vdash \forall H(t_1 = t_1'), \ldots, E \vdash \forall H(t_n = t_n')}{E \vdash \forall H(g(t_1, \ldots, t_n) = g(t_1', \ldots, t_n'))},$$

where $\forall H(t_1 = t_1'), \ldots, \forall H(t_n = t_n')$ are of **subsumptive** types $T_1, \ldots, T_n$, and $T_1, \ldots, T_n$ is an admissible domain of g

$$(\textit{invar})_{\text{NOM}}\qquad \frac{E \vdash \forall H'(t[x_1\colon T_1, \ldots, x_n\colon T_n] = t'[x_1\colon T_1, \ldots, x_n\colon T_n])}{E \vdash \forall H(t[t_1, \ldots, t_n] = t'[t_1, \ldots, t_n])},$$

where $t_1, \ldots, t_n$ are of **nominal** H-types $T_1, \ldots, T_n$

$$(\textit{invar})_{\text{SUB}}\qquad \frac{E \vdash \forall H'(t[x_1\colon T_1, \ldots, x_n\colon T_n] = t'[x_1\colon T_1, \ldots, x_n\colon T_n])}{E \vdash \forall H(t[t_1, \ldots, t_n] = t'[t_1, \ldots, t_n])},$$

where $t_1, \ldots, t_n$ are of **subsumptive** H-types $T_1, \ldots, T_n$.

FIGURE 6. Axioms and inference rules for nominal and subsumptive equational logic with subtyping over a set of equations E. The differences between the nominal and subsumptive forms of (*cong*) and (*invar*) are highlighted in boldface.

For example, we have $E_{pt} \vdash_{\text{NOM}} \texttt{x}(\texttt{movex}(\texttt{CPtOrigin}, \texttt{1})) = \texttt{1}$.

Theorem 2.19 (Completeness and Soundness Theorem for Equational Logic with Subtypes)**.** *Let $\langle \Sigma, E\rangle$ be an equational specification, and let $\forall H(t = t')$ be an equation over Σ.*

(i) $E \vDash_{\text{NOM}} \forall H(t = t')$ *iff* $E \vdash_{\text{NOM}} \forall H(t = t')$.
(ii) $E \vDash_{\text{ORD}} \forall H(t = t')$ *iff* $E \vdash_{\text{SUB}} \forall H(t = t')$.

Proof. (i) $\Leftarrow$. Assume $E \vdash_{\text{NOM}} \forall H(s = s')$ and let $\boldsymbol{A} \in \mathsf{Mod}_{\text{NOM}}(E)$ and let $\bar{a}$ be a dynamic H-environment of $\boldsymbol{A}$. We prove $\forall H(s = s')$ is satisfied in $\bar{a}$ by induction on the length of the derivation of $\forall H(s = s')$. If $\forall H(s = s') \in E$ this follows from the definition of $\mathsf{Mod}_{\text{NOM}}(E)$.

Suppose $\forall H(s = s')$ is of the form $\forall H(g(t_1, \dots, t_n) = g(t'_1, \dots, t'_n))$ and is obtained from $\forall H(t_1 = t'_1), \dots, \forall H(t_n = t'_n)$ of nominal type $T_1, \dots, T_n$, which is an admissible domain of g, by application of $(cong)_{\text{NOM}}$. By the induction hypothesis $t_i^{\boldsymbol{A}}[\bar{a}] = t_i'^{\boldsymbol{A}}[\bar{a}]$ for $i \le n$. Thus $s^{\boldsymbol{A}}[\bar{a}] = g^{\boldsymbol{A}}(t_1^{\boldsymbol{A}}[\bar{a}], \dots, t_n^{\boldsymbol{A}}[\bar{a}]) = g^{\boldsymbol{A}}(t_1'^{\boldsymbol{A}}[\bar{a}], \dots, t_n'^{\boldsymbol{A}}[\bar{a}]) = s'^{\boldsymbol{A}}[\bar{a}]$.

Suppose now that $\forall H(s = s')$ is of the form $\forall H(t[t_1, \dots, t_n] = t'[t_1, \dots, t_n])$ and is obtained from $\forall H'(t[x_1{:}\, T_1, \dots, x_n{:}\, T_n] = t'[x_1{:}\, T_1, \dots, x_n{:}\, T_n])$ by an application of $(invar)_{\text{NOM}}$, where $H' = \langle x_1{:}\, T_1, \dots, x_m{:}\, T_n \rangle$ and $t_1, \dots, t_n$ are nominal H-types $T_1, \dots, T_n$. Let $\bar{b} = \langle t_1^{\boldsymbol{A}}[\bar{a}], \dots, t_n^{\boldsymbol{A}}[\bar{a}] \rangle$ and $\bar{b}' = \langle t_1'^{\boldsymbol{A}}[\bar{a}], \dots, t_n'^{\boldsymbol{A}}[\bar{a}] \rangle$. Then $\bar{b}$ and $\bar{b}'$ are dynamic H'-environments, since $\boldsymbol{A}$ is nominal, and $\bar{b} = \bar{b}'$ by the induction hypothesis. So $s^{\boldsymbol{A}}[\bar{a}] = t^{\boldsymbol{A}}[\bar{b}] = t^{\boldsymbol{A}}[\bar{b}'] = s'^{\boldsymbol{A}}[\bar{a}]$.

The verifications that $\forall H(t = t')$ is satisfied in $\bar{a}$ when it is obtained using (*refl*), (*sym*), or (*trans*) are similar.

(ii) $\Leftarrow$. Assume now that $E \vdash_{\text{SUB}} \forall H(s = s')$ and let $\boldsymbol{A} \in \mathsf{Mod}_{\text{ORD}}(E)$ and let $\bar{a}$ be a dynamic H-environment of $\boldsymbol{A}$. Then the proof is similar to that above with the "nominal" concepts replaced by the corresponding "subsumptive" ones. The reason the $\bar{b}$ and $\bar{b}'$ obtained as above are dynamic H'-environments, is because in this case $\boldsymbol{A}$ is order-sorted.

(i) $\Rightarrow$. We prove the contrapositive. Assume $E \nvdash_{\text{NOM}} \forall H(s = s')$. We construct a nominal model of E and an H-environment in which $\forall H(s = s')$ is not valid. Let $\Theta = \langle\, \Theta_T : T \in \text{TYPE} \,\rangle$ be the sorted binary relation on $\text{Tm}_\Sigma(H) = \langle\, \text{Tm}_\Sigma(H)_T : T \in \text{TYPE} \,\rangle$ defined by $\Theta_T = \{\, \langle t, t' \rangle : E \vdash_{\text{NOM}} \forall H(t = t') \,\}$. Θ is a equivalence relation by the rules (*refl*), (*sym*), and (*trans*). To see that Θ is a congruence relation, let $g \in \text{Op}_n$ with admissible type $T_1, \dots, T_n \to T$, and suppose $t_1, \dots, t_n$ and $t'_1, \dots, t'_n$ are two sequences of well H-typed terms of nominal type $T_1, \dots, T_n$ such that $\langle t_1, t'_1 \rangle \in \Theta_{T_1}, \dots, \langle t_n, t'_n \rangle \in \Theta_{T_n}$. Then by definition of Θ, $E \vdash_{\text{NOM}} \forall H(t_1 = t'_1), \dots, E \vdash_{\text{NOM}} \forall H(t_n = t'_n)$. Thus $E \vdash_{\text{NOM}} \forall H(g(t_1, \dots, t_n) = g(t'_1, \dots, t'_n))$ by $(cong)_{\text{NOM}}$. So $\langle g(t_1, \dots, t_n), g(t'_1, \dots, t'_n) \rangle \in \Theta_T$.

The quotient algebra $\mathbf{Tm}_\Sigma(H)/\Theta$ is nominal since $\mathbf{Tm}_\Sigma(H)$ is nominal. Let

$$\forall H'(t[y_1{:}\, S_1, \dots, y_n{:}\, S_n] = t'[y_1{:}\, S_1, \dots, y_n{:}\, S_n]) \in E,$$

where $H' = \langle y_1{:}\, S_1, \dots, y_n{:}\, S_n \rangle$, and let the equivalence class representatives $t_1/\Theta, \dots, t_n/\Theta$ be of nominal types $S_1, \dots, S_n$ in $\mathbf{Tm}_\Sigma(H)/\Theta$. Then the terms $t_1, \dots, t_n$ are well H-typed and of nominal H-type $S_1, \dots, S_n$ since $\mathbf{Tm}_\Sigma(H)$ is nominal. So, by $(invar)_{\text{NOM}}$,

$$\begin{aligned} t^{\mathbf{Tm}_\Sigma(H)/\Theta}[t_1/\Theta, \dots, t'_n/\Theta] &= t[t_1{:}\, T_1, \dots, t_n{:}\, T_n]/\Theta \\ &= t'[t_1{:}\, T_1, \dots, t_n{:}\, T_n]/\Theta \\ &= t'^{\mathbf{Tm}_\Sigma(H)/\Theta}[t_1/\Theta, \dots, t_n/\Theta]. \end{aligned}$$

So $\forall H'(t[t_1{:}\, T_1, \dots, t_n{:}\, T_n] = t'[t_1{:}\, T_1, \dots, t_n{:}\, T_n]) \in E$ is valid in $\mathbf{Tm}_\Sigma(H)/\Theta$ and hence $\mathbf{Tm}_\Sigma(H)/\Theta$ is a nominal model of E.

Let $H = \langle x_1{:}\, T_1, \dots, x_n{:}\, T_n \rangle$ and let $H/\Theta = \langle x_1/\Theta, \dots, x_n/\Theta \rangle$ be the canonical H-environment of $\mathbf{Tm}_\Sigma(H)/\Theta$.

$$\begin{aligned} s^{\mathbf{Tm}_\Sigma(H)/\Theta}[x_1/\Theta, \dots, x_n/\Theta] &= s[x_1{:}\, T_1, \dots, x_n{:}\, T_n]/\Theta \\ &\neq s'[x_1{:}\, T_1, \dots, x_n{:}\, T_n]/\Theta \\ &= s'^{\mathbf{Tm}_\Sigma(H)/\Theta}[x_1/\Theta, \dots, x_n/\Theta], \end{aligned}$$

where the inequality holds by assumption. So $E \not\models_{\text{NOM}} \forall H(s = s')$.

(ii) $\Rightarrow$. Assume $E \nvdash_{\text{SUB}} \forall H(s = s')$. Let $\Theta^\circ = \langle\, \Theta^\circ_T : T \in \text{TYPE}\,\rangle$ be the sorted binary relation on $\text{Tm}_\Sigma(H)^\circ = \langle\, \text{Tm}_\Sigma(H)^\circ_T : T \in \text{TYPE}\,\rangle$ defined by $\Theta^\circ_T = \{\, \langle t, t'\rangle : E \vdash_{\text{SUB}} \forall H(t = t')\,\}$. A verification that $\forall H(s = s')$ is not valid in $\mathbf{Tm}_\Sigma(H)^\circ/\Theta^\circ$ is obtained from the one above by making the obvious modifications. □

The completeness and soundness theorem shows that the formal methods of equational logic alone are adequate for characterizing both the order-sorted and the nominal models of an equational specification (corresponding respectively to whether or not (*subsumption*) is admitted as a typing rule). It is well known, however, that they are not adequate for those models in which identity coincides with behavioral equivalence and hence inadequate for characterizing the correctly behaviorally subtyped models of the specification (see Section 5). A new equational logic is required with an expanded notion of consequence [3, 7, 33, 34, 35]. These ideas are developed in the next section.

Before turning to this however, we give an alternative formalization of equational consequence that will prove useful in developing the proof theory of behavioral equivalence.

Definition 2.20. Let E be a set of equations and let $\breve{E} = \{\forall H(t = t') : \forall H(t' = t) \in E\,\}$. Let $H = \langle x_1{:}\,T_1, \dots, x_n{:}\,T_n\rangle$ be a fixed but arbitrary type context. Define $\equiv_E(H) = \langle\, \equiv_E(H)_T\colon T \in \text{TYPE}\,\rangle$ to be the TYPE-sorted binary relation on $\text{Tm}_\Sigma(H)$ such that, for all H-terms t and t' of nominal type T, $t \equiv_E(H)_T\ t'$ iff there is a Σ-term $r[z{:}\,S, x_1{:}\,T_1, \dots, x_n{:}\,T_n]$ of type T, with exactly one occurrence of a distinguished variable z, and an equation $\forall K(e[y_1{:}\,S_1, \dots, y_m{:}\,S_m] = e'[y_1{:}\,S_1, \dots, y_m{:}\,S_m])$ of type S in $E \cup \breve{E}$ such that

$$t = r\big[e[u_1, \dots, u_m], x_1, \dots, x_n\big] \quad \text{and} \quad t' = r\big[e'[u_1, \dots, u_m], x_1, \dots, x_n\big],$$

where $u_i[x_1{:}\,T_1, \dots, x_n{:}\,T_n]{:}\,S_i$, for $i \le n$, are arbitrary H-terms.

Define $\equiv_E(H)^* = \langle\, \equiv_E(H)^*_T\colon T \in \text{TYPE}\,\rangle$ to be the reflexive and transitive closure of $\equiv_E(H)$. □

Note that $\equiv_E(H)^*$ is symmetric because $\equiv_E(H)$ is obviously symmetric.

Lemma 2.21 (Alternative Formalization of Equational Deduction)**.** *Let $\langle \Sigma, E\rangle$ be an equational specification and $\forall H(t = t')$ be an H-equation of type T. Then*

$$E \vdash_{\text{NOM}} \forall H(t = t') \quad \textit{iff} \quad t \equiv_E(H)^*_T\ t'.$$

Proof. The proof of the implication from right to left is straightforward and is omitted.

For the implication in the opposite direction we show that $t \equiv_E(H)^*_T\ t'$ for every equation $\forall H(t = t')$ of type T included in E, and that $\equiv_E(H)^*$ is closed under the rules of equational logic in the sense that the set of equations $\forall H(t = t')$ such that $t \equiv_E(H)^*_T\ t'$, where T is the type of $\forall H(t = t')$, is closed under equational deduction. Consider any equation $\forall H(e[x_1{:}\,T_1, \dots, x_m{:}\,T_m] = e'[x_1{:}\,T_1, \dots, x_m{:}\,T_m])$ in E and let S be its type. Taking $r[z{:}\,S, x_1{:}\,T_1, \dots, x_m{:}\,T_m]$ to be $z{:}\,S$ and $u_i[x_1{:}\,T_1, \dots, x_n{:}\,T_n]$ to be $x_i{:}\,T_i$ for each $i \le n$ we get

$$r\big[e[u_1, \dots, u_m], x_1, \dots, x_n\big] = e \quad \text{and} \quad r\big[e'[u_1, \dots, u_m], x_1, \dots, x_n\big] = e'.$$

So by definition $e \equiv_E(H)_S\ e'$ and hence $e \equiv_E(H)^*_S\ e'$.

The relation $\equiv_E(H)^*$ is closed under the rules (*refl*), (*sym*), and (*trans*) by construction. We now show it is also closed under $(\textit{cong})_{\text{NOM}}$. For this purpose we first show that the closure of $\equiv_E(H)$ under $(\textit{cong})_{\text{NOM}}$ is included in $\equiv_E(H)^*$. Let $g \in \mathrm{Op}_n$ and let $T_1, \ldots, T_n \to T$ be an admissible type of g. Suppose $t_i \equiv_E(H)_{T_i}\ t'_i$ for $i \leq n$. To show $g(t_1, \ldots, t_n) \equiv_E(H)^*_T\ g(t'_1, \ldots, t'_n)$ it suffices to show that

$$g(t'_1, \ldots, t'_{i-1}, t_i, t_{i+1}, \ldots, t_n) \equiv_E(H)_T\ g(t'_1, \ldots, t'_{i-1}, t'{}_i, t_{i+1}, \ldots, t_n) \tag{1}$$

for each $i \leq n$. There is an equation $\forall K(e[y_1\colon S_1, \ldots, y_m\colon S_m] = e'[y_1\colon S_1, \ldots, y_m\colon S_m])$ in $E \cup \breve{E}$ of type S such that $t_i = r\big[e[\bar{u}], \bar{x}\big]$ and $t'_i = r\big[e'[\bar{u}], \bar{x}\big]$, where $\bar{u} = u_1, \ldots, u_m$ and $\bar{x} = x_1, \ldots, x_n$. Thus

$$g(t'_1, \ldots, t'_{i-1}, t_i, t_{i+1}, \ldots, t_n) = g\big(t'_1, \ldots, t'_{i-1}, r\big[e[\bar{u}], \bar{x}\big], t_{i+1}, \ldots, t_n\big),$$
$$g(t'_1, \ldots, t'_{i-1}, t'_i, t_{i+1}, \ldots, t_n) = g\big(t'_1, \ldots, t'_{i-1}, r\big[e'[\bar{u}], \bar{x}\big], t_{i+1}, \ldots, t_n\big).$$

So $g(t'_1, \ldots, t'_{i-1}, t_i, t_{i+1}, \ldots, t_n) = s\big[e[\bar{u}], \bar{x}\big]$ and $g(t'_1, \ldots, t'_{i-1}, t'_i, t_{i+1}, \ldots, t_n) = s\big[e'[\bar{u}], \bar{x}\big]$, where

$$s[z\colon T_i, x_1\colon T_1, \ldots, x_n\colon T_n] = g\big(t'_0[\bar{x}], \ldots, t'_{i-1}[\bar{x}], z, t_{i+1}[\bar{x}], \ldots, t_n[\bar{x}]\big).$$

So (1) holds.

Suppose now that, $g \in \mathrm{Op}_n$, $T_1, \ldots, T_n \to T$ is an admissible type of g, and for each $i \leq n$, $t_i \equiv_E(H)^*_{T_i}\ t'_i$. Then for each $i \leq n$ there exist terms $s_i^1, \ldots, s_i^m$ such that

$$t_i = s_i^1 \equiv_E(H)_{T_i}\ s_i^2 \equiv_E(H)_{T_i} \cdots \equiv_E(H)_{T_i}\ s_i^{m-1} \equiv_E(H)_{T_i}\ s_i^m = t'_i$$

(note that he length of the sequences $s_i^1, \ldots, s_i^m$ can be taken to be the same for all i because $\equiv_E(H)_{T_i}$ is reflexive). By the result just established, $g(s_1^1, \ldots, s_n^1) \equiv_E(H)^*_T \cdots \equiv_E(H)^*_T$ $g(s_1^m, \ldots, s_n^m)$. So by transitivity of $\equiv_E(H)^*_T$, $g(t_1, \ldots, t_n) \equiv_E(H)^*_T\ g(t'_1, \ldots, t'_n)$, as desired.

To show that $\equiv_E(H)^*$ is closed under $(\textit{invar})_{\text{NOM}}$, we first show that $\equiv_E(H)$ is closed under $(\textit{invar})_{\text{NOM}}$. Suppose $t[z_1\colon R_1, \ldots, z_p\colon R_p] \equiv_E(H')_R\ t'[z_1\colon R_1, \ldots, z_p\colon R_p]$ for some type context H', and suppose $w_1, \ldots, w_p$ are terms of nominal H-type $R_1, \ldots, R_p$. We must show that $t[w_1, \ldots, w_p] \equiv_E(H)_R\ t'[w_1, \ldots, w_p]$. Now, by definition of $\equiv_E(H')_R$, there is a term $r[z\colon S, z_1\colon R_1, \ldots, z_p\colon R_p]$ and an equation $\forall K(e[y_1\colon S_1, \ldots, y_m\colon S_m] = e'[y_1\colon S_1, \ldots, y_m\colon S_m])$ in $E \cup \breve{E}$ of type S such that

$$t = r\big[e\big[u_1[\bar{z}], \ldots, u_m[\bar{z}]\big], \bar{z}\big] \quad \text{and} \quad t' = r\big[e'\big[u_1[\bar{z}], \ldots, u_m[\bar{z}]\big], \bar{z}\big],$$

where $\bar{z} = z_1, \ldots, z_p$. Now define $v_i[x_1, \ldots, x_n] = u_i\big[w_1[x_1, \ldots, x_n], \ldots, w_p[x_1, \ldots, x_n]\big]$ for every $i \leq m$, and consider the term

$$s[z : S, x_1, \ldots, x_n] = r\big[z : S, w_1[x_1, \ldots, x_n], \ldots, w_p[x_1, \ldots, x_n]\big].$$

Thus

$$t[w_1, \ldots, w_p] = s\big[e[v_1, \ldots, v_m], x_1, \ldots, x_n\big] \quad \text{and}$$
$$t'[w_1, \ldots, w_p] = s\big[e'[v_1, \ldots, v_m], x_1, \ldots, x_n\big],$$

so by definition of $\equiv_E(H)_R$, $t[w_1, \ldots, w_p] \equiv_E(H)_R\ t'[w_1, \ldots, w_p]$.

The closure of $\equiv_E(H)^*$ under $(\textit{invar})_{\text{NOM}}$ is an immediate consequence of the closure of $\equiv_E(H)$. □

3. The Logic of Behavioral Equivalence

The notion of behavioral equivalence is simple in concept: two hidden objects of the same type are behaviorally equivalent if any procedure whose parameter is of this type returns the same visible result when executed with either of the two objects as input. The notion arises from the alternative view of a data structure as a transition system in which the hidden objects represent states of the system and the methods that are not attributes induce transitions between states. (The term “constructors” which we introduced earlier for such methods is appropriate for the algebraic but not for the transition-system paradigm.) At a more abstract level the transition-system paradigm can be characterized as coalgebraic [19, 20, 35] with coinduction as the basis of the corresponding proof theory. (This is in contrast to the algebraic paradigm where induction forms the basis of the proof theory.) Behavioral equivalence has proved to be a useful device for importing transition-system and coalgebraic methods into the algebraic paradigm, and has served as the basis of a formal theory of subtyping in our earlier work [22, 23].

In the context of logic the basic idea is an old one, going back to the beginnings of algebraic logic and the Lindenbaum-Tarski process [38]. More recently it has been the principal feature of *abstract algebraic logic* (which can also be viewed as *coalgebraic logic*; see for instance [4, 8, 30]). In computer science it has also been around for a long time; see [22] for a systematic development of the non-proof-theoretical part of the theory in the context of correct subtyping. A more extensive development that includes parts of the proof theory but not subtyping can be found in the series of papers on hidden algebras; see [15, 16] and the many papers referenced there.

To reason effectively about behavioral equivalence and subtyping, we introduce the concept of a *behavioral consequence*—a relation between sets of equations and equations rather than pairs of hidden objects. The proof theory of behavioral consequence is based on coinduction, and various methods of applying coinduction in a reasonably efficient way have been investigated in the literaure (see for example [15, 33, 34]). We consider one such method here that uses the notion of a *local context*. But verifying behavioral consequence can be a laborious process under the best of circumstances. One of the main things we prove in this section is the closure of behavioral consequence under nominal equational consequence; i.e., that any equation that is nominally provable from a set of behavioral consequences is itself a behavioral consequence. This turns out to be an effective method for proving correct behavioral subtyping in many cases. It works when all the equations associated with a given set of coercion functions can be proved by nominal equational logic from just a few judiciously chosen ones, which then are proved to be behavioral consequences directly by coinduction. An example in this kind is considered in some detail in the sequel.

As was alluded to in the previous paragraph, the most important application of the concept of behavioral consequence in this work is to the specification of coercion functions; correct behavioral subtyping results when the equations for a set of coercion functions are all behavioral consequences. In short, coinduction gives a sound proof theory for behavioral consequence, and hence for correct behavioral subtyping via coercion. This is treated in detail in the next section.

We begin by giving the definition of behavioral equivalence; it requires an auxiliary notion, which following Goguen and Malcolm [15] we call *context* (not to be confused with a type-context).

Definition 3.1 (Context)**.** Let Σ be a hidden signature and T a type. By a (Σ, T)*-context*, we mean a Σ-term of visible type of the form

$$r[z{:}\,T, x_1{:}\,S_1, \ldots, x_m{:}\,S_m], \tag{2}$$

where z, called the *designated variable* of the S-context, is distinct from the $y_1, \ldots, y_m$ and has just one occurrence in r. □

Subtyping does not play a role in the definition of a (Σ, T)-context. Also, we often call a (Σ, T)-context a "T-context" when the signature Σ is understood.

Definition 3.2 (Behavioral Equivalence)**.** Let Σ be a signature, $\boldsymbol{A}$ a Σ-algebra, and T a type. Elements a and a' of $\boldsymbol{A}$ of type T are said to be *behaviorally equivalent* if, for each T-context and all $(b_1, \ldots, b_m) \in A_{S_1} \times \cdots \times A_{S_m}$,

$$r^{\boldsymbol{A}}[a, b_1, \ldots, b_m] = r^{\boldsymbol{A}}[a', b_1, \ldots, b_m]. \tag{3}$$

The TYPE-sorted *behavioral equivalence relation* $\mathcal{BE}(\boldsymbol{A}) = \langle\, \mathcal{BE}_T(\boldsymbol{A}) : T \in \text{TYPE}\,\rangle$ on $\boldsymbol{A}$ is defined in the obvious way, namely, $\mathcal{BE}_T(\boldsymbol{A})$ is the set of all behaviorally equivalent pairs of elements of A_T. □

We will write "$\mathcal{BE}$" for "$\mathcal{BE}(\boldsymbol{A})$" when the algebra $\boldsymbol{A}$ is clear from context. Note that, by taking r to be the variable z itself, we see that two visible data elements are behaviorally equivalent iff they are identical.

A context is the formal representation of a procedure that takes an object a of hidden type T, together with the parameters $b_1, \ldots, b_m$, as input and, upon execution, outputs the visible data element $r^{\boldsymbol{A}}[a, b_1, \ldots, b_m]$. Intuitively, two objects are behaviorally equivalent if they give the same output for all procedures and all choices of parameters.[4]

In its most primitive form coinduction amounts to verifying the equalities (3) for all contexts r, a potentially infinite set. Many of the methods for effectively managing coinduction amount to coming up with a relatively small set of contexts that can be used in place of all of them. We consider one such set now, what Goguen and Malcom call *local* contexts [15]. Intuitively, a local context is one that only applies one attribute to the distinguished context variable.

Definition 3.3. A (Σ, T)-context $r[z{:}\,T, y_1{:}\,S_1, \ldots, y_m{:}\,S_m]$ is *local* if it is of the form

$$r[z{:}\,T, y_1{:}\,S_1, \ldots, y_m{:}\,S_m] = g(h[z, y_1, \ldots, y_m], y_1, \ldots, y_m]), \tag{4}$$

where g is an attribute and h is a term containing the single occurrence of z in r whose only non-constant operation symbols are constructors. □

[4]If the system of parameters is held fixed, we get a related notion of behavioral equivalence localized to the environment. This local notion of behavioral equivalence relates environments rather than individual objects and is the one used in our earlier work on the model theory of correct behavioral subtyping [21, 23]; see Appendix A.

Lemma 3.4. *Let $\boldsymbol{A}$ be a Σ-algebra and T a hidden type. Let $a, a' \in A_T$. Then $a \,\mathcal{BE}\, a'$ iff for every local T-context $r[z\colon T, y_1\colon S_1, \dots, y_m\colon S_m]$ and all $(b_1, \dots, b_m) \in A_{S_1} \times \cdots \times A_{S_m}$,*

$$r^{\boldsymbol{A}}[a, b_1, \dots, b_m] = r^{\boldsymbol{A}}[a', b_1, \dots, b_m]. \tag{5}$$

Proof. The implication from right to left is trivial, since every local context is a priori a context. For the opposite direction, assume that every local context with arbitrary choice of parameters returns the same element when evaluated with input a and a'. Let $r[z\colon T, y_1\colon S_1, \dots, y_m\colon S_m]$ be an arbitrary T-context. Since r is visible and z is hidden, r must contain at least one operation symbol, and its leading operation symbol must be visible or an attribute. Let $r'[z, y_1, \dots, y_m]$ be the smallest subterm of r that includes the lone occurrence of z, and whose leading operation symbol, say g, is visible or an attribute. Since z is hidden, g must be an attribute, and r' must be of the form $g[h(z, \bar{y}), \bar{y}]$, where h is a term whose only non-constant operation symbols are constructors; i.e., r' is a local context. Moreover, r is of the form $q\big[r'[z, \bar{y}], \bar{y}\big]$ for some Σ-term $q[w, \bar{y}]$. By assumption, for all $\bar{b} \in A_{S_1} \times \cdots \times A_{S_m}$, $r'^{\boldsymbol{A}}[a, \bar{b}] = r'^{\boldsymbol{A}}[a', \bar{b}]$. Thus $r^{\boldsymbol{A}}[a, \bar{b}] = q^{\boldsymbol{A}}\big[r'^{\boldsymbol{A}}[a, \bar{b}], \bar{b}\big] = q^{\boldsymbol{A}}\big[r'^{\boldsymbol{A}}[a', \bar{b}], \bar{b}\big] = r^{\boldsymbol{A}}[a', \bar{b}]$. Hence $a \,\mathcal{BE}_T\, a'$. □

Definition 3.5 (Behavioral Consequence)**.** Let $\langle \Sigma, E\rangle$ be a hidden equational specification and H a type context. An H-equation $\forall H(t = t')$ over Σ is a *behavioral consequence of* E, in symbols $E \vDash_{\text{BEH}} \forall H(t = t')$, if, for every $\boldsymbol{A} \in \mathsf{Mod}_{\text{NOM}}(E)$ and every H-environment $\bar{a}$ of $\boldsymbol{A}$, $t^{\boldsymbol{A}}[\bar{a}] \;\mathcal{BE}_T\; t'^{\boldsymbol{A}}[\bar{a}]$, where T is the type of $\forall H(t = t')$. □

It follows immediately from the definition that $\vDash_{\text{BEH}}$ is monotonic, i.e., If $E \vDash_{\text{BEH}} F$, then $E' \vDash_{\text{BEH}} F$ for every set E' of equations such that $E' \supseteq E$.

Definition 3.6 (Behavioral Provability)**.** Let $\langle \Sigma, E\rangle$ be a hidden equational specification, and H a type context. An H-equation $\forall H(t = t')$ of type T over Σ is *behaviorally provable from* E, in symbols $E \vdash_{\text{BEH}} \forall H(t = t')$, if, for every type context $K = \langle y_1\colon S_1, \dots, y_m\colon S_m\rangle$ disjoint from H and every T-context $r[z\colon T, y_1\colon S_1, \dots, y_m\colon S_m]$,

$$E \vdash_{\text{NOM}} \forall(H \cup K)(r[t, y_1, \dots, y_m] = r[t', y_1, \dots, y_m]).$$

□

The following completeness and soundness theorem is an easy corollary of the definitions of $\vDash_{\text{BEH}}$ and $\vdash_{\text{BEH}}$.

Theorem 3.7 (Completeness and Soundness Theorem for Behavioral Logic)**.** *Let $\langle \Sigma, E\rangle$ be a hidden equational specification and H a type context. Then, for every H-equation $\forall H(t = t')$,*

$$E \vDash_{\text{BEH}} \forall H(t = t') \quad \textit{iff} \quad E \vdash_{\text{BEH}} \forall H(t = t').$$

Proof.

$E \models_{\mathrm{BEH}} \forall H(t = t')$

iff (by definition of $\models_{\mathrm{BEH}}$),

for every $\boldsymbol{A} \in \mathsf{Mod}_{\mathrm{NOM}}(E)$ and every H-environment $\bar{a}$,

$t^{\boldsymbol{A}}[\bar{a}]\ \mathcal{BE}_T\ t'^{\boldsymbol{A}}[\bar{a}]$

iff (by definition of $\mathcal{BE}$)

for every $\boldsymbol{A} \in \mathsf{Mod}_{\mathrm{NOM}}(E)$, every H-environment $\bar{a}$,

every type context K, every T-context $r[z, \bar{y}]$, and every K-environment $\bar{b}$,

$r^{\boldsymbol{A}}\big[t^{\boldsymbol{A}}[\bar{a}], \bar{b}\big] = r^{\boldsymbol{A}}\big[t'^{\boldsymbol{A}}[\bar{a}], \bar{b}\big]$

iff (by definition of $\models_{\mathrm{NOM}}$)

for every type context K and every T-context $r[z, \bar{y}]$,

$E \models_{\mathrm{NOM}} \forall(H \cup K)(r[t, y_1, \ldots, y_m] = r[t', y_1, \ldots, y_m])$

iff (by completeness and soundness theorem of equational logic)

for every type context K and every T-context $r[z, \bar{y}]$,

$E \vdash_{\mathrm{NOM}} \forall(H \cup K)(r[t, y_1, \ldots, y_m] = r[t', y_1, \ldots, y_m])$

iff (by definition of $\vdash_{\mathrm{BEH}}$)

$E \vdash_{\mathrm{BEH}} \forall H(t = t')$.

□

Thus, in the sequel we use $\models_{\mathrm{BEH}}$ and $\vdash_{\mathrm{BEH}}$ interchangeably, often referring to the latter as "behavioral consequence".

Corollary 3.8. *Let $\langle \Sigma, E\rangle$ be a hidden equational specification and $\forall H(t = t')$ an equation over Σ. Then $E \vdash_{\mathrm{BEH}} \forall H(t = t')$ iff, for every type context $K = \langle y_1{:}\, S_1, \ldots, y_m{:}\, S_m\rangle$ disjoint from H and every local T-context $r[z{:}\, T, y_1{:}\, S_1, \ldots, y_m{:}\, S_m]$,*

$$E \vdash_{\mathrm{NOM}} \forall(H \cup K)(r[t, y_1, \ldots, y_m] = r[t', y_1, \ldots, y_m]).$$

Proof. It suffices to observe that, if $\forall H(t = t')$ is of visible type, all the equivalences of the preceding proof continue to hold when r is restricted to local contexts. In this case the second equivalence is justified by Lem. 3.4. □

Local contexts can be an effective tool for verifying behavioral provability and hence behavioral consequence in particular cases.

Example 3.9. Each of the three equations

$$\begin{aligned}\texttt{movex}(\texttt{movex}(p, i), j) &= \texttt{movex}(p, \texttt{add}(i, j))\\ \texttt{movey}(\texttt{movey}(p, i), j) &= \texttt{movey}(p, \texttt{add}(i, j))\\ \texttt{movex}(\texttt{movey}(p, i), j) &= \texttt{movey}(\texttt{movex}(p, j), i)\end{aligned}$$

is behaviorally provable from E_{cp} and hence also from E_{cpt} by the monotonicity of $\vdash_{\mathrm{BEH}}$. To verify this for the first equation it suffices by Cor. 3.8 to show both of the following.

(6) $E \vdash_{\mathrm{NOM}} \texttt{x}(h[\texttt{movex}(\texttt{movex}(p, i), j), y_1, \ldots, y_m]) = \texttt{x}(h[\texttt{movex}(p, \texttt{add}(i, j)), y_1, \ldots, y_m])$,

(7) $E \vdash_{\mathrm{NOM}} \texttt{y}(h[\texttt{movex}(\texttt{movex}(p, i), j), y_1, \ldots, y_m]) = \texttt{y}(h[\texttt{movex}(p, \texttt{add}(i, j)), y_1, \ldots, y_m])$

with

$$h(z, y_1, \ldots, y_m) = \texttt{f}_m(\cdots(\texttt{f}_2(\texttt{f}_1(z, y_1), y_2)\cdots), y_m),$$

where $\mathtt{f}_1, \dots, \mathtt{f}_m$ is any sequence of movex's and movey's. We prove (6) and (7) hold by induction on the structure of h. If $m = 0$, i.e., h is a variable, then the two equations take the form

$$E \vdash_{\text{NOM}} \mathtt{x}(\mathtt{movex}(\mathtt{movex}(p, i), j)) = \mathtt{x}(\mathtt{movex}(p, \mathtt{add}(i, j))),$$
$$E \vdash_{\text{NOM}} \mathtt{y}(\mathtt{movex}(\mathtt{movex}(p, i), j)) = \mathtt{y}(\mathtt{movex}(p, \mathtt{add}(i, j))).$$

We verify the first of them and leave the second to the reader.

$$\begin{aligned}
\mathtt{x}(\mathtt{movex}(\mathtt{movex}(p, i), j)) &= \mathtt{add}(\mathtt{x}(\mathtt{movex}(p, i), j), && \text{by axiom } (\textit{x-movex}) \text{ of } E_{pt} \\
&= \mathtt{add}(\mathtt{add}(\mathtt{x}(p), i), j), && \text{by axiom } (\textit{x-movex}) \text{ of } E_{pt} \\
&= \mathtt{add}(\mathtt{x}(p), \mathtt{add}(i, j)), && \text{by axioms for integers} \\
&= \mathtt{x}(\mathtt{movex}(p, \mathtt{add}(i, j))), && \text{by axiom } (\textit{x-movex}) \text{ of } E_{pt}.
\end{aligned}$$

Assume $m > 0$, and consider the case $\mathtt{f}_m = \mathtt{movey}$. Let

$$h'(z, y_1, \dots, y_{m-1}) = \mathtt{f}_{m-1}(\cdots(\mathtt{f}_2(\mathtt{f}_1(z, y_1), y_2)\cdots), y_{m-1}),$$

so that $h(z, y_1, \dots, y_m) = \mathtt{movey}(h'(z, y_1, \dots, y_{m-1}), y_m)$. Then (6) and (7) become

$$\begin{aligned}
E \vdash_{\text{NOM}} &\ \mathtt{x}(\mathtt{movey}(h'[\mathtt{movex}(\mathtt{movex}(p, i), j), y_1, \dots, y_{m-1}], y_m)) \\
&= \mathtt{x}(\mathtt{movey}(h'[\mathtt{movex}(p, \mathtt{add}(i, j)), y_1, \dots, y_{m-1}], y_m),
\end{aligned}$$

$$\begin{aligned}
E \vdash_{\text{NOM}} &\ \mathtt{y}(\mathtt{movey}(h'[\mathtt{movex}(\mathtt{movex}(p, i), j), y_1, \dots, y_{m-1}], y_m)) \\
&= \mathtt{y}(\mathtt{movey}(h'[\mathtt{movex}(p, \mathtt{add}(i, j)), y_1, \dots, y_{m-1}], y_m)).
\end{aligned}$$

We verify the first of them and again leave the second to the reader.

$$\begin{aligned}
&\mathtt{x}(\mathtt{movey}(h'[\mathtt{movex}(\mathtt{movex}(p, i), j), y_1, \dots, y_{m-1}], y_m) \\
&\quad = \mathtt{x}(h'[\mathtt{movex}(\mathtt{movex}(p, i), j), y_1, \dots, y_{m-1}], y_m), && \text{by axiom } (\textit{x-movey}) \text{ of } E_{pt} \\
&\quad = \mathtt{x}(h'[\mathtt{movex}(p, \mathtt{add}(i, j)), y_1, \dots, y_{m-1}], y_m), && \text{by induction hypothesis} \\
&\quad = \mathtt{x}(\mathtt{movey}(h'[\mathtt{movex}(p, \mathtt{add}(i, j)), y_1, \dots, y_{m-1}], y_m)), && \text{by axiom } (\textit{x-movey}) \text{ of } E_{pt}.
\end{aligned}$$

This verifies that the equation $\mathtt{movex}(\mathtt{movex}(p, i), j) = \mathtt{movex}(p, \mathtt{add}(i, j))$ is a formal behavioral equivalence over E_{pt}. The formal behavioral equivalence of $\mathtt{movey}(\mathtt{movey}(p, i), j) = \mathtt{movey}(p, \mathtt{add}(i, j))$ and $\mathtt{movex}(\mathtt{movey}(p, i), j) = \mathtt{movey}(\mathtt{movex}(p, j), i)$ over E_{pt} can be established in a similar way. □

The next theorem gives an alternate proof-theoretic characterization that will be used to prove the closure of behavioral consequence under nominal equational consequence. Simply put it says that an equation $\forall H(t = t')$ is a behavioral consequence of E iff no visible equation is nominally provable from E together with $\forall H(t = t')$ that is not provable from E alone.

Theorem 3.10. *Let $\langle \Sigma, E\rangle$ be a hidden equational specification and let F be a set of Σ-equations. Then the following are equivalent.*

(i) *For every $\forall H(t = t') \in F$, $E \vdash_{\text{BEH}} \forall H(t = t')$.*

(ii) *For all equations $\forall K(s = s')$ of visible type,*

$$E \cup F \vdash_{\text{NOM}} \forall K(s = s') \quad \textit{implies} \quad E \vdash_{\text{NOM}} \forall K(s = s').$$

Proof. We first prove that (ii) implies (i). So assume (ii) holds and let $\forall H(t = t')$ be an equation in F. We show that $E \vdash_{\text{BEH}} \forall H(t = t')$ by verifying the condition of Defn. 3.6. Let $K = \langle y_1{:}\, S_1, \dots, y_m{:}\, S_m\rangle$ be a type context disjoint from H, and let $r[z{:}\, T, y_1{:}\, S_1, \dots, y_m{:}\, S_m]$ be a T-context, where T is the type of $\forall H(t = t')$. By $(cong)_{\text{NOM}}$,

$$E \cup \{\forall H(t = t')\} \vdash_{\text{NOM}} \forall(H \cup K)(r[t, y_1, \dots, y_m] = r[t', y_1, \dots, y_m]).$$

Thus $E \cup F \vdash_{\text{NOM}} \forall(H \cup K)(r[t, y_1, \dots, y_m] = r[t', y_1, \dots, y_m])$, and hence by (ii),

$$E \vdash_{\text{NOM}} \forall(H \cup K)(r[t, y_1, \dots, y_m] = r[t', y_1, \dots, y_m]).$$

So, by Defn. 3.6, $E \vdash_{\text{BEH}} \forall H(t = t')$ for each $\forall H(t = t') \in F$, i.e., (i) holds.

Now assume (i) holds. Let $\forall K(s = s')$ be an equation of visible type V such that $E \cup F \vdash_{\text{NOM}} \forall K(s = s')$. We can assume without loss of generality that $K = \langle y_1{:}\, S_1, \dots, y_m{:}\, S_m\rangle$ is disjoint from $H = \langle x_1{:}\, T_1, \dots, x_m{:}\, T_m\rangle$. By Lem. 2.21, $s \equiv_{E\cup F}(K)^*_V\ s'$. Thus there is a sequence $s_1, \dots, s_n$ of K-terms of type V such that

$$s = s_1 \equiv_{E\cup F}(K)_V\ s_2 \equiv_{E\cup F}(K)_V \cdots \equiv_{E\cup F}(K)_V\ s_{n-1} \equiv_{E\cup F}(K)_V\ s_n = s'.$$

For each $i < n$ we have either $s_i \equiv_E(K)_V\ s_{i+1}$ or

(8) $$s_i \equiv_{\{\forall H(t=t')\}}(K)_V\ s_{i+1} \text{ for some } \forall H(t[x_1, \dots, x_n] = t'[x_1, \dots, x_n]) \in F.$$

In the latter case, by definition of $\equiv_{\{\forall H(t=t')\}}(K)_V$, there is a term $r[z{:}\, T, y_1, \dots, y_m]$ and K-terms $u_1, \dots, u_n$ such that either

$$s_i = r\big[t[u_1, \dots, u_n], y_1, \dots, y_m\big] \quad \text{and} \quad s_{i+1} = r'\big[t'[u_1, \dots, u_n], y_1, \dots, y_m\big],$$

or the same equalities hold with the "t" and "t'" interchanged. $r[z{:}\, T, y_1, \dots, y_m]$ is of the same visible type V as all the s_j and hence a T-context. So, because of the assumption (i), we have by Defn. 3.6 that

$$E \vdash_{\text{NOM}} \forall(H \cup K)(r\big[t[x_1, \dots, x_n], y_1, \dots, y_m\big] = r\big[t'[x_1, \dots, x_n], y_1, \dots, y_m\big]).$$

Hence $E \vdash_{\text{NOM}} \forall K(s_i = s_{i-1})$ by an application of $(invar)_{\text{NOM}}$. So in fact we have $s_i \equiv_E(K)_V\ s_{i+1}$ for each i such that (8) holds. Thus $s \equiv_E(K)^*_V\ s'$, and hence $E \vdash_{\text{NOM}} \forall K(s = s')$ by Lem. 2.21. We conclude that $E \vdash_{\text{BEH}} F$, and hence that (ii) holds. □

Corollary 3.11 (Closure of Behavioral Consequence under Equational Consequence)**.** *Let $\langle \Sigma, E\rangle$ be a specification and let $F \cup \{\forall H(t = t')\}$ be a set of Σ-equations. If $E \vdash_{\text{BEH}} F$ and $E \cup F \vdash_{\text{NOM}} \forall H(t = t')$, then $E \vdash_{\text{BEH}} \forall H(t = t')$.*

Proof. Assume $E \vdash_{\text{BEH}} F$ and $E \cup F \vdash_{\text{NOM}} \forall H(t = t')$. In order to show that $E \vdash_{\text{BEH}} \forall H(t = t')$ it suffices by Thm. 3.10 to show that, for every equation $\forall K(s = s')$ of visible type,

$$E \cup \{\forall H(t = t')\} \vdash_{\text{NOM}} \forall K(s = s') \quad \text{implies} \quad E \vdash_{\text{BEH}} \forall K(s = s').$$

Assume $E \cup \{\forall H(t = t')\} \vdash_{\text{NOM}} \forall K(s = s')$. Then, since $E \cup F \vdash_{\text{NOM}} \forall H(t = t')$, we have $E \cup F \vdash_{\text{NOM}} \forall K(s = s')$ by the transitivity of $\vdash_{\text{NOM}}$. Thus, since $E \vdash_{\text{BEH}} F$ by assumption, we have by Thm. 3.10 that $E \vdash_{\text{NOM}} \forall K(s = s')$. Hence $E \vdash_{\text{BEH}} \forall H(t = t')$. □

Example 3.12. Let F_3 be the set of the three equations of Example 3.9. We showed in example 3.9 that $E_{pt} \vdash_{\text{BEH}} F_3$ and $E_{cpt} \vdash_{\text{BEH}} F_3$. Hence any equation nominally provable from $E_{pt} \cup F_3$ is a behavioral consequence of E_{pt}, and the same is true with "E_{pt}" replaced by "E_{cpt}". This observation will prove to be useful in the next section in proving that $\langle \Sigma_{pt}, E_{cpt} \rangle$ is correctly behaviorally subtyped. $\square$

As another immediate corollary of Theorem 3.10 we get that the behavioral consequence relation is coinductive in the sense that its complement is inductive (relative to the complement of the of the visible equational specification).

Corollary 3.13. *Let $\langle \Sigma, E \rangle$ be an equational specification. Then, for every Σ-equation $\forall H(t = t')$, $E \nvdash_{\text{BEH}} \forall H(t = t')$ iff there exists a visible equation $\forall K(s = s')$ such that*

$$E \cup \{\forall H(t = t')\} \vdash_{\text{NOM}} \forall K(s = s') \quad \textit{and} \quad E \nvdash_{\text{NOM}} \forall K(s = s'). \qquad \square$$

Note that if $\vdash_{\text{NOM}}$ is recursively enumerable (RE), in particular, if E is finite, then $\vdash_{\text{BEH}}$ is at level $\prod_2^0$ in the arithmetical hierarchy. It is shown in [6] that there are finite equational specifications such that $\vdash_{\text{BEH}}$ is $\prod_2^0$-complete. But note also that, if the visible part of $\vdash_{\text{NOM}}$ is recursive, then $\vdash_{\text{BEH}}$ is co-RE.

It will be useful when studying the role of coercion in verifying correct behavioral subtyping to have an analogue for behavioral consequence of the behavioral equivalence relation between the elements of an Σ-algebra.

Definition 3.14 (Formal Behavioral Equivalence Relation)**.** Let $\langle \Sigma, E \rangle$ be an equational specification and let $\boldsymbol{A}$ be a Σ-algebra. By the *formal behavioral equivalence relation over E on $\boldsymbol{A}$* we mean the sorted binary relation

$$\mathcal{FB}(E, \boldsymbol{A}) = \langle\, \mathcal{FB}(E, \boldsymbol{A})_T : T \in \text{TYPE}\,\rangle,$$

where, for each type T, $\mathcal{FB}(E, \boldsymbol{A})_T$ is the set of all pairs $\langle t^{\boldsymbol{A}}[\bar{a}],\, t'^{\boldsymbol{A}}[\bar{a}]\rangle$ such that H is a type context, $\bar{a}$ is an H-environment in $\boldsymbol{A}$, and $\forall H(t = t')$ is an equation of type T such that $E \vdash_{\text{BEH}} \forall H(t = t')$. $\square$

As in the case of behavioral equivalence we write "$\mathcal{FB}(E)$" in place of "$\mathcal{FB}(E, \boldsymbol{A})$" when the algebra $\boldsymbol{A}$ is understood.

As an immediate consequence of the definition we have that, for every model $\boldsymbol{A}$ of E, $\mathcal{FB}(E, \boldsymbol{A}) \subseteq \mathcal{BE}(\boldsymbol{A})$. It is easy to find examples where the inclusion is proper.

4. Equational Specifications with Coercions

In this section we define coercion systems and relate them to specifications. Coercions are specified by their actions on variables, and thus are an effective way to equationally define coercion functions. The next section will show that a specification with coercions automatically has correct behavioral subtyping.

Let Σ be a signature with subtyping. By a *pre-coercion system* for Σ we mean a sorted set of Σ-terms $\{\, c_{S,T}[x_S\colon S] : S, T \in \text{TYPE}, S \le T, \text{ and } \Sigma; \langle x_S\colon S\rangle \vdash c_{S,T}[x_S\colon S]\colon T \,\}$ such that for each $T \in \text{VIS}$, $c_{T,T}[x_T\colon T]$ is lexically equal to x_T. For brevity we usually write $\{\, c_{S,T} \,\}$ for a pre-coercion system.

Example 4.1. We define a pre-coercion system for Σ_{pt}, $C_{pt} = \{c_{S,T}\}$, as follows.

$$\begin{aligned}
c_{\mathtt{int},\mathtt{int}}[x_{\mathtt{int}}\colon \mathtt{int}] &= x_{\mathtt{int}}\\
c_{\mathtt{Pt},\mathtt{Pt}}[x_{\mathtt{Pt}}\colon \mathtt{Pt}] &= \mathtt{movey}(\mathtt{movex}(\mathtt{PtOrigin}, \mathtt{x}(x_{\mathtt{Pt}})), \mathtt{y}(x_{\mathtt{Pt}}))\\
c_{\mathtt{CPt},\mathtt{Pt}}[x_{\mathtt{CPt}}\colon \mathtt{CPt}] &= \mathtt{movey}(\mathtt{movex}(\mathtt{PtOrigin}, \mathtt{x}(x_{\mathtt{CPt}})), \mathtt{y}(x_{\mathtt{CPt}}))\\
c_{\mathtt{CPt},\mathtt{CPt}}[x_{\mathtt{CPt}}\colon \mathtt{CPt}] &= \mathtt{movey}(\mathtt{movex}(\mathtt{changec}(\mathtt{CPtOrigin}, \mathtt{color}(x_{\mathtt{CPt}})), \mathtt{x}(x_{\mathtt{CPt}})), \mathtt{y}(x_{\mathtt{CPt}}))
\end{aligned}$$

Definition 4.2. Let $\langle \Sigma, E\rangle$ be an equational specification and let $C = \{\, c_{S,T}\,\}$ be a pre-coercion system.

(i) Then C is a *coercion system for* $\langle \Sigma, E\rangle$ if, for every $n \in \mathbb{N}$, every $g \in \mathrm{Op}_n$ with admissible type $T_1, \dots, T_n \to T$, and for all $(S_1, \dots, S_n) \le (T_1, \dots, T_n)$,

(*coer-fbeh*) $$E \vdash_{\mathrm{BEH}} \forall H\big(g\big(c_{S_1,T_1}[y_1], \dots, c_{S_n,T_n}[y_n]\big) = c_{S,T}[g(y_1, \dots, y_n)]\big),$$

where $S = \mathrm{ResType}(g, S_1, \dots, S_n)$ and $H = \langle y_1\colon S_1, \dots, y_n\colon S_n\rangle$.

(ii) We say that $\langle \Sigma, E, C\rangle$ is an *equational specification with coercions* if $\langle \Sigma, E\rangle$ is an equational specification and C is a coercion system for $\langle \Sigma, E\rangle$. □

Example 4.3. We show that the pre-coercion system of Example 4.1, C_{pt}, is not a coercion system for $\langle \Sigma_{pt}, E_{pt}\rangle$, and thus $\langle \Sigma_{pt}, E_{pt}, C_{pt}\rangle$ is not an equational specification with coercions. For this purpose we show that one of the coercion equations, viz., the following $\langle cp\colon \mathtt{CPt}, i\colon \mathtt{int}\rangle$-equation, fails to be a formal behavioral equivalence over E_{pt}:

$$\mathtt{movex}(c_{\mathtt{CPt},\mathtt{CPt}}[cp], c_{\mathtt{int},\mathtt{int}}[i]) = c_{\mathtt{CPt},\mathtt{CPt}}[\mathtt{movex}(cp, i)]$$

In view of Def 3.6 it suffices to show that the following equation

(fails-in-pt) $$\mathtt{color}\big(\mathtt{movex}(c_{\mathtt{CPt},\mathtt{CPt}}[cp], c_{\mathtt{int},\mathtt{int}}[i])\big) = \mathtt{color}\big(c_{\mathtt{CPt},\mathtt{CPt}}[\mathtt{movex}(cp, i)]\big)$$

fails to be a nominal equational consequence of E_{pt}, and for this purpose we show that the $\langle \Sigma_{pt}, E_{pt}\rangle$-equation (fails-in-pt) is invalid in the model $\boldsymbol{PT}$ of E_{pt}. To see this consider the $\langle cp\colon \mathtt{CPt}, i\colon \mathtt{int}\rangle$-environment $\langle\langle 3, 4, 5\rangle, 6\rangle$; in this environment we can calculate as follows for the left-hand side of equation (fails-in-pt). It is convenient to present the calculation in the style of Dijkstra and Gries.

$$\begin{aligned}
&\mathtt{color}\big(\mathtt{movex}(c_{\mathtt{CPt},\mathtt{CPt}}[cp], c_{\mathtt{int},\mathtt{int}}[i])\big)^{\boldsymbol{PT}}[\langle 3,4,5\rangle, 6]\\
=\ &\langle\text{by definition of the coercions}\rangle\\
&\mathtt{color}\big(\mathtt{movex}(\mathtt{movey}(\mathtt{movex}(\mathtt{changec}(\mathtt{CPtOrigin}, \mathtt{color}(cp)), \mathtt{x}(cp)), \mathtt{y}(cp))), i)\big)^{\boldsymbol{PT}}\\
&\hfill[\langle 3,4,5\rangle, 6]\\
=\ &\langle\text{by definition of evaluation}\rangle\\
&\mathtt{color}^{\boldsymbol{PT}}\big(\mathtt{movex}^{\boldsymbol{PT}}(\mathtt{movey}^{\boldsymbol{PT}}(\mathtt{movex}^{\boldsymbol{PT}}(\mathtt{changec}^{\boldsymbol{PT}}(\mathtt{CPtOrigin}^{\boldsymbol{PT}}, \mathtt{color}^{\boldsymbol{PT}}(\langle 3,4,5\rangle)),\\
&\hfill \mathtt{x}^{\boldsymbol{PT}}(\langle 3,4,5\rangle)), \mathtt{y}^{\boldsymbol{PT}}(\langle 3,4,5\rangle)), 6)\big)\\
=\ &\langle\text{by definition of } \mathtt{color}, \mathtt{x}, \text{ and } \mathtt{y} \text{ in } \boldsymbol{PT}\rangle\\
&\mathtt{color}^{\boldsymbol{PT}}\big(\mathtt{movex}^{\boldsymbol{PT}}(\mathtt{movey}^{\boldsymbol{PT}}(\mathtt{movex}^{\boldsymbol{PT}}(\mathtt{changec}^{\boldsymbol{PT}}(\mathtt{CPtOrigin}^{\boldsymbol{PT}}, 5), 3), 4), 6)\big)\\
=\ &\langle\text{by definition of } \mathtt{CPtOrigin} \text{ and } \mathtt{changec} \text{ in } \boldsymbol{PT}\rangle\\
&\mathtt{color}^{\boldsymbol{PT}}\big(\mathtt{movex}^{\boldsymbol{PT}}(\mathtt{movey}^{\boldsymbol{PT}}(\mathtt{movex}^{\boldsymbol{PT}}(\langle 0,0,5\rangle), 3), 4), 6)\big)\\
=\ &\langle\text{by definition of } \mathtt{movex} \text{ in } \boldsymbol{PT}\rangle\\
&\mathtt{color}^{\boldsymbol{PT}}\big(\mathtt{movex}^{\boldsymbol{PT}}(\mathtt{movey}^{\boldsymbol{PT}}(\langle 3,0,0\rangle, 4), 6)\big)\\
=\ &\langle\text{by definition of } \mathtt{movey} \text{ in } \boldsymbol{PT}\rangle\\
&\mathtt{color}^{\boldsymbol{PT}}\big(\mathtt{movex}^{\boldsymbol{PT}}(\langle 3,4,0\rangle, 6)\big)
\end{aligned}$$

$=$ $\langle$by definition of $\texttt{movex}$ in $\boldsymbol{PT}\rangle$
  $\texttt{color}^{\boldsymbol{PT}}\big(\langle 9,4,3\rangle\big)$
$=$ $\langle$by definition of $\texttt{color}$ in $\boldsymbol{PT}\rangle$
  3

However, the result obtained above is different than that for the right-hand side of equation (fails-in-pt) in the same environment, as the following calculation shows.

  $\texttt{color}\big(c_{\texttt{CPt},\texttt{CPt}}[\texttt{movex}(cp,i)]^{\boldsymbol{PT}}[\langle 3,4,5\rangle,6]\big)$
$=$ $\langle$by definition of $c_{\texttt{CPt},\texttt{CPt}}[x_{\texttt{CPt}}\colon \texttt{CPt}]\rangle$
  $\texttt{color}\big(\texttt{movey}(\texttt{movex}(\texttt{changec}(\texttt{CPtOrigin},\texttt{color}(\texttt{movex}(cp,i))),\texttt{x}(\texttt{movex}(cp,i))),$
  $\texttt{y}(\texttt{movex}(cp,i)))^{\boldsymbol{PT}}[\langle 3,4,5\rangle,6]\big)$
$=$ $\langle$by definition of evaluation$\rangle$
  $\texttt{color}^{\boldsymbol{PT}}\big(\texttt{movey}^{\boldsymbol{PT}}(\texttt{movex}^{\boldsymbol{PT}}(\texttt{changec}^{\boldsymbol{PT}}(\texttt{CPtOrigin}^{\boldsymbol{PT}},\texttt{color}^{\boldsymbol{PT}}(\texttt{movex}^{\boldsymbol{PT}}(\langle 3,4,5\rangle,6))),$
  $\texttt{x}^{\boldsymbol{PT}}(\texttt{movex}^{\boldsymbol{PT}}(\langle 3,4,5\rangle,6))),\texttt{y}^{\boldsymbol{PT}}(\texttt{movex}^{\boldsymbol{PT}}(\langle 3,4,5\rangle,6)))\big)$
$=$ $\langle$by definition of $\texttt{movex}$ in $\boldsymbol{PT}\rangle$
  $\texttt{color}^{\boldsymbol{PT}}\big(\texttt{movey}^{\boldsymbol{PT}}(\texttt{movex}^{\boldsymbol{PT}}(\texttt{changec}^{\boldsymbol{PT}}(\texttt{CPtOrigin}^{\boldsymbol{PT}},\texttt{color}^{\boldsymbol{PT}}(\langle 9,4,3\rangle),$
  $\texttt{x}^{\boldsymbol{PT}}(\langle 9,4,3\rangle)),\texttt{y}^{\boldsymbol{PT}}(\langle 9,4,3\rangle))\big)$
$=$ $\langle$by definition of $\texttt{color}$, $\texttt{x}$, and $\texttt{y}$ in $\boldsymbol{PT}\rangle$
  $\texttt{color}^{\boldsymbol{PT}}\big(\texttt{movey}^{\boldsymbol{PT}}(\texttt{movex}^{\boldsymbol{PT}}(\texttt{changec}^{\boldsymbol{PT}}(\texttt{CPtOrigin}^{\boldsymbol{PT}},3),9),4)\big)$
$=$ $\langle$by definition of $\texttt{CPtOrigin}$ and $\texttt{changec}$ in $\boldsymbol{PT}\rangle$
  $\texttt{color}^{\boldsymbol{PT}}\big(\texttt{movey}^{\boldsymbol{PT}}(\texttt{movex}^{\boldsymbol{PT}}(\langle 0,0,3\rangle,9),4)\big)$
$=$ $\langle$by definition of $\texttt{movex}$ in $\boldsymbol{PT}\rangle$
  $\texttt{color}^{\boldsymbol{PT}}\big(\texttt{movey}^{\boldsymbol{PT}}(\langle 9,0,0\rangle,4)\big)$
$=$ $\langle$by definition of $\texttt{movey}$ in $\boldsymbol{PT}\rangle$
  $\texttt{color}^{\boldsymbol{PT}}(\langle 9,4,0\rangle)$
$=$ $\langle$by by definition of $\texttt{color}$ in $\boldsymbol{PT}\rangle$
  0 $\square$

Example 4.4. On the other hand, the pre-coercion system of Example 4.1, C_{pt}, is a coercion system for $\langle \Sigma_{pt}, E_{cpt}\rangle$, and hence $\langle \Sigma_{pt}, E_{cpt}, C_{pt}\rangle$ is an equational specification with coercions. Because the coercion $c_{\texttt{int},\texttt{int}}[x_{\texttt{int}}]$ is lexically equal to $x_{\texttt{int}}$, all the equations of the form (*coer-fbeh*) that involve only visible operations are lexical identities. Thus it suffices to consider only operations in the signature that have admissible types that take arguments of the non-visible types $\texttt{Pt}$ and $\texttt{CPt}$ and to show that the associated equations (*coer-fbeh*) are formal behavioral equivalences over E_{cpt}. Thus, in view of Cor. 3.11 it suffices to show that each equation in Figure 7 is a nominal equational consequence of E_{cpt} with the three formal behavioral equivalences of Example 3.9 adjoined.

The first six formal behavioral equivalences in Figure 7, for the operations $\texttt{x}$ and $\texttt{y}$, all have similar proofs. The following proof for the second of these is representative.

for all p: `Pt`, p_1: `Pt`, p_2: `Pt`, cp: `CPt`, cp_1: `CPt`, cp_2: `CPt`, and i: `int`:

$$
\begin{aligned}
&\mathtt{x}(c_{\mathtt{Pt},\mathtt{Pt}}[p]) = c_{\mathtt{int},\mathtt{int}}[\mathtt{x}(p)]\\
&\mathtt{x}(c_{\mathtt{CPt},\mathtt{Pt}}[cp]) = c_{\mathtt{int},\mathtt{int}}[\mathtt{x}(cp)]\\
&\mathtt{x}(c_{\mathtt{CPt},\mathtt{CPt}}[cp]) = c_{\mathtt{int},\mathtt{int}}[\mathtt{x}(cp)]\\
&\mathtt{y}(c_{\mathtt{Pt},\mathtt{Pt}}[p]) = c_{\mathtt{int},\mathtt{int}}[\mathtt{y}(p)]\\
&\mathtt{y}(c_{\mathtt{CPt},\mathtt{Pt}}[cp]) = c_{\mathtt{int},\mathtt{int}}[\mathtt{y}(cp)]\\
&\mathtt{y}(c_{\mathtt{CPt},\mathtt{CPt}}[cp]) = c_{\mathtt{int},\mathtt{int}}[\mathtt{y}(cp)]\\
&\mathtt{movex}(c_{\mathtt{Pt},\mathtt{Pt}}[p], c_{\mathtt{int},\mathtt{int}}[i]) = c_{\mathtt{Pt},\mathtt{Pt}}[\mathtt{movex}(p, i)]\\
&\mathtt{movex}(c_{\mathtt{CPt},\mathtt{Pt}}[cp], c_{\mathtt{int},\mathtt{int}}[i]) = c_{\mathtt{Pt},\mathtt{Pt}}[\mathtt{movex}(cp, i)]\\
&\mathtt{movex}(c_{\mathtt{CPt},\mathtt{CPt}}[cp], c_{\mathtt{int},\mathtt{int}}[i]) = c_{\mathtt{CPt},\mathtt{CPt}}[\mathtt{movex}(cp, i)]\\
&\mathtt{movey}(c_{\mathtt{Pt},\mathtt{Pt}}[p], c_{\mathtt{int},\mathtt{int}}[i]) = c_{\mathtt{Pt},\mathtt{Pt}}[\mathtt{movey}(p, i)]\\
&\mathtt{movey}(c_{\mathtt{CPt},\mathtt{Pt}}[cp], c_{\mathtt{int},\mathtt{int}}[i]) = c_{\mathtt{Pt},\mathtt{Pt}}[\mathtt{movey}(cp, i)]\\
&\mathtt{movey}(c_{\mathtt{CPt},\mathtt{CPt}}[cp], c_{\mathtt{int},\mathtt{int}}[i]) = c_{\mathtt{CPt},\mathtt{CPt}}[\mathtt{movey}(cp, i)]\\
&\mathtt{xdiff}(c_{\mathtt{Pt},\mathtt{Pt}}[p_1], c_{\mathtt{Pt},\mathtt{Pt}}[p_2]) = c_{\mathtt{int},\mathtt{int}}[\mathtt{xdiff}(p_1, p_2)]\\
&\mathtt{xdiff}(c_{\mathtt{CPt},\mathtt{Pt}}[cp], c_{\mathtt{Pt},\mathtt{Pt}}[p]) = c_{\mathtt{int},\mathtt{int}}[\mathtt{xdiff}(cp, p)]\\
&\mathtt{xdiff}(c_{\mathtt{Pt},\mathtt{Pt}}[p], c_{\mathtt{CPt},\mathtt{Pt}}[cp]) = c_{\mathtt{int},\mathtt{int}}[\mathtt{xdiff}(p, cp)]\\
&\mathtt{xdiff}(c_{\mathtt{CPt},\mathtt{Pt}}[cp_1], c_{\mathtt{CPt},\mathtt{Pt}}[cp_2]) = c_{\mathtt{int},\mathtt{int}}[\mathtt{xdiff}(cp_1, cp_2)]
\end{aligned}
$$

FIGURE 7. Equations to be proved formal behavioral equivalences over E_{cpt} to show that the pre-coercion system of Exam. 4.1, C_{pt}, is a coercion system for $\langle \Sigma_{pt}, E_{cpt}\rangle$.

$$
\begin{aligned}
&\mathtt{x}(c_{\mathtt{CPt},\mathtt{Pt}}[cp])\\
&\quad = \mathtt{x}(\mathtt{movey}(\mathtt{movex}(\mathtt{PtOrigin}, \mathtt{x}(cp)), \mathtt{y}(cp))), && \text{by def. of } c_{\mathtt{CPt},\mathtt{Pt}}[x_{\mathtt{CPt}}\colon \mathtt{CPt}]\\
&\quad = \mathtt{x}(\mathtt{movex}(\mathtt{PtOrigin}, \mathtt{x}(cp))), && \text{by axiom } (\textit{x-movey}) \text{ of } E_{cpt}\\
&\quad = \mathtt{add}(\mathtt{x}(\mathtt{PtOrigin}), \mathtt{x}(cp)), && \text{by axiom } (\textit{x-movex}) \text{ of } E_{cpt}\\
&\quad = \mathtt{add}(\mathtt{0}, \mathtt{x}(cp)), && \text{by axiom in } (\textit{x-PtOrigin}) \text{ of } E_{cpt}\\
&\quad = \mathtt{x}(cp), && \text{by axioms for the integers}\\
&\quad = c_{\mathtt{int},\mathtt{int}}[\mathtt{x}(cp)], && \text{by def. of } c_{\mathtt{int},\mathtt{int}}[x_{\mathtt{int}}\colon \mathtt{int}].
\end{aligned}
$$

The second set of six formal behavioral equivalences in Figure 7, for the operations `movex` and `movey`, all have similar proofs. The following proof for the third of these (which is equation (fails-in-pt)) is representative. Note the crucial use of the equations from Example 3.9.

$$
\begin{aligned}
&\quad \mathtt{movex}(c_{\mathtt{CPt},\mathtt{CPt}}[cp], c_{\mathtt{int},\mathtt{int}}[i])\\
= &\quad \langle\text{by definition of the coercions}\rangle\\
&\quad \mathtt{movex}(\mathtt{movey}(\mathtt{movex}(\mathtt{changec}(\mathtt{CPtOrigin}, \mathtt{color}(cp)), \mathtt{x}(cp)), \mathtt{y}(cp)), i)\\
= &\quad \langle\text{by the third formal behavioral equivalence in Example 3.9}\rangle\\
&\quad \mathtt{movey}(\mathtt{movex}(\mathtt{movex}(\mathtt{changec}(\mathtt{CPtOrigin}, \mathtt{color}(cp)), \mathtt{x}(cp)), i), \mathtt{y}(cp))\\
= &\quad \langle\text{by the first formal behavioral equivalence in Example 3.9}\rangle\\
&\quad \mathtt{movey}(\mathtt{movex}(\mathtt{changec}(\mathtt{CPtOrigin}, \mathtt{color}(cp)), \mathtt{add}(\mathtt{x}(cp), i)), \mathtt{y}(cp))\\
= &\quad \langle\text{by axiom } (\textit{x-movex}) \text{ of } E_{cpt}\rangle
\end{aligned}
$$

$$\begin{array}{ll}
& \texttt{movey}(\texttt{movex}(\texttt{changec}(\texttt{CPtOrigin}, \texttt{color}(cp)), \texttt{x}(\texttt{movex}(cp, i))), \texttt{y}(cp)) \\
= & \langle \text{by axiom } (\textit{color-movex}) \text{ of } E_{cpt} \rangle \\
& \texttt{movey}(\texttt{movex}(\texttt{changec}(\texttt{CPtOrigin}, \texttt{color}(\texttt{movex}(cp, i))), \texttt{x}(\texttt{movex}(cp, i))), \texttt{y}(cp)) \\
= & \langle \text{by axiom } (\textit{color-movey} \text{ of } E_{cpt} \rangle \\
& \texttt{movey}(\texttt{movex}(\texttt{changec}(\texttt{CPtOrigin}, \texttt{color}(\texttt{movex}(cp, i))), \texttt{x}(\texttt{movex}(cp, i))), \texttt{y}(\texttt{movex}(cp, i))) \\
= & \langle \text{by definition of } c_{\texttt{CPt},\texttt{CPt}}[x_{\texttt{CPt}} \colon \texttt{CPt}] \rangle \\
& c_{\texttt{CPt},\texttt{CPt}}[\texttt{movex}(cp, i)]
\end{array}$$

The last four formal behavioral equivalences in Figure 7 have straight-forward proofs. □

5. Correct Behavioral Subtyping

We show in this section that every specification with coercions is sound in the sense that it implies correct behavioral subtyping. This requires a few preliminaries.

Recall that we defined an algebra to be a model of an equational specification $\langle \Sigma, E\rangle$, (i.e., a member of $\mathsf{Mod}(E)$), if the equations of E are satisfied in every dynamic environment; an alternative, narrower notion of model requires that the equations of E be satisfied by every virtual environment. As noted earlier a Σ-algebra is a model in this sense if and only if its order-sorted transform is model (in the wider sense); so restricting ourselves to this narrower notion of model is tantamount to considering only order-sorted models. But order-sorted models are not a suitable framework for developing a useful theory of correct behavioral subtyping because every class of order-sorted algebras, which meets the criteria of a specification given in [23], is automatically correctly behaviorally subtyped; this idea is made precise in Section 7. Therefore, to avoid begging the question of what correct behavioral subtyping means, we restrict our attention to models of $\langle \Sigma, E\rangle$ in the wider sense. Moreover, we consider only nominal models.

Definition 5.1. Let $\langle \Sigma, E\rangle$ be an equational specification, Then $\langle \Sigma, E\rangle$ is *correctly behaviorally subtyped* if $\mathsf{Mod}_{\mathrm{NOM}}(E)$ is correctly behaviorally subtyped. □

See Appendix A (and our earlier work [23]) for the model-theoretic definition of correct behavioral subtyping. Instead of using the model-theoretic definition directly, we use a result from our earlier work that nominal standard simulations are sound for correct behavioral subtyping [21, 23]. The reason for doing this is that standard simulations correspond directly to coercion systems. We now define the notions of homomorphic standard relation and nominal standard simulation between algebras.

Definition 5.2 (Standard Homomorphic Relation, VIS-Identical, Nominal [22, 23])**.** Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be Σ-algebras. A *homomorphic standard relation between* $\boldsymbol{A}$ *and* $\boldsymbol{B}$ is a sorted binary relation $\mathcal{R} = \langle\, \mathcal{R}_T : T \in \mathrm{TYPE}, \mathcal{R}_T \subseteq A^{\circ}_T \times B^{\circ}_T \,\rangle$ such that the following *homomorphism* condition holds: for all types T and $(S_1, \dots, S_n) \leq (T_1, \dots, T_n)$, for every operation $g \in \mathrm{Op}_n$ such that $T = \mathrm{ResType}(g, T_1, \dots, T_n)$, and for all $(a_1, \dots, a_n) \in A_{S_1} \times \cdots \times A_{S_n}$ and $(b_1, \dots, b_n) \in B_{T_1} \times \cdots \times B_{T_n}$,

$$(a_1, \dots, a_n)\, \mathcal{R}_{T_1} \times \cdots \times \mathcal{R}_{T_n}\, (b_1, \dots, b_n) \quad \textit{implies} \quad g^{\boldsymbol{A}}(a_1, \dots, a_n)\, \mathcal{R}_T\, g^{\boldsymbol{B}}(b_1, \dots, b_n).$$

A homomorphic standard relation $\mathcal{R}$ is VIS-*identical* if the visible reducts of $\boldsymbol{A}$ and $\boldsymbol{B}$ are identical and $\mathcal{R}_V$ is included in the identity relation on A_V $(= B_V)$ for every visible type V, i.e., the only visible element that a given visible element can simulate is itself.

A homomorphic standard relation $\mathcal{R}$ is *nominal* if $\mathcal{R}_T \subseteq A_T \times B_T$ for every type T. □

Recall the definition (Def. 3.14) of the formal behavioral equivalence relation $\mathcal{FB}(E, \boldsymbol{A})$ on $\boldsymbol{A}$ over an equational specification E.

Lemma 5.3. *Let $\langle \Sigma, E\rangle$ be an equational specification and let $\boldsymbol{A}$ be a nominal Σ-algebra. $\mathcal{FB}(E, \boldsymbol{A})$ is a nominal homomorphic standard relation on $\boldsymbol{A}$ and itself. Moreover, if $\boldsymbol{A}$ is a model of E, then $\mathcal{FB}(E, \boldsymbol{A})$ is* VIS*-identical.*

Proof. We observe first of all that $\mathcal{FB}(E, \boldsymbol{A})$ is nominal because $\boldsymbol{A}$ is nominal, so Def. 3.14 gives a nominal relation. Let $g \in \mathrm{Op}_n$, and let $T_1, \dots, T_n \to T$ be an admissible type of g. Let $(a_1, \dots, a_n)$ and $(b_1, \dots, b_n)$ be sequences in $A_{T_1} \times \cdots \times A_{T_n}$ such that $a_i \, \mathcal{FB}(E)_{T_i} \, b_i$ for each $i \le n$. For each $i \le n$ choose a type context H_i, an H_i-environment $\bar{c}_i$ in $\boldsymbol{A}$, and an H_i-equation $\forall H_i(t_i = t_i')$ such that $E \vdash_{\text{BEH}} \forall H_i(t_i = t_i')$ and $\langle a_i, b_i\rangle = \langle t_i^{\boldsymbol{A}}[\bar{c}_i],\, t_i'^{\boldsymbol{A}}[\bar{c}_i]\rangle$ for every $i \le n$. Clearly these choices can be made so that the $\mathrm{Var}(H_i)$ are pairwise disjoint. Let $H = H_1 \cup \cdots \cup H_n$ and $\bar{c} = \bar{c}_1 \cup \cdots \cup \bar{c}_n$. Then by the fact that behavioral consequence is closed under equational consequence (Cor. 3.11) we have $E \vdash_{\text{BEH}} \forall H(g(t_1, \dots, t_n) = g(t_1', \dots, t_n'))$. Thus $\langle g^{\boldsymbol{A}}(a_1, \dots, a_n),\, g^{\boldsymbol{A}}(b_1, \dots, b_n)\rangle = \langle g(t_1, \dots, t_n)^{\boldsymbol{A}}[\bar{c}],\, g(t_1', \dots, t_n')^{\boldsymbol{A}}[\bar{c}]\rangle \in \mathcal{FB}(E)_T$. And so $\mathcal{FB}(E)$ is a homomorphic standard relation on $\boldsymbol{A}$.

Now assume $\boldsymbol{A}$ is a model of E. Suppose $V \in \mathrm{VIS}$ and $a \, \mathcal{FB}(E)_V \, b$. Choose an type context H, an H-environment $\bar{c}$, and an H-equation $\forall H(t = t')$ of type V such that $E \vdash_{\text{BEH}} \forall H(t = t')$ and $\langle a, b\rangle = \langle t^{\boldsymbol{A}}[\bar{c}],\, t'^{\boldsymbol{A}}[\bar{c}]\rangle$. Trivially, $E \cup \{\forall H(t = t')\} \vdash_{\text{NOM}} \forall H(t = t')$, and hence, since $\forall H(t = t')$ is of visible type, $E \vdash_{\text{NOM}} \forall H(t = t')$ by Thm. 3.10. Thus $t^{\boldsymbol{A}}[\bar{c}] = t'^{\boldsymbol{A}}[\bar{c}]$, since $\boldsymbol{A}$ is a model of E, i.e., $a = b$. Thus $\mathcal{FB}(E)$ is VIS-identical. □

It is clear from its definition that $\mathcal{FB}(E)$ is sorted equivalence relation on A, it particular it is transitive.

Definition 5.4 (Nominal Standard Simulation [23]). A standard relation, $\mathcal{R}$, between $\boldsymbol{A}$ and $\boldsymbol{B}$ is a *nominal standard simulation of $\boldsymbol{B}$ by $\boldsymbol{A}$* if $\mathcal{R}$ is homomorphic, VIS-identical, and if for every type T, and for every element a of $\boldsymbol{A}$ of virtual type T (i.e., for every element $a \in A_T^{\circ}$), there is an element b of $\boldsymbol{B}$ of dynamic type T (i.e., $b \in B_T$), such that $a \, \mathcal{R}_T \, b$. □

The meaning of "nominal" for a standard simulation is different from that of a homomorphic standard relation. For nominal standard simulations, for every element of $\boldsymbol{A}$ and each of its virtual types, there is an element of $\boldsymbol{B}$ of corresponding dynamic type to which it is related; however, it may also be related to elements of $\boldsymbol{B}$ that are not of corresponding dynamic type. On the other hand, there may be elements of $\boldsymbol{A}$ that a homomorphic standard relation relates to no element of $\boldsymbol{B}$, but each element to which it is related must be of the same dynamic type. So a nominal standard simulation need not be nominal as a homomorphic relation and vice-versa.

The statement of the algebraic soundness theorem below uses as a hypothesis the fixed data assumption that was introduced in Section 2.4 but not actually used up to this point. This is the assumption that the class of restricted models of a specification is VIS-categorical, i.e., that their visible reducts are the same.

Theorem 5.5 (Algebraic Soundness Theorem [23]). *Let $\langle \Sigma, E\rangle$ be an equational specification and let $\mathsf{Mod}_{\text{NOM}}(E)$ be* VIS*-categorical. If for every $\boldsymbol{A} \in \mathsf{Mod}_{\text{NOM}}(E)$ there exists a $\boldsymbol{B} \in \mathsf{Mod}_{\text{NOM}}(E)$ and a nominal standard simulation of $\boldsymbol{B}$ by $\boldsymbol{A}$, then $\langle \Sigma, E\rangle$ is correctly behaviorally subtyped.*

The proof of this theorem is given in Section A.1. □

Theorem 5.6 (Proof-Theoretic Soundness Theorem)**.** *Let $\langle \Sigma, E, C\rangle$ be an equational specification with coercions. Then $\langle \Sigma, E\rangle$ is correctly behaviorally subtyped.*

Proof. Let $C = \{ c_{S,T} \}$ be the coercion system. Let $\boldsymbol{A} \in \mathsf{Mod}_{\text{NOM}}(E)$. Let $\mathcal{C}^{\boldsymbol{A}} = \langle \mathcal{C}^{\boldsymbol{A}}_T \colon T \in \text{TYPE}\,\rangle$ be the sorted binary relation such that for each type T, $\mathcal{C}^{\boldsymbol{A}}_T$ is defined as follows:

$$\mathcal{C}^{\boldsymbol{A}}_T := \{ \langle a, c_{S,T}[x_S : S]^{\boldsymbol{A}}[a]\rangle : S \leq T, a \in A_S \}.$$

Note that $\mathcal{C}^{\boldsymbol{A}}$ is a sorted function, i.e., $\mathcal{C}^{\boldsymbol{A}}_T$ is a function from A_S to A_T for all $S, T \in$ TYPE such that $S \leq T$; that is, $\mathcal{C}^{\boldsymbol{A}}_T$ is a function from A°_T to A_T.

Let $\mathcal{C}^{\boldsymbol{A}};\ \mathcal{FB}(E, \boldsymbol{A}) = \langle \mathcal{C}^{\boldsymbol{A}}_T;\ \mathcal{FB}(E, \boldsymbol{A})_T : T \in \text{TYPE}\,\rangle$ be the composition of the two relations; that is, $a\ \mathcal{C}^{\boldsymbol{A}}_T;\ \mathcal{FB}(E, \boldsymbol{A})_T\ b$ if there exists a d such that $a\ \mathcal{C}^{\boldsymbol{A}}_T\ d$ and $d\,\mathcal{FB}(E, \boldsymbol{A})_T\ b$. We will show that $\mathcal{C}^{\boldsymbol{A}};\ \mathcal{FB}(E, \boldsymbol{A})$ is a nominal standard simulation of $\boldsymbol{A}$ by itself.

We note first of all that $\mathcal{C}^{\boldsymbol{A}};\ \mathcal{FB}(E, \boldsymbol{A})$ is nominal (as a simulation) since $\boldsymbol{A}$ is nominal and $\mathcal{FB}(E, \boldsymbol{A})$ is nominal (as a homomorphic relation). For each visible type $V \in$ VIS, $\mathcal{C}^{\boldsymbol{A}}_V$ is the identity function, because $c_{V,V}[x_V \colon V]$ is lexically equal to x_V, and $\mathcal{FB}(E, \boldsymbol{A})_V$ is also the identity relation, by Lem. 5.3. Thus $\mathcal{C}^{\boldsymbol{A}};\ \mathcal{FB}(E, \boldsymbol{A})$ is VIS-identical. We now verify the homomorphism condition.

Let $g \in \text{Op}_n$ with admissible types $T_1, \ldots, T_n \to T$ and $S_1, \ldots, S_n \to S$ with $(S_1, \ldots, S_n) \leq (T_1, \ldots, T_n)$, and hence necessarily also $S \leq T$. Let $(a_1, \ldots, a_n) \in A_{S_1} \times \cdots \times A_{S_n}$ and $(b_1, \ldots, b_n) \in A_{T_1} \times \cdots \times A_{T_n}$ such that

$$a_i\ \mathcal{C}^{\boldsymbol{A}}_{T_i};\ \mathcal{FB}(E, \boldsymbol{A})_{T_i}\ b_i \text{ for each } i \leq n. \tag{9}$$

Then by definition of the composition of relations, there are $(d_1, \ldots, d_n) \in A_{T_1} \times \cdots \times A_{T_n}$ such that

$$a_i\ \ \mathcal{C}^{\boldsymbol{A}}_{T_i}\ \ d_i\ \mathcal{FB}(E, \boldsymbol{A})_{T_i}\ b_i \text{ for each } i \leq n. \tag{10}$$

So by definition of $\mathcal{C}^{\boldsymbol{A}}$, for each $i \leq n$, $d_i = c^{\boldsymbol{A}}_{S_i,T_i}[a_i]$. We calculate as follows.

$$\begin{aligned}
g^{\boldsymbol{A}}(a_1, \ldots, a_n)\ \ \mathcal{C}^{\boldsymbol{A}}_T\ & c_{S,T}[x_S \colon S]^{\boldsymbol{A}}[g^{\boldsymbol{A}}(a_1, \ldots, a_n)] \\
&= c_{S,T}\big[g(c_{S_1,T_1}[x_{S_1}], \ldots, c_{S_n,T_n}[x_{S_n}])\big]^{\boldsymbol{A}}[a_1, \ldots, a_n] \\
&\qquad \text{by definition of evaluation} \\
\mathcal{FB}(E, \boldsymbol{A})_T\ & g(c_{S_1,T_1}[x_{S_1}], \ldots, c_{S_n,T_n}[x_{S_n}])^{\boldsymbol{A}}[a_1, \ldots, a_n] \\
&\qquad \text{by definition of coercion system} \\
&= g^{\boldsymbol{A}}(c^{\boldsymbol{A}}_{S_1,T_1}[a_1], \ldots, c^{\boldsymbol{A}}_{S_n,T_n}[a_n]) \\
&\qquad \text{by definition of evaluation} \\
&= g^{\boldsymbol{A}}(d_1, \ldots, d_n).
\end{aligned}$$

Thus $g^{\boldsymbol{A}}(a_1, \ldots, a_n)\ \ \mathcal{C}^{\boldsymbol{A}}_T;\ \mathcal{FB}(E, \boldsymbol{A})_T\ g^{\boldsymbol{A}}(d_1, \ldots, d_n)$. Now from assumption (10) and from the fact that $\mathcal{FB}(E, \boldsymbol{A})$ is a standard homomorphic relation, we obtain

$$g^{\boldsymbol{A}}(d_1, \ldots, d_n)\ \mathcal{FB}(E, \boldsymbol{A})_T\ g^{\boldsymbol{A}}(b_1, \ldots, b_n).$$

Putting this together with the above calculation we obtain that

$$g^{\boldsymbol{A}}(a_1,\ldots,a_n)\ \mathcal{C}_T^{\boldsymbol{A}};\mathcal{FB}(E,\boldsymbol{A})_T\,;\mathcal{FB}(E,\boldsymbol{A})_T\ g^{\boldsymbol{A}}(b_1,\ldots,b_n),$$

and hence $g^{\boldsymbol{A}}(a_1,\ldots,a_n)\ \mathcal{C}_T^{\boldsymbol{A}};\mathcal{FB}(E,\boldsymbol{A})_T\ g^{\boldsymbol{A}}(b_1,\ldots,b_n)$ since $\mathcal{FB}(E,\boldsymbol{A})$ is transitive.

So $\mathcal{C}_T^{\boldsymbol{A}};\ \mathcal{FB}(E,\boldsymbol{A})_T$ is a nominal simulation of $\boldsymbol{A}$ by itself. Thus $\langle\Sigma, E\rangle$ is correctly behaviorally subtyped by the algebraic soundness theorem, Thm. 5.5. □

6. Coercion Systems are not Complete

In this section we give an counterexample to the converse of Theorem 5.6. That is, this example shows that a coercion system does not always exist for an equational specification that is correctly behaviorally subtyped.

Example 6.1. Let Σ_3 be a signature with the three types: `bool`, `sort`, and `subsort`. The only visible type is `bool`, and the subtype relation is such that $T \leq T$, for each type T, and also `subsort` $\leq$ `sort`. There are four constants: `s0` and `s1` of type `sort`, and `ss0` and `ss1` of type `subsort`. The only operation, apart from the standard booleans, is the unary operation `test` with $\mathrm{ResType}(\texttt{test},\texttt{sort}) = \mathrm{ResType}(\texttt{test},\texttt{subsort}) = \texttt{bool}$. The set of equations, E_3, contains only the boolean axioms and the equations shown in Figure 8. (The constants `ss0` and `ss1` are not involved in these equations; they are introduced here for use in the following example.) The class of models, $\mathsf{Mod}_{\mathrm{NOM}}(E_3)$, consists of all models $\boldsymbol{A}$ of E_3 such that $\boldsymbol{A}_{\texttt{bool}}$ is the two-element boolean algebra $\{\mathrm{tt},\mathrm{ff}\}$, where $\texttt{true}^{\boldsymbol{A}} = \mathrm{tt}$ and $\texttt{false}^{\boldsymbol{A}} = \mathrm{ff}$.

(*test-s0*)	$\texttt{test}(\texttt{s0}) = \texttt{false}$
(*test-s1*)	$\texttt{test}(\texttt{s1}) = \texttt{true}$

Figure 8. Equations in E_3; to save space, the standard equations for the booleans are not shown.

To show that $\langle\Sigma_3, E_3\rangle$ is correctly behaviorally subtyped, by Theorem 5.5 it suffices to show that for all restricted models, $\boldsymbol{A}$ in $\mathsf{Mod}_{\mathrm{NOM}}(E_3)$, there is an $\boldsymbol{B} \in \mathsf{Mod}_{\mathrm{NOM}}(E_3)$ such that there is a nominal standard simulation $\mathcal{R}$ of $\boldsymbol{B}$ by $\boldsymbol{A}$. So let $\boldsymbol{A}$ be a restricted model in $\mathsf{Mod}_{\mathrm{NOM}}(E_3)$. Our choice for $\boldsymbol{B}$ is the algebra $\boldsymbol{A}$ itself. To define the required simulation, it is helpful to define two data elements $b_{\mathrm{t}}, b_{\mathrm{f}} \in A_{\texttt{sort}}$ such that $b_{\mathrm{f}} = \texttt{s0}^{\boldsymbol{A}}$ and $b_{\mathrm{t}} = \texttt{s1}^{\boldsymbol{A}}$. Note that by the equations in E_3, $\texttt{test}^{\boldsymbol{A}}(b_{\mathrm{f}}) = \mathrm{ff}$ and $\texttt{test}^{\boldsymbol{A}}(b_{\mathrm{t}}) = \mathrm{tt}$.

Now define $\mathcal{R}$ such that $\mathcal{R}_{\texttt{bool}}$ and $\mathcal{R}_{\texttt{subsort}}$ are the identity relations on the two-element boolean algebra and $A_{\texttt{subsort}}$, respectively. $\mathcal{R}_{\texttt{sort}} \cap (A_{\texttt{sort}} \times B_{\texttt{sort}})$ is the identity relation, and $\mathcal{R}_{\texttt{sort}} \cap (A_{\texttt{subsort}} \times B_{\texttt{sort}})$ is

$$\{\, a \in A_{\texttt{subsort}} : \texttt{test}^{\boldsymbol{A}}(a) = \mathrm{tt}\,\} \times \{b_{\mathrm{t}}\} \cup \{\, a \in A_{\texttt{subsort}} : \texttt{test}^{\boldsymbol{A}}(a) = \mathrm{ff}\,\} \times \{b_{\mathrm{f}}\}.$$

It is easy to see $\mathcal{R}$ is a homomorphic. For example, suppose $a\,\mathcal{R}_{\texttt{sort}}\,b$, with $a \in A_{\texttt{subsort}}$ and $b \in A_{\texttt{sort}}$. If $\texttt{test}^{\boldsymbol{A}}(a) = \mathrm{tt}$, then $b = b_{\mathrm{t}}$, and if $\texttt{test}^{\boldsymbol{A}}(a) = \mathrm{ff}$, then $b = b_{\mathrm{f}}$; in both case we have $\texttt{test}^{\boldsymbol{A}}(a) = \texttt{test}^{\boldsymbol{A}}(b)$. $\mathcal{R}$ is obviously nominal and VIS-identical. So it is a nominal, standard simulation.

There is however no coercion system for $\langle\Sigma_3, E_3\rangle$. We show this by contradiction. Suppose that there were a coercion system $\{\, c_{S,T}\,\}$. The only Σ_3-terms of nominal type `sort` are variables and the two constants `s0` and `s1`. So there are only three possibilities for coercions from `subsort` to `sort`.

The first possibility for this coercion is that the coercion defined using a variable of type `sort`, such as:

$$c_{\mathtt{subsort},\mathtt{sort}}[x_{\mathtt{subsort}}\colon \mathtt{subsort}] = x_{\mathtt{sort}}.$$

In this case, it follows that

$$\mathtt{test}(c_{\mathtt{subsort},\mathtt{sort}}[x_{\mathtt{subsort}}\colon \mathtt{subsort}]) \quad \text{is the term} \quad \mathtt{test}(x_{\mathtt{sort}})$$

and because `bool` is a visible type, $c_{\mathtt{bool},\mathtt{bool}}$ is the identity, and thus

$$c_{\mathtt{bool},\mathtt{bool}}[\mathtt{test}(x_{\mathtt{subsort}})] \quad \text{is the term} \quad \mathtt{test}(x_{\mathtt{subsort}}).$$

Taking the operation g to be `test`, S to be `subsort` and T to be `sort` in the defining condition (*coer-ident*) for a coercion system, we get

$$E_3 \vdash \mathtt{test}(x_{\mathtt{sort}}) = \mathtt{test}(x_{\mathtt{subsort}}).$$

But this is invalid. To see this, let $\boldsymbol{D} \in \mathsf{Mod}_{\mathrm{NOM}}(E_3)$, and let $d \in D_{\mathtt{subsort}}$ be given. Then a counter-example to the validity of this equation is provided by the nominal environment that maps $x_{\mathtt{subsort}}$ to d and $x_{\mathtt{sort}}$ to $\mathtt{s1}^{\boldsymbol{D}}$ if $\mathtt{test}^{\boldsymbol{D}}(d) = \mathrm{ff}$, and $\mathtt{s0}^{\boldsymbol{D}}$ otherwise. For example, if $\mathtt{test}^{\boldsymbol{D}}(d) = \mathrm{ff}$, then the right hand side of this equation is ff, while the left hand side is $\mathtt{test(s1)}^{\boldsymbol{D}} = \mathrm{tt}$.

The second possibility for this coercion is that the coercion defined using the constant `s0` of type `sort`, such as:

$$c_{\mathtt{subsort},\mathtt{sort}}[x_{\mathtt{subsort}}\colon \mathtt{subsort}] = \mathtt{s0}.$$

As above, it follows that $\mathtt{test}(c_{\mathtt{subsort},\mathtt{sort}}[x_{\mathtt{subsort}}\colon \mathtt{subsort}])$ is the term $\mathtt{test}(\mathtt{s0})$ and thus that

$$E_3 \vdash \mathtt{test}(s0) = \mathtt{test}(x_{\mathtt{subsort}}).$$

But this is also invalid, because there are models of E_3 in which `test` returns tt for elements of the subsort.

Similarly the third possibility for this coercion, that it is defined using the constant `s1` of type `sort` also leads to an invalid equation. Hence there is no coercion system for this example. □

Note that $\langle \Sigma_3, E_3 \rangle$ in the above example has only unary methods (see Thm. A.3 below), so the algebraic completeness theorem for specifications with only unary methods does not lead to a corresponding completeness result for coercion.

Example 6.2. Coercions are also not complete for term-generated specifications. To see this, restrict the set of models in the previous example to be term-generated. This does not change the example's character or its conclusions. □

The above examples are quite underspecified; even though they have constructors for objects of type `sort` and `subsort`, the constructors of type `subsort` are unconstrained by the equations in E_3. A similar result can be shown if there were no constructors at all. This is often the case in object-oriented programming; for example, interfaces in Java have no operations that correspond to constructors. Hence the technique of using coercion functions seems to be fundamentally incomplete for proving behavioral subtyping.

One way to address this incompleteness would be to force users to give coercion functions along with specifications of all subtypes. However, this would prevent one from specifying some examples, such as those above. An alternative would be to use a more general notion of coercion relation that reflects at the formal level the nominal standard simulations of the algebraic completeness theorem, which are not in general functional. We leave this as future work.

7. Discussion

In this section we discuss some further aspects of our work.

7.1. Order-sorted algebras and correct behavioral subtyping.

We make precise the assertion, made at the beginning of Section 5, that the restriction to order-sorted models begs the question whether an equational specification is correctly behaviorally subtyped.

Define the class $\mathsf{Mod}^{\circ}_{\mathrm{NOM}}(E)$ of order-sorted transforms of nominal models of a set of equations, E, to be $\{\boldsymbol{A}^{\circ} : \boldsymbol{A} \in \mathsf{Mod}_{\mathrm{NOM}}(E)\}$.

Theorem 7.1. *An equational specification $\langle \Sigma, E\rangle$ is correctly behaviorally subtyped iff*

$$\mathsf{Mod}^{\circ}_{\mathrm{NOM}}(E) = \mathsf{Mod}_{\mathrm{ORD}}(E).$$

Proof. Assume $\langle \Sigma, E\rangle$ is correctly behaviorally subtyped. The inclusion $\mathsf{Mod}_{\mathrm{ORD}}(E) \subseteq \mathsf{Mod}_{\mathrm{NOM}}(E)$ holds in general, and, for any order-sorted Σ-algebra $\boldsymbol{A}$, trivially $\boldsymbol{A}^{\circ} = \boldsymbol{A}$. Thus $\mathsf{Mod}_{\mathrm{ORD}}(E) = \mathsf{Mod}^{\circ}_{\mathrm{ORD}}(E) \subseteq \mathsf{Mod}^{\circ}_{\mathrm{NOM}}(E)$. For the reverse inclusion, let $\boldsymbol{A}^{\circ} \in \mathsf{Mod}^{\circ}_{\mathrm{NOM}}(E)$, with $\boldsymbol{A} \in \mathsf{Mod}_{\mathrm{NOM}}(A)$. We must show that $\boldsymbol{A}^{\circ}$ is also a model of $\langle \Sigma, E\rangle$. For this purpose consider any $\forall H(t = t') \in E$, of type V that is necessarily visible, and let $\bar{a}$ be any H-environment of $\boldsymbol{A}^{\circ}$. Then $\bar{a}$ is a virtual H-environment of $\boldsymbol{A}$. Since $\langle \Sigma, E\rangle$ is correctly behaviorally subtyped by assumption, $\mathsf{Mod}_{\mathrm{NOM}}(E)$ is correctly behaviorally subtyped and hence there is a $\boldsymbol{B} \in \mathsf{Mod}_{\mathrm{NOM}}(E)$ such that $\boldsymbol{A}$ is nominally VIS-reducible to $\boldsymbol{B}$ (see Appendix A, Def. A.1). Thus there is a (dynamic) H-environment $\bar{b}$ of $\boldsymbol{B}$ such that $\bar{a}$ and $\bar{b}$ are behaviorally equivalent and hence $t^{\boldsymbol{A}}[\bar{a}] = t^{\boldsymbol{B}}[\bar{b}]$ and $t'^{\boldsymbol{A}}[\bar{a}] = t'^{\boldsymbol{B}}[\bar{b}]$ since $\forall H(t = t')$ is visible. But $\boldsymbol{B}$ is a model of $\langle \Sigma, E\rangle$ and $\bar{b}$ is a dynamic environment. So $t^{\boldsymbol{B}}[\bar{b}] = t'^{\boldsymbol{B}}[\bar{b}]$, and hence $t'^{\boldsymbol{A}}[\bar{a}] = t'^{\boldsymbol{A}^{\circ}}[\bar{a}]$. Thus $\boldsymbol{A}^{\circ} \in \mathsf{Mod}_{\mathrm{NOM}}(E)$, and hence $\mathsf{Mod}_{\mathrm{ORD}}(E) \subseteq \mathsf{Mod}^{\circ}_{\mathrm{NOM}}(E)$.

Assume now that $\mathsf{Mod}^{\circ}_{\mathrm{NOM}}(E) = \mathsf{Mod}_{\mathrm{ORD}}(E)$. Let $\boldsymbol{A} \in \mathsf{Mod}_{\mathrm{NOM}}(E)$. Then by assumption $\boldsymbol{A}^{\circ}$ is a model of $\langle \Sigma, E\rangle$. Since every H-environment $\bar{a}$ of $\boldsymbol{A}$ is automatically a dynamic H-environment of $\boldsymbol{A}^{\circ}$, $\boldsymbol{A}$ is nominally VIS-behaviorally reducible to $\boldsymbol{A}^{\circ}$ by the identity relation on environments. So $\mathsf{Mod}_{\mathrm{NOM}}(E)$, and hence $\langle \Sigma, E\rangle$, are correctly behaviorally subtyped (Def. A.1). $\square$

Corollary 7.2. *Let $\langle \Sigma, E\rangle$ be an equational specification. The following are equivalent.*

(i) *$\langle \Sigma, E\rangle$ is correctly behaviorally subtyped;*
(ii) *For every equation $\forall H(t = t')$ over Σ, $E \vDash_{\mathrm{NOM}} \forall H(t = t')$ iff $E \vDash_{\mathrm{ORD}} \forall H(t = t')$;*
(iii) *For every equation $\forall H(t = t')$ over Σ, $E \vdash_{\mathrm{NOM}} \forall H(t = t')$ iff $E \vdash_{\mathrm{SUB}} \forall H(t = t')$.*

Proof. The equivalence of (i) and (ii) is just a restatement of Theorem 7.1 in terms of nominal and order-sorted consequence (Def. 2.17). The equivalence of (ii) and (iii) follows immediately from the completeness and soundness theorem for equational logic with subtypes (Thm. 2.19). $\square$

If $\langle \Sigma, E\rangle$ is correctly behaviorally subtyped then adding the subsumption rule to the equational logic does not allow one to prove any more equations. Conversely, any equational specification extended by subsumption is automatically behaviorally correctly subtyped. This observation also has a proof-theoretic formulation: if the subsumption rule is admitted, then the identity system $\{c_{S,T}\}$, where $c_{S,T}[x_S\colon S] = x_S$ is a coercion system.

7.2. **A deductive system for behavioral equivalence.** As has been observed, the relation of behavioral provability $\vdash_{\text{BEH}}$ is not in general "inductive" in the sense of being recursively enumerable, but rather "coinductive". Its coinductive nature can be expressed in various ways: compare Cor. 3.8, Thm. 3.10, and Cor. 3.13.

It would be valuable to find useful criteria for determining when formal behavioral equivalence is actually inductive in the above sense; more specifically, when can it be given by an equational specification? Considerable attention has been paid to this problem in the literature [3, 7, 33, 34, 35].

The following theorem is one of the main results on behavioral consequence; see [15, Theorem 25] where it is viewed as a another form of coinduction. It can be obtained as a corollary of an analogous characterization theorem for the more general notion of behavioral equivalence between environments described in Appendix A; see [22, Corollary 3.13] and also [23, Corollary 4.14]. However it can also be obtained with little difficulty directly from the definition of behavioral equivalence.

Theorem 7.3. *For every Σ-algebra $\boldsymbol{A}$, the behavioral equivalence relation $\mathcal{BE}(\boldsymbol{A})$ is the largest* VIS*-identical homomorphic standard relation between $\boldsymbol{A}$ and itself.* □

In order to show that two objects a and a' of the same hidden type are behavioral equivalent it is sufficient to find some VIS-identical homomorphic standard relation on $\boldsymbol{A}$ that relates a and a'. Using this method directly to show that an equation $\forall H(t = t')$ is a behavioral consequence of an equational specification $\langle \Sigma, E\rangle$ is more of a challenge: one must find, for every model $\boldsymbol{A}$ of E and every H-environment $\bar{a}$, a VIS-identical homomorphic relation that relates $t^{\boldsymbol{A}}(\bar{a})$ and $t'^{\boldsymbol{A}}(\bar{a})$. Practically speaking, this can only be done by exhibiting a system of visible equations that explicitly define the homomorphic relation uniformly for all models and H-environments. Corollary 3.8 actually provides such a system, namely the local contexts. We call a set of local contexts that alone suffices to define behavioral equivalence a *cobasis* for an equational specification $\langle \Sigma, E\rangle$. (A closely related notion is introduced in [33] under the same name.) If the specification has a finite cobasis then behavioral consequence is clearly inductive. For example, it is easy to see from the discussion in Example 3.9 that in order to verify than an equation is a behavioral consequence of the specifications E_{cp} and E_{cpt} it suffices to consider only the two local contexts $\mathtt{x}(z)$ and $\mathtt{y}(z)$.

An analogue of Theorem 7.3 is presented in [22] for the more general notion of behavioral equivalence between environments. (See [23, Corollary 3.13] for a formulation of the result that more closely resembles that of Theorem 7.3.) There is a proof theory for this more general notion of behavioral equivalence that parallels that for behavioral equivalence presented in Section 3 and which leads to proof-theoretic characterizations similar to Corollary 3.8, Theorem 3.10, and Corollary 3.13. It is problematic as to how useful this is for the specification of correct behavioral subtyping.

8. Related Work

In regard to the material on behavioral equivalence in the context of equational specifications that we present here, the most closely related work in the literature is that Goguen *et al.* on *hidden algebra* [15, 16] (see also [14, 33, 34]). Much of the material in Sections 2 and 3 can be found in some form in at least one these papers. However our development is proof-theoretic while that of hidden algebra is mainly model-theoretic. The characterization of behavioral equivalence given in Theorem 7.3 can be found in [15, Theorem 25]. Corollary 3.8 is closely related to the notion a cobasis for an equational specification considered in [33]. Theorem 3.10 and Corollary 3.13 seem to be new.

The genesis for behavioral equivalence in hidden algebra seems to be Reichel [31]. The idea of behavioral equivalence, in the guise of logical equivalence, as the largest congruence satisfying a certain property can be traced back to the process of forming the Lindenbaum-Tarski algebra of the classical propositional calculus by Tarski [38]. A logical analog of Theorem 7.3 can be found in [36]. Subsequently these ideas formed the basis of abstract algebraic logic [4, 8, 12, 30]

The dichotomy between behavioral equivalence as a relation between objects and between environments seems to be novel, although elements of the latter (in the form of homomorphic generalized relations) can be found in [37] in the context of type theory. As mentioned above Theorem 7.3 is a consequence of a similar result for the environment version of behavioral equivalence in [22]. The latter is just one result in a general theory of behavior and realization that is developed in [22]. In [23] it is applied to obtain an algebraic completeness and soundness theorem for the notion of correct behavioral subtyping based on environment behavioral equivalence. Theorem 7.3 can be applied in a similar way to give an analogous completeness and soundness theorem for a more restricted notion of correct behavioral subtyping that is based on the behavioral equivalence of objects. The algebraic soundness theorem (Theorem 5.5) that serves as the principal lemma for the main result of the present paper (Theorem 5.6) is a consequence of both of these theorems, since object behavioral equivalence implies behavioral equivalence of environments in a natural sense. However, Theorem 7.1 and Corollary 7.2 suggest that behavioral equivalence of environments is the more natural of the two.

Techniques for proving correct object behavioral subtyping have been studied by several authors [2, 5, 9, 10, 21, 24, 25, 26]. While most of these authors have studied the soundness of their techniques, to the best of our knowledge none have studied their completeness.

In earlier work [21] we showed that the use of nominal standard simulations as described above is sound for correct behavioral subtyping based on environment behavioral equivalence. However, more recently we showed that this technique is complete only for term-generated specifications and for specifications that do not use multiple dispatch [23].

We have observed that behavioral equivalence can be viewed as a device for introducing the notion state transitions into the algebraic theory of ADTs. Pure state transition systems constitute an important alternative to algebras as a basis for the semantics of OO languages. Gordon and Rees [18] investigate behavioral equivalence and its formalization in the context of $\mathbf{Ob}_{1<:\mu}$, Abadi and Cardelli's single dispatch first-order calculus of objects with subtyping and with rewriting rules [1]. In this calculus data elements are identified with programs, i.e., expressions of the calculus, and logical equivalence (equality) is defined as reduction to the same normal form. Gordon and Rees define two programs to be *contextual equivalent* (their interpretation of behavioral equivalent) if, whenever they are placed in a larger boolean-valued program, the resulting programs either both converge or both diverge. *Bisimulation* is defined in terms of a transition system with the programs as states; here the fact that the calculus is single dispatch plays an essential role. Analogues of the algebraic completeness theorem for unary signatures (Thm. A.3) and the proof-theoretic soundness theorem (Thm. 5.6) are obtained, and the limitations of the method of bisimulation are pointed out; these turn out to be similar to the problems with coercion discussed above. One advantage of our work compared with Gordon and Rees's is that our theory is not limited to single dispatch languages. There seems to be no analogue in their work of the notion of a local context and the refinement of the process of verifying correct behavioral subtyping that it leads to.

9. Conclusions

Coercion can be an effective way of verifying that an equational specification is correctly behaviorally subtyped, provided there is a effective method of verifying behavioral consequence. We studied one such method, local contexts, that seem to work well in many cases. But there can be no general method of this kind because in general there is no effectively computable procedure for verifying behavioral consequence.

Coercion does not form a complete method for showing correct behavioral subtyping however since there is an example of an equational specification that is correctly behaviorally subtyped but has no coercion system. Moreover, the specification does use multiple dispatch, so the incompleteness does not result from the use of multiple dispatch. Furthermore, restriction to term-generated specifications does not give a completeness result for the coercion method.

Equational logic with subsumption cannot cannot be used for the purpose of verifying correct behavioral subtyping because its models are essentially all order-sorted and order-sorted classes of models are automatically correctly behaviorally subtyped.

Appendix A. Model Theory of Correct Behavioral Subtyping

The following is taken from sections 4 and 5 of [23].

Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be Σ-algebras with the same carrier sets for their visible types. Let $H = \langle x_1\colon T_1, \dots, x_n\colon T_n\rangle$ be a type context over Σ. Virtual H-environments $\bar{a}$ and $\bar{b}$ in $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively, are said to be VIS-*behaviorally equivalent* if, for every H-term $t[x_1\colon T_1, \dots, x_n\colon T_n]\colon V$ of visible type V, $t^{\boldsymbol{A}}[\bar{a}] = t^{\boldsymbol{B}}[\bar{b}]$.

$\boldsymbol{A}$ is *nominally* VIS-*behaviorally reducible to* $\boldsymbol{B}$ if, for each type context H and each virtual H-environment $\bar{a}$ in $\boldsymbol{A}$, there exists a dynamic H-environment $\bar{b}$ in $\boldsymbol{B}$ such that $\bar{a}$ and $\bar{b}$ are VIS-behaviorally equivalent.

Note that if $\boldsymbol{A}$ is order-sorted, then $\boldsymbol{A}$ is nominally VIS-behaviorally reducible to itself, since every H-environment is is dynamic.

Definition A.1 ([23])**.** Let $\langle \Sigma, E\rangle$ be an equational specification. A class $\mathcal{M}$ of VIS-categorical models of $\langle \Sigma, E\rangle$ is *correctly behaviorally subtyped* if every $\boldsymbol{A} \in \mathcal{M}$ is nominally VIS-behaviorally reducible to some $\boldsymbol{B} \in \mathcal{M}$. $\square$

Note that the models $\boldsymbol{A}$ and $\boldsymbol{B}$ in the above definition can be equal. The idea here is that the correct subtyping means that the objects of subtypes behave just like objects of supertypes, and that the behavior of a virtual environment of $\boldsymbol{A}$ is determined by the pattern of values it returns under all observations.

A.1. **Proof of the Algebraic Soundness Theorem.** The proof of this theorem requires the following lemma (whose proof is by induction).

Lemma A.2 ([22, 23])**.** *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be Σ-algebras and assume $\mathcal{R}$ is a nominal standard simulation of $\boldsymbol{B}$ by $\boldsymbol{A}$. Let T and $(S_1, \ldots, S_n) \leq (T_1, \ldots, T_n)$ be types. Then for all Σ-terms $t[x_1{:}\,T_1, \ldots, x_n{:}\,T_n]{:}\,T$,*

$$\langle a_1, \ldots, a_n \rangle \, \mathcal{R}_{T_1} \times \cdots \times \mathcal{R}_{T_n} \, \langle b_1, \ldots, b_n \rangle \quad \textit{implies} \quad t^{\boldsymbol{A}}[a_1, \ldots, a_n] \, \mathcal{R}_T \, t^{\boldsymbol{A}}[b_1, \ldots, b_n]. \quad \square$$

Recall that the algebraic soundness theorem itself reads as follows.

Theorem 5.5 (Algebraic Soundness Theorem [23])**.** *Let $\langle \Sigma, E \rangle$ be an equational specification, and let $\mathsf{Mod}_{\mathrm{NOM}}(E)$ be* VIS*-categorical. If for every $\boldsymbol{A} \in \mathsf{Mod}_{\mathrm{NOM}}(E)$ there exists a $\boldsymbol{B} \in \mathsf{Mod}_{\mathrm{NOM}}(E)$ and a nominal standard simulation of $\boldsymbol{B}$ by $\boldsymbol{A}$, then $\langle \Sigma, E \rangle$ is correctly behaviorally subtyped.*

Proof. Let $\boldsymbol{A}, \boldsymbol{B} \in \mathsf{Mod}_{\mathrm{NOM}}(E)$, and let $\mathcal{R} = \langle \mathcal{R}_T \subseteq A^{\circ}_T \times B^{\circ}_T : T \in \mathrm{TYPE} \rangle$ be a nominal standard simulation of $\boldsymbol{B}$ by $\boldsymbol{A}$. Let $H = \langle x_1{:}\,T_1, \ldots, x_n{:}\,T_n \rangle$ be a type context over Σ and let $\bar{a} = a_1, \ldots, a_n$ be a virtual H-environment in $\boldsymbol{A}$. Then $\bar{a} \in A_{S_1} \times \cdots \times A_{S_n}$ for some $S_1, \ldots, S_n \leq T_1, \ldots, T_n$. Since $\mathcal{R}$ is nominal, there exists a dynamic H-environment $\bar{b} = b_1, \ldots, b_n$ such that $\bar{a} \, \mathcal{R}_{T_1} \times \cdots \times \mathcal{R}_{T_n} \, \bar{b}$. We will show that $\bar{a}$ is VIS-behaviorally equivalent to $\bar{b}$. For this purpose let $t[x_1{:}\,T_1, \ldots, x_n{:}\,T_n]{:}\,V$ be an H-term of visible type V. By Lem. A.2

$$t^{\boldsymbol{A}}[\bar{a}] \, \mathcal{R}_V \, t^{\boldsymbol{B}}[\bar{b}].$$

Hence $t^{\boldsymbol{A}}[\bar{a}] = t^{\boldsymbol{B}}[\bar{b}]$, since $\mathcal{R}$ is VIS-identical.

Since $\bar{a}$ is VIS-behaviorally equivalent to $\bar{b}$, $\mathsf{Mod}_{\mathrm{NOM}}(E)$ is correctly behaviorally subtyped, and hence, by definition, $\langle \Sigma, E \rangle$ is correctly behaviorally subtyped. $\square$

A.2. **Algebraic Completeness for Unary Signatures.** For standard simulations, there is no algebraic completeness theorem that corresponds to the algebraic soundness theorem. A equational specification, call it $\langle \Sigma_1, E_1 \rangle$, is given in our previous work [23] that is correctly behaviorally subtyped but has a restricted model $\boldsymbol{A}_1$ such that $\boldsymbol{A}_1$ is not nominally behaviorally reducible to any restricted model of $\langle \Sigma_1, E_1 \rangle$, including itself. On the other hand, in the same previous work we show that no specification of this kind can exist if the signature in question has only unary methods. Thus the method of nominal standard simulation is complete for correct behavioral subtyping if only unary methods are allowed. We restate the theorem here.

Theorem A.3 (Algebraic Completeness Theorem for Unimethod Signatures, [23])**.** *Let Σ be signature with subtyping that has only unary methods. Let $\langle \Sigma, E \rangle$ be an equational specification such that $\mathsf{Mod}_{\mathrm{NOM}}(E)$ is* VIS*-categorical. If $\langle \Sigma, E \rangle$ is correctly behaviorally subtyped, then for every $\boldsymbol{A} \in \mathsf{Mod}_{\mathrm{NOM}}(E)$ there exists a $\boldsymbol{B} \in \mathsf{Mod}_{\mathrm{NOM}}(E)$ and a nominal standard simulation of $\boldsymbol{B}$ by $\boldsymbol{A}$.* $\square$

A similar result holds for equational specifications for which each restricted model is term-generated [22, 23].

References

[1] Martín Abadi and Luca Cardelli, *A Theory of Objects.* Monographs in Computer Science, Springer, New York, 1996.

[2] Pierre America, *Designing an Object-Oriented Programming Language with Behavioral Subtyping,* Foundations of Object-Oriented Languages, REX School/Workshop, Noordijkerhout, The Netherlands, May/June 1990 (J. W. de Bakker and W. P. de Roever and G. Rozenberg, eds.), Lecture Notes in Computer Science, vol. 489, Springer-Verlag, New York, 1991, 60–90.

[3] Michael Bidoit and Rolf Hennicker, *Behavioral Theories and the Proof of Behavioral Properties,* Theor. Comp. Sci. **165**, 1996, 38–53.

[4] Willem Blok and Don Pigozzi, *Algebraizable Logics.* Mem. Amer. Math. Soc., vol. 396, Amer. Math. Soc., Providence, 1989.

[5] Kim B. Bruce and Peter Wegner, *An Algebraic Model of Subtype and Inheritance,* Advances in Database Programming Languages (F. Bançilhon and P. Buneman, eds.), Addison-Wesley, Reading, Mass., 1990, 75–96.

[6] Samuel Buss and Grigore Roşu, *Incompleteness of Behavioral Logics.* In **Proceedings of Coalgebraic Methods in Computer Science (CMCS'00), Berlin, Germany, March 2000.** H. Reichel, ed., Electronic Notes in Theoretical Computer Science, vol. 33, Elsevier Science, 2000, 61–79.

[7] Andrea Corradini. Technical Report SEN-R9723, ISSN 1386-396X, CWI, 1997.

[8] Janusz Czelakowski, *Protoalgebraic Logics.* Kluwer Acad. Publ., Amsterdam, 2000.

[9] Krishna Kishore Dhara and Gary T. Leavens, *Weak Behavioral Subtyping for Types with Mutable Objects*, Mathematical Foundations of Programming Semantics, Eleventh Annual Conference (S. Brookes, M. Main, A. Melton and M. Mislove, eds.), Electronic Notes in Theoretical Computer Science, vol. 1, Elsevier, 1995.

[10] Krishna Kishore Dhara and Gary T. Leavens, *Forcing Behavioral Subtyping Through Specification Inheritance,* Proceedings of the 18th International Conference on Software Engineering, Berlin, Germany, IEEE Computer Society Press, 1996, 258–267

[11] Hartmut Ehrig and Bernd Mahr, *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics.* EATCS Monographs on Theoretical Computer Science, Springer-Verlag, New York, N.Y., 1985.

[12] Josep Maria Font and Ramon Jansana, **A general algebraic semantics for sentential logics.** Lecture Notes in Logic, vol. 7, Springer-Verlag, 1996.

[13] Joseph A. Goguen, *Order Sorted Algebras,* Technical Report 14, UCLA Computer Science Department (1978), Semantics and Theory of Computation Series.

[14] Joseph Goguen and Grigore Roşu, *Hiding More of Hidden Algebra.* In **FM'99 – Formal Methods.** Lecture Notes in Computer Sciences, vol. 1709, Proceedings of World Congress on Formal Methods, Toulouse, France, Springer-Verlag, 1999, 1704–1719.

[15] Joseph A. Goguen and Grant Malcolm, *Hidden Coinduction: Behavioral Correctness Proofs for Objects,* Math. Struct. in Comp. Science, 1999.

[16] Joseph A. Goguen and Grant Malcolm, *A Hidden Agenda,* Theor. Comp. Sci., 1999

[17] Joseph Goguen and José Meseguer, *Order-Sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations,* Theor. Comp. Sci. **105**(1987), 217–273

[18] Andrew D. Gordon and Gareth D. Rees, *Bisimilarity for a First-Order Calculus of Objects with Subtyping*, POPL '96: The 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Conference Record, 387–395

[19] Bart Jacobs, *Objects and Classes Co-Algebraically.* In B. Freitag, C.B. Jones, C. Lengauer, and H-J. Schek, eds., **Object-Orientation and Parallelism and Persistence.** Kluwer Acad. Publ., 1996, 83–103.

[20] Bart Jacobs and Jan Rutten, *A Tutorial on (C0)Algebras and (Co)Induction,* EATCS Bulletin. **62**(1997), 222–259.

[21] Gary T. Leavens and Don Pigozzi, *Typed Homomorphic Relations Extended with Subtypes,* Mathematical Foundations of Programming Semantics '91 (S. Brookes, ed.), Lecture Notes in Computer Science, vol. 598, Springer-Verlag, New York, 1992, 144–167

[22] Gary T. Leavens and Don Pigozzi, *The Behavior-Realization Adjunction and Generalized Homomorphic Relations,* Theor. Comp. Sci. **177**(1997), 183–216.

[23] Gary T. Leavens and Don Pigozzi, *A Complete Algebraic Characterization of Behavioral Subtyping,* Acta Informatica. **36**(2000), 617–663.

[24] Gary T. Leavens and William E. Weihl, *Reasoning about Object-oriented Programs that Use Subtypes (extended abstract),* ECOOP/OOPSLA '90 Proceedings (M. Meyrowitz, ed.), ACM SIGPLAN Notices, vol. 25, ACM, October, 1990, 212–223.

[25] Gary T. Leavens and William E. Weihl, *Specification and Verification of Object-Oriented Programs Using Supertype Abstraction,* Acta Informatica **32**(1995), no. 8, 705–778.

[26] Barbara Liskov and Jeannette Wing, *A Behavioral Notion of Subtyping,* ACM Transactions on Programming Languages and Systems **16**(1994), 1811–1841.

[27] Eliot Mendelson, *Introduction to Mathematical Logic* Wadsworth and Brooks/Cole, 1987.

[28] Narciso Marti-Oliet, and Jose Meseguer. *Inclusions and Subtypes.* Technical Report SRI-CLS-90-16, Computer Science Laboratory, SRI International (December, 1990).

[29] Tobias Nipkow, *Non-deterministic Data Types: Models and Implementations,* Acta Informatica **22**(1986), 629–661.

[30] Don Pigozzi, *Abstract Algebraic Logic,* Encyclopaedia of Mathematics. Supplement III. Kluwer Acad. Publ., to appear.

[31] Horst Reichel, *Behavioral Equivalence–a Unifying Concept for Initial and Final Specifications.* In **Proceedings, Third Hungarian Computer Science Conference.** Akademiai Kiado, Budapest, 1981.

[32] John C. Reynolds, *Using Category Theory to Design Implicit Conversions and Generic Operators,* Semantics-Directed Compiler Generation, Proceedings of a Workshop, Aarhus, Denmark (N. D. Jones, ed.), Lecture Notes in Computer Science, vol. 94, Springer-Verlag, New York, 1980, 211–258.

[33] Grigore Roşu and Joseph Goguen, *Hidden Congruence Deduction.* In **Automated Deduction in Classical and Non-Classical Logics.** R. Caferra and G. Alzer, eds., Lecture Notes in Artificial Intelligence, vol. 1761, Springer-Verlag, 2000, 252–267.

[34] Grigore Roşu and Joseph Goguen, *Circular Coinduction.* In **Circular Coinduction. International Joint Conference on Automated Reasoning (IJCAR'01)**, 2001.

[35] J. J. M. M. Rutten, *Universal Coalgebra: a Theory of Systems.* Technical Report CC-R9652, 1996.

[36] D. J. Shoesmith and T. J. Smiley, **Multiple-Conclusion Logic** Cambridge University Press, Cambridge, 1978.

[37] K. Sieber, *Reasoning about Sequential Functions via Logical Relations,* In **Applications of Categories in Computer Science. Proceedings of the LMS Symposium, Durham 1991.** M. P. Fourman, P. T. Johnstone, and A. M. Pitts, eds., London Mathematical Society Lecture Note Series, vol. 177, Cambridge University Press, Cambridge, 1992, 259–269.

[38] Alfred Tarski, *Grundzüge der Systemenkalküls. Erster Teil,* Fund. Math. **25**(1935), 503–526. English translation in Alfred Tarski, **Logic, Semantics, Metamathematics. Papers from 1923 to 1938,** Second, revised edition. J. Corcoran, ed. Hackett Pub. Co., Indianapolis, Indiana, 1983.

Department of Computer Science, Iowa State University, Ames, IA 50011-1041, USA,
E-mail address: `leavens@cs.iastate.edu`

Department of Mathematics, Iowa State University, Ames, IA 50011, USA
E-mail address: `dpigozzi@iastate.edu`