Hidden Congruent Deduction

Grigore Roşu[1] and Joseph Goguen
Department of Computer Science & Engineering
University of California at San Diego

1 Introduction

Cleverly designed software often fails to satisfy its requirements strictly, but instead satisfies them *behaviorally*, in the sense that they *appear* to be satisfied under every experiment that can be performed on the system. A good example is the traditional implementation of sets by lists, where union as implemented by append fails to strictly satisfy basic laws like commutativity and idempotency, but does satisfy them behaviorally. It is becoming increasingly clear that behavioral specification is more appropriate to software engineering than traditional approaches that rely on strict satisfaction of axioms, and it is therefore becoming increasingly important to develop powerful techniques for behavioral verification. This paper presents some techniques of this kind in the area called *hidden algebra*, clustered around the central notion of *coinduction*. We believe hidden algebra is the natural next step in the evolution of algebraic semantics and its first order proof technology. Hidden algebra originated in [7], and was developed further in [8, 10, 3, 12, 5] among other places; the most comprehensive survey currently available is [12].

Proofs by coinduction are *dual* to proofs by induction, in that the former are based on a largest congruence, and the latter on a smallest subalgebra (e.g., see [12]). Inductive proofs require choosing a set of constructors, often called a basis; the dual notion is *cobasis*, and as with bases for induction, the right choice can result in a dramatically simplified proof. An interesting complication is that the best choice may not be part of the given signature, but rather contain operations that can be defined over it.

An important recent development is the notion of *congruent* operations (these were called "coherent"[2] in [5, 4], where they were introduced), which considerably expands the applicability of hidden algebra and coinduction by allowing operations that have more than one hidden argument, thus going well beyond what is possible in coalgebra (e.g., see [14, 17]).

The most significant contributions of this paper are a slightly more general notion of congruence, the notion of cobasis, some rules of deduction for hidden algebra, and an easy to check criterion for operations to be congruent; the first two items build on work in [4]. There is also a hidden version of the so called "theorem of constants," and Theorem 25, which says congruent operations can be added or subtracted to the set of behavioral operations as convenient, still yielding an equivalent specification. The main conceptual advance of this paper is to extend all main concepts and results of hidden algebra to encompass operations with more than one hidden argument.

Because of space limitations, we must omit some proofs, and assume familiarity with many sorted first order equational logic, including the notions of many sorted signature, algebra, homomorphism, term, equation, and satisfaction; e.g., see [12, 11]. We let $f;g$ denote the composition of $f\colon A \to B$ with $g\colon B \to C$. Recall that $T_\Sigma(X)$ denotes the Σ-algebra of all Σ-terms with variables from X.

2 Hidden Algebra

Definition 1 A **hidden signature** is a triple (Ψ, D, Σ), often denoted just Σ, where

[1] On leave from Fundamentals of Computer Science, Faculty of Mathematics, University of Bucharest, Romania.

[2] We feel that the word "congruent" better describes the role that these operations actually play.

- Ψ is a V-sorted signature and D is a Ψ-algebra, called the **data algebra**,
- Σ is a $(V \cup H)$-sorted signature extending Ψ and such that each operation in Σ with both its arguments and its result visible lies in Ψ, and
- V and H are disjoint sets, called **visible sorts** and **hidden sorts**, respectively.

For technical reasons, we assume that for every element d in the data algebra D there exists exactly one constant in Ψ, also denoted d.

The operations in Σ with one hidden argument and visible result are called **attributes**, those with one hidden argument and hidden result are called **methods**, and those with visible arguments and hidden result are called **hidden constants**. A **hidden subsignature of** Σ is a hidden signature (Ψ, D, Γ) with $\Gamma \subseteq \Sigma$. A **behavioral** (or **hidden**) Σ-**specification** or -**theory** is a triple (Σ, Γ, E), where Σ is a hidden signature, Γ is a hidden subsignature of Σ, and E is a set of Σ-equations. The operations in $\Gamma - \Psi$ are called **behavioral**.

A **hidden Σ-algebra** is a many sorted Σ-algebra A such that $A|_\Psi = D$. □

The behavioral operations in a specification are the ones that can be used in experiments, i.e., they define behavioral equivalence. The results of experiments lie in the data algebra. Philosophically, it seems that an assertion that an operation is behavioral should be a kind of sentence; from this view, it is an accident that the set of such sentences forms a signature, as in the "extended signatures" of [4].

Example 2 Below is a behavioral specification for sets, written in the CafeOBJ language [5] (however, the CafeOBJ parser does not accept it, because behavioral operations with more than one hidden argument are currently prohibited):

```
mod* SET1 { *[ Set ]* pr(NAT)
bop _in_ : Nat Set -> Bool  ** attribute
op empty : -> Set           ** hidden const
bop add   : Nat Set -> Set  ** method
bop _U_   : Set Set -> Set
bop _&_   : Set Set -> Set
bop neg   : Set -> Set      ** method
vars N N' : Nat  vars X X' : Set
eq N in empty = false .
eq N in add(N',X) = (N == N') or (N in X) .
eq N in (X U X')  = (N in X)  or (N in X') .
eq N in (X & X')  = (N in X) and (N in X') .
eq N in neg(X)    = not (N in X) . }
```

Here "`*[Set]*`" declares `Set` to be a hidden sort, "`bop`" indicates a behavioral operation, and "`pr(NAT)`" indicates that the module `NAT` of natural numbers is imported in "protecting" mode, i.e., the naturals are not compromised by the new declarations and equations. The constant `empty` is the only non-behavioral operation, and `neg` is complement with respect to the set of all natural numbers. We will see later that this spec is equivalent to another having `in` as its only behavioral operation. □

Definition 3 Given a hidden signature Γ, an (appropriate) Γ-**context** of sort s is a visible term in $T_\Gamma(\{z\} \cup Z)$ having exactly one occurrence of a special variable[3] z of sort s, where Z is an infinite set of special variables. We let $C_\Gamma[z : s]$ denote the set of all Γ-contexts of sort s, and $var(c)$ the finite set of variables of c, except z. Given a hidden signature Σ, a hidden subsignature Γ of Σ, and a Σ-algebra A, each Γ-context c generates a map $A_c \colon A_s \times A^{var(c)} \to D$ defined by $A_c(a, \theta) = a^*_\theta(c)$, where a^*_θ is the unique extension of the map (denoted a_θ) that takes z to a and each $z' \in var(c)$ to $\theta(z')$. The equivalence

[3] "Special variables" are assumed to be different from any other variable in a given situation.

given by $a \equiv_{\Sigma}^{\Gamma} a'$ iff $A_c(a,\theta) = A_c(a',\theta)$ for all Γ-contexts c and all maps $\theta\colon var(c) \to A$ is called **Γ-behavioral equivalence on** A. Given any equivalence $\sim$ on A, an operation σ in $\Sigma_{s_1...s_n,s}$ is **congruent for** $\sim$ iff $A_\sigma(a_1,...,a_n) \sim A_\sigma(a'_1,...,a'_n)$ whenever $a_i \sim a'_i$ for $i = 1...n$. An operation σ is **Γ-behaviorally congruent for** A iff σ is congruent for $\equiv_{\Sigma}^{\Gamma}$; we will often say just "congruent" instead of "behaviorally congruent"[4]. A **hidden Γ-congruence on** A is an equivalence on A which is the identity on visible sorts and such that each operation in Γ is congruent for it. □

The following is the basis for several of our results, especially coinduction; it generalizes a similar result in [12] to operations that can have more than one hidden argument.

Theorem 4 Given a hidden subsignature Γ of Σ and a hidden Σ-algebra A, then Γ-behavioral equivalence is the largest hidden Γ-congruence on A.

Proof: We first show that $\equiv_{\Sigma}^{\Gamma}$ is a hidden Γ-congruence. It is straightforward that it is the identity on visible sorts because we can take the context $c = z$. Now let $\sigma : s_1...s_n \to s$ be any operation in Γ, let $a_1 \equiv_{\Sigma,s_1}^{\Gamma} a'_1$, ..., $a_n \equiv_{\Sigma,s_n}^{\Gamma} a'_n$, let c be any Γ-context of sort s, and let $\theta\colon var(c) \to A$ be any map. Let $z_1,...,z_n$ be variables in Z distinct from z and from those in $var(c)$, and take the Γ-contexts $c_j = c[\sigma(z_1,...,z_{j-1},z,z_{j+1},...,z_n)]$ of sorts s_j and the maps $\theta_j\colon \{z_1,...,z_{j-1},z,z_{j+1},...,z_n\} \cup var(c) \to A$ to be defined by $\theta_j(z_i) = a'_i$ for $1 \le i < j$, $\theta_j(z_i) = a_i$ for $j < i \le n$, and $\theta_j(z') = \theta(z')$ for $z' \in var(c)$, for $1 \le j \le n$. Notice that $A_c(A_\sigma(a_1,...,a_n)) = A_{c_1}(a_1,\theta_1)$, that $A_{c_j}(a_j,\theta_j) = A_{c_j}(a'_j,\theta_j)$ for all $1 \le j \le n$ because $a_j \equiv_{\Sigma,s_j}^{\Gamma} a'_j$ and c_j and θ_j are appropriate Γ-contexts and maps, that $A_{c_j}(a'_j,\theta_j) = A_{c_{j+1}}(a_{j+1},\theta_{j+1})$ for all $1 \le j < n$, and that $A_{c_n}(a'_n,\theta_n) = A_c(A_\sigma(a'_1,...,a'_n),\theta)$. Then $A_c(A_\sigma(a_1,...,a_n),\theta) = A_c(A_\sigma(a'_1,...,a'_n),\theta)$, that is, $A_\sigma(a_1,...,a_n) \equiv_{\Sigma,s}^{\Gamma} A_\sigma(a'_1,...,a'_n)$. Therefore σ is Γ-behaviorally congruent for A, and so $\equiv_{\Sigma}^{\Gamma}$ is a hidden Γ-congruence.

Now let $\sim$ be another hidden Γ-congruence on A and let $a \sim_s a'$. Because each operation in Γ is congruent for $\sim$, $A_c(a,\theta) \sim A_c(a',\theta)$ for any Γ-context c of sort s and any map $\theta\colon var(c) \to A$, and because $\sim$ is the identity on visible sorts, $A_c(a,\theta) = A_c(a',\theta)$. Therefore $a \equiv_{\Sigma,s}^{\Gamma} a'$, that is, $\sim \subseteq \equiv_{\Sigma}^{\Gamma}$. □

Definition 5 A hidden Σ-algebra A **Γ-behaviorally satisfies** a conditional Σ-equation $e = (\forall X)\ t = t'$ **if** $t_1 = t'_1, ..., t_n = t'_n$ iff for each $\theta\colon X \to A$, if $\theta(t_i) \equiv_{\Sigma}^{\Gamma} \theta(t'_i)$ for $i = 1,...,n$, then $\theta(t) \equiv_{\Sigma}^{\Gamma} \theta(t')$; in this case we write $A \models_{\Sigma}^{\Gamma} e$. If E is a set of Σ-equations, we write $A \models_{\Sigma}^{\Gamma} E$ if A Γ-behaviorally satisfies each equation in E. When Σ and Γ are clear from context, we may write $\equiv$ and $\models$ instead of $\equiv_{\Sigma}^{\Gamma}$ and $\models_{\Sigma}^{\Gamma}$, respectively. We say that A **behaviorally satisfies** (or **is a model of**) a behavioral specification $\mathcal{B} = (\Sigma,\Gamma,E)$ iff $A \models_{\Sigma}^{\Gamma} E$, and in this case we write $A \models \mathcal{B}$; we write $\mathcal{B} \models e$ whenever $A \models \mathcal{B}$ implies $A \models_{\Sigma}^{\Gamma} e$. An operation $\sigma \in \Sigma$ is **behaviorally congruent for** $\mathcal{B}$ iff σ is behaviorally congruent for every $A \models \mathcal{B}$. □

Example 6 Let `SET2` be `SET1` without the operation `neg` and the last equation. Then one model of `SET2` is finite lists of natural numbers, with `in` as membership, `empty` the empty list, `add` placing a number at the front of a list, `_U_` appending two lists, and `_&_` giving a list containing each element in the first list that also appears in the second. Notice that multiple occurrences of natural numbers are allowed in the "sets" of this model. Two lists are behaviorally equivalent iff they contain exactly the same natural numbers, without regard to order or number of occurrences. □

Fact 7 If $\mathcal{B} = (\Sigma,\Gamma,E)$ is a behavioral specification, then all operations in Γ and all hidden constants are behaviorally congruent for $\mathcal{B}$. □

[4]A similar notion has been given by Padawitz [16].

Example 8 All operations in `SET1` in Example 2 are congruent. Moreover, we will show that they are going to be congruent even if `in` is the only behavioral operation. □

The following reduces behavioral congruence to behavioral satisfaction of a certain equation, which further underlines the assertional character of this property.

Proposition 9 Given a behavioral specification $\mathcal{B} = (\Sigma, \Gamma, E)$ and an operation $\sigma \in \Sigma_{v_1...v_m h_1...h_k, s}$, let e_σ be the conditional Σ-equation $(\forall Y, x_1, x'_1, ..., x_k, x'_k)\ \sigma(Y, x_1, ..., x_k) = \sigma(Y, x'_1, ..., x'_k)$ **if** $x_1 = x'_1, ..., x_k = x'_k$, where $Y = \{y_1 : v_1, ..., y_m : v_m\}$. Then

1. σ is Γ-behaviorally congruent for a hidden Σ-algebra A iff $A \models^{\Gamma}_{\Sigma} e_\sigma$ and
2. σ is behaviorally congruent for $\mathcal{B}$ iff $\mathcal{B} \models e_\sigma$. □

The next result supports the elimination of hidden universal quantifiers in proofs.

Theorem 10 Theorem of Hidden Constants: If $\mathcal{B} = (\Sigma, \Gamma, E)$ is a behavioral specification, e is the Σ-equation $(\forall Y, X)\ t = t'$ **if** $t_1 = t'_1, ..., t_n = t'_n$, and e_X is the $(\Sigma \cup X)$-equation $(\forall Y)\ t = t'$ **if** $t_1 = t'_1, ..., t_n = t'_n$, where $\Sigma \cup X$ is the hidden signature obtained from Σ adding the variables in X as hidden constants, then $\mathcal{B} \models e$ iff $\mathcal{B}_X \models e_X$, where $\mathcal{B}_X = (\Sigma \cup X, \Gamma, E)$.

Proof: Suppose $\mathcal{B} \models e$, let A' be a $(\Sigma \cup X)$-algebra such that $A' \models \mathcal{B}_X$, and let A be the Σ-algebra $A|_\Sigma$. Notice that the Γ-behavioral equivalences on A and A' coincide, and that $A \models^{\Gamma}_{\Sigma} E$. Let $\theta \colon Y \to A'$ be such that $\theta^*(t_i) = \theta^*(t'_i)$ for $i = 1...n$, and let $\tau \colon Y \cup X \to A'$ be defined by $\tau(y) = \theta(y)$ for all $y \in Y$, and $\tau(x) = A'_x$ for all $x \in X$. Notice that $\tau^*(t_i) = \theta^*(t_i) = \theta^*(t'_i) = \tau^*(t'_i)$ for $i = 1...n$, and $A \models^{\Gamma}_{\Sigma} e$ since $A \models^{\Gamma}_{\Sigma} E$. Therefore $\tau^*(t) = \tau^*(t')$, so that $\theta^*(t) = \theta^*(t')$. Consequently, $A' \models^{\Gamma}_{\Sigma \cup X} e_X$, so that $\mathcal{B}_X \models e_X$.

Conversely, suppose $\mathcal{B}_X \models e_X$, let A be a Σ-algebra with $A \models \mathcal{B}$, and let $\tau \colon Y \cup X \to A$ be such that $\tau^*(t_i) = \tau^*(t'_i)$ for $i = 1...n$. Let A' be the $(\Sigma \cup X)$-algebra with the same carriers as A, and the same interpretations of operations in Σ, but with $A'_x = \tau(x)$ for each x in X. Notice that the Γ-behavioral equivalences on A and A' coincide. Also notice that $A' \models^{\Gamma}_{\Sigma \cup X} E$, so that $A' \models^{\Gamma}_{\Sigma \cup X} e_X$. Let $\theta \colon Y \to A'$ be the map defined by $\theta(y) = \tau(y)$ for each $y \in Y$. It is straightforward that $\theta^*(t_i) = \tau^*(t_i) = \tau^*(t'_i) = \theta^*(t'_i)$ for $i = 1...n$, so that $\theta^*(t) = \theta^*(t')$, that is, $\tau^*(t) = \tau^*(t')$. Therefore $A \models^{\Gamma}_{\Sigma} e$, so that $\mathcal{B} \models e$. □

The following justifies implication elimination for conditional hidden equations:

Fact 11 Given behavioral specification $\mathcal{B} = (\Sigma, \Gamma, E)$ and $t_1, t'_1, ..., t_n, t'_n$ ground hidden terms, let E' be $E \cup \{(\forall \emptyset)\ t_1 = t'_1, ..., (\forall \emptyset)\ t_n = t'_n\}$, and let $\mathcal{B}'$ be the behavioral specification (Σ, Γ, E'). Then $\mathcal{B}' \models (\forall X)\ t = t'$ iff $\mathcal{B} \models (\forall X)\ t = t'$ **if** $t_1 = t'_1, ..., t_n = t'_n$. □

3 Rules of Inference

This section introduces and justifies our rules for hidden congruent deduction. $\mathcal{B} = (\Sigma, \Gamma, E)$ is a fixed hidden specification throughout. The following shows soundness.

Proposition 12 The following hold:

1. $\mathcal{B} \models (\forall X)\ t = t$.
2. $\mathcal{B} \models (\forall X)\ t = t'$ implies $\mathcal{B} \models (\forall X)\ t' = t$.
3. $\mathcal{B} \models (\forall X)\ t = t'$ and $\mathcal{B} \models (\forall X)\ t' = t''$ imply $\mathcal{B} \models (\forall X)\ t = t''$.
4. If $\mathcal{B} \models (\forall Y)\ t = t'$ **if** $t_1 = t'_1, ..., t_n = t'_n$ and $\theta \colon Y \to T_\Sigma(X)$ is a substitution such that
   $\mathcal{B} \models (\forall X)\ \theta^*(t_i) = \theta^*(t'_i)$ for $i = 1...n$, then $\mathcal{B} \models (\forall X)\ \theta^*(t) = \theta^*(t')$.
5. If $\sigma \in \Sigma_{s_1...s_n, s}$ is a congruent operation for $\mathcal{B}$ and $t_i, t'_i \in T_{\Sigma, s_i}(X)$ for $i = 1...n$ such that
   $\mathcal{B} \models (\forall X)\ t_i = t'_i$ for $i = 1...n$, then $\mathcal{B} \models (\forall X)\ \sigma(t_1, ..., t_n) = \sigma(t'_1, ..., t'_n)$.

□

Substituting equal terms into a term is not always sound for behavioral satisfaction, because 5 above is not valid for non-congruent operations; the rules below take account of this fact. Let us define $\Vdash$ by $\mathcal{B} \Vdash (\forall X)\ t = t'$ iff $(\forall X)\ t = t'$ is derivable from $\mathcal{B}$ using (1)–(5) below.

(1) Reflexivity: $(\forall X)\ t = t$ is derivable,

(2) Symmetry: $(\forall X)\ t = t'$ derivable implies $(\forall X)\ t' = t$ derivable,

(3) Transitivity: $(\forall X)\ t = t'$ and $(\forall X)\ t' = t''$ derivable imply $(\forall X)\ t = t''$ derivable,

(4) Substitution: Given $(\forall Y)\ t = t'$ **if** $t_1 = t'_1, ..., t_n = t'_n$ in E and $\theta\colon Y \to T_\Sigma(X)$ such that $(\forall X)\ \theta(t_i) = \theta(t'_i)$ for $i = 1...n$ are derivable, then $(\forall X)\ \theta(t) = \theta(t')$ is derivable,

(5) Congruence: If $\sigma \in \Sigma_{s_1...s_n,s}$ is a congruent operation and $(\forall X)\ t_i = t'_i$ are derivable for $i = 1...n$, then $(\forall X)\ \sigma(t_1, ..., t_n) = \sigma(t'_1, ..., t'_n)$ is derivable.

Rule (5) is not fully syntactic, because the notion of congruent operation is semantic. But Fact 7 tells us that all visible and all behavioral operations, as well as all hidden constants are congruent, so we already have many cases where (5) can be applied. Later we will see how other operations can be shown congruent; this is important because our inference system becomes more powerful with each new operation proved congruent. The following result expresses soundness of these rules with respect to both equational and behavioral satisfaction.

Proposition 13 If $\mathcal{B} \Vdash (\forall X)\ t = t'$ then $E \models_\Sigma (\forall X)\ t = t'$ and $\mathcal{B} \models\!\!\equiv (\forall X)\ t = t'$. If all operations are behaviorally congruent, then equational reasoning is sound for the behavioral satisfaction. $\Box$

The rules (1)–(5) above differ from those in [4] in allowing both congruent and non-congruent operations; moreover, CafeOBJ's behavioral rewriting [5] is a special case in the same way that standard rewriting is a special case of equational deduction. Unlike equational deduction, these rules are not complete for behavioral satisfaction; but they do seem to provide proofs for most cases of interest.

3.1 Coinduction and Cobases

In this subsection, we assume $\mathcal{B}' = (\Sigma', \Gamma', E')$ is a **conservative extension** of $\mathcal{B}$, i.e., Σ is a hidden subsignature of Σ' and for every model A of $\mathcal{B}$ there exists a model A' of $\mathcal{B}'$ such that $A'|_\Sigma = A$. Also let Δ be a hidden subsignature of Σ'.

Definition 14 Let $T(\Gamma', \Delta; z\!:\!h, Z)$ be the indexed set of all $(\Gamma' \cup \Delta)$-terms γ with variables in $\{z\} \cup Z$, such that each subterm of γ rooted in any operation δ in Δ has exactly one occurrence of z which is an argument of δ, and these are the only occurrences of z in γ. We let $var(\gamma)$ denote the set of variables different from z of γ. Then Δ is a **cobasis** for $\mathcal{B}$ iff for any Γ-context c over z of hidden sort h there is some γ in $T(\Gamma', \Delta; z\!:\!h, var(c))$ such that $\mathcal{B}' \models\!\!\equiv (\forall z, var(c))\ c = \gamma$, and Δ is **context complete** for $\mathcal{B}$ iff for any Γ-context c over z of sort h there is some γ in $T_\Delta(\{z\} \cup var(c))$ such that $\mathcal{B}' \models\!\!\equiv (\forall z, var(c))\ c = \gamma$. $\Box$

Often $\mathcal{B} = \mathcal{B}'$ and $\Delta = \Gamma$, and in this case Δ is both context complete and a cobasis for $\mathcal{B}$. The following justifies coinduction:

Lemma 15 If Δ is a cobasis for $\mathcal{B}$ and $\mathcal{B}' \models\!\!\equiv (\forall Z_j, X)\ \delta(Z_j, t) = \delta(Z_j, t')$ for all appropriate (in the sense that the sort of t and t' is s_j) $\delta\colon s_1...s_n \to s$ in Δ and for $j = 1, ..., n$, where Z_j is the set of variables $\{z_1\!:\!s_1, ..., z_{j-1}\!:\!s_{j-1}, z_{j+1}\!:\!s_{j+1}, ..., z_n\!:\!s_n\}$ and $\delta(Z_j, t)$ is the term $\delta(z_1, ..., z_{j-1}, t, z_{j+1}, ..., z_n)$, then $\mathcal{B} \models\!\!\equiv (\forall X)\ t = t'$.

Proof: We first prove by structural induction that $\mathcal{B}' \models\!\!\equiv (\forall Z, X)\ \gamma[t] = \gamma[t']$ for all γ in $T(\Gamma', \Delta; z : h, Z)$. If $\gamma = \sigma(\gamma_1, ..., \gamma_n)$ with $\sigma \in \Gamma'$ and $\gamma_1, ..., \gamma_n$ are terms in $T(\Gamma', \Delta; z\!:\!h, Z)$ such that $\mathcal{B}' \models\!\!\equiv (\forall Z, X)\ \gamma_i[t] = \gamma_i[t']$ for $i = 1...n$, then by 5 of Proposition 12, $\mathcal{B}' \models\!\!\equiv (\forall Z, X)\ \sigma(\gamma_1[t], ..., \gamma_n[t]) = \sigma(\gamma_1[t'], ..., \gamma_n[t'])$. If $\gamma = \delta(\gamma_1, ..., \gamma_{j-1}, z, \gamma_{j+1}, ..., \gamma_n)$

with $\delta : s_1...s_n \to s$ in Δ and $\gamma_1, ..., \gamma_{j-1}, \gamma_{j+1}, ..., \gamma_n \in T_{\Gamma'}(Z)$, then by 4 of Proposition 12, $\mathcal{B}' \models (\forall Z, X)\ \delta(\gamma_1, ..., \gamma_{j-1}, t, \gamma_{j+1}, ..., \gamma_n) = \delta(\gamma_1, ..., \gamma_{j-1}, t', \gamma_{j+1}, ..., \gamma_n)$. Thus $\mathcal{B}' \models (\forall Z, X)\ \gamma[t] = \gamma[t']$.

Let c be a Γ-context for t and t' over z, and let γ be a term in $T(\Gamma', \Delta; z\!:\!h, var(c))$ such that $\mathcal{B}' \models (\forall z, var(c))\ c = \gamma$. By 4 of Proposition 12, $\mathcal{B}' \models (\forall\, var(c), X)\ c[t] = \gamma[t]$ and $\mathcal{B}' \models (\forall\, var(c), X)\ c[t'] = \gamma[t']$, so by 3 of Proposition 12, $\mathcal{B}' \models (\forall\, var(c), X)\ c[t] = c[t']$. Let A be any hidden Σ-algebra such that $A \models \mathcal{B}$ and let A' be a hidden Σ'-algebra such that $A' \models \mathcal{B}'$ and $A'|_\Sigma A$. Then $A' \models (\forall\, var(c), X)\ c[t] = c[t']$, i.e., $A \models (\forall\, var(c), X)\ c[t] = c[t']$. Because c was arbitrary, $A \models (\forall X)\ t = t'$, and thus $\mathcal{B} \models (\forall X)\ t = t'$. □

(6) Δ-Coinduction: Given $t, t' \in T_\Sigma(X)$ such that $(\forall Z_j, X)\ \delta(Z_j, t) = \delta(Z_j, t')$ is derivable from $\mathcal{B}'$ for all appropriate $\delta : s_1...s_n \to s$ in Δ and for $j = 1, ..., n$, then $(\forall X)\ t = t'$ is derivable from $\mathcal{B}$.

Notice that equations previously proved by coinduction can be used in another such proof.

Proposition 16 Define $\Vdash_\Delta$ by $\mathcal{B} \Vdash_\Delta (\forall X)\ t = t'$ iff $(\forall X)\ t = t'$ is derivable from $\mathcal{B}$ under rules (1)–(6). Then $\Vdash_\Delta$ is sound for behavioral satisfaction if Δ is a cobasis for $\mathcal{B}$. □

Notice that $\Vdash_\Delta$ is not necessarily sound with respect to equational satisfaction. The special case of Δ-coinduction where Δ consists of all the attributes is called **attribute coinduction**. The special case of attribute coinduction is implemented in Kumo [9], and we will soon implement the coinduction rule (6) in its general form.

Definition 17 Given a (not necessary hidden) signature Σ, a **derived operation** $\gamma : s_1...s_n \to s$ of Σ is a term in $T_{\Sigma,s}(\{z_1, ..., z_n\})$, where $z_1, ..., z_n$ are special variables of sorts $s_1, ..., s_n$. For any Σ-algebra A, the interpretation of γ in A is the map $\gamma_A\colon A_{s_1} \times \cdots \times A_{s_n} \to A_s$ defined as $\gamma_A(a_1, ..., a_n) = \theta^*(\gamma)$, where $\theta\colon \{z_1, ..., z_n\} \to A$ takes z_i to a_i for all $i = 1...n$. We let $Der(\Sigma)$ denote the signature of all derived operations of Σ. □

A common case is that $\mathcal{B}' = (Der(\Sigma), \Gamma, E)$ and Δ is a subsignature of derived operations over Γ. The following further extends the applicability of coinduction:

Fact 18 If $\mathcal{B}' = (Der(\Sigma), \Gamma, E)$ then $\mathcal{B}'$ is a conservative extension of $\mathcal{B}$, and if in addition $\Delta \subseteq Der(\Gamma)$ is context complete for $\mathcal{B}$, then Δ is a cobasis for $\mathcal{B}$. □

In many cases, the form of equations suggests which operations to put into Δ, as in the `STACK` specification (Example 20), where it is easily seen that any context over `top`, `push` and `pop` is equivalent to a context over only `top` and `pop`. Following [1, 2], an algorithm for reducing the number of contexts based on context rewriting is given in [15] for certain behavioral specifications[5] to reduce the number of contexts; it can be applied to get a context complete Δ (when $\mathcal{B}' = (Der(\Sigma), \Gamma, E)$, see Fact 18).

The first effective algebraic proof technique for behavioral properties was context induction, introduced by Rolf Hennicker [13] and further developed in joint work with Michel Bidoit; unfortunately, context induction can be awkward to apply in practice, as noticed in [6]. Hidden coinduction was proposed as a way to avoid this awkwardness.

4 Proving Congruence

This section discusses techniques for proving that operations are congruent with respect to Γ. We first give a general method that requires deduction, and then a more specific but surprisingly applicable method that only requires checking the form of equations.

Example 19 Consider the following behavioral theory of sets that differs from the one in Example 2 by having just one behavioral operation, `in`; it is also written in CafeOBJ:

[5] There is only one hidden sort and all operations have at most one hidden sort, but we think the method should extend to behavioral operations with many hidden sorts as in our framework.

```
mod* SET { *[ Set ]* pr(NAT)
bop _in_ : Nat Set -> Bool    ** attribute
op empty : -> Set             ** hidden constant
op add   : Nat Set -> Set     ** method
op _U_   : Set Set -> Set
op _&_   : Set Set -> Set
op neg   : Set -> Set         ** method
vars N N' : Nat  vars X X' : Set
eq N in empty = false .
eq N in add(N',X) = (N == N') or (N in X) .
eq N in (X U X')  = (N in X)  or (N in X') .
eq N in (X & X')  = (N in X) and (N in X') .
eq N in neg(X)    = not (N in X) . }
```

We prove that all operations are congruent. By Fact 7, both `in` and `empty` are congruent. Let Δ be the signature of `NAT` together with `in` and notice that Δ is a cobasis for `SET` (because Δ contains exactly the signature of `NAT` and the behavioral operations), so the six inference rules are sound for the behavioral satisfaction.

Congruence of add: By Proposition 9, we have to prove that

$\texttt{SET} \models\!\equiv (\forall N : \texttt{Nat}, X, X' : \texttt{Set})\ \texttt{add}(N, X) = \texttt{add}(N, X')$ **if** $X = X'$.

By the theorem of hidden constants (Theorem 10), this is equivalent to proving that

$\texttt{SET}_X \models\!\equiv (\forall N : \texttt{Nat})\ \texttt{add}(N, x) = \texttt{add}(N, x')$ **if** $x = x'$,

where $\texttt{SET}_X$ adds to `SET` two hidden constants, x and x'. By Fact 11, it is equivalent to

$\texttt{SET}' \models\!\equiv (\forall N : \texttt{Nat})\ \texttt{add}(N, x) = \texttt{add}(N, x')$,

where $\texttt{SET}'$ adds to $\texttt{SET}_X$ the equation $(\forall\emptyset)\ x = x'$. Now we use the six inference rules to prove that $\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall N : \texttt{Nat})\ \texttt{add}(N, x) = \texttt{add}(N, x')$. The following inferences give the proof:

1.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall M, N : \texttt{Nat})\ M = M$	(1)
2.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall M, N : \texttt{Nat})\ x = x'$	(4)
3.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall M, N : \texttt{Nat})\ M\ \texttt{in}\ x = M\ \texttt{in}\ x'$	(5)
4.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall M, N : \texttt{Nat})\ (M == N) = (M == N)$	(1)
5.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall M, N : \texttt{Nat})\ (M == N)\ \texttt{or}\ (M\ \texttt{in}\ x) = (M == N)\ \texttt{or}\ (M\ \texttt{in}\ x')$	(5)
6.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall M, N : \texttt{Nat})\ M\ \texttt{in}\ \texttt{add}(N, x) = (M == N)\ \texttt{or}\ (M\ \texttt{in}\ x)$	(4)
7.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall M, N : \texttt{Nat})\ M\ \texttt{in}\ \texttt{add}(N, x') = (M == N)\ \texttt{or}\ (M\ \texttt{in}\ x')$	(4)
8.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall M, N : \texttt{Nat})\ M\ \texttt{in}\ \texttt{add}(N, x) = M\ \texttt{in}\ \texttt{add}(N, x')$	(2), (3)
9.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall N : \texttt{Nat})\ \texttt{add}(N, x) = \texttt{add}(N, x')$	(6)

The rest follows by the soundness of the six rule inference system.

Congruence of _U_: By Proposition 9, Theorem 10 and Fact 11, this is equivalent to $\texttt{SET}' \models\!\equiv (\forall\emptyset)\ x_1\ \texttt{U}\ x_2 = x'_1\ \texttt{U}\ x'_2$, where $\texttt{SET}'$ adds to `SET` the hidden constants x_1, x'_1, x_2, x'_2 and the equations $(\forall\emptyset)\ x_1 = x'_1, (\forall\emptyset)\ x_2 = x'_2$. One can infer the following:

1.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall N : \texttt{Nat})\ N\ \texttt{in}\ x_1 = N\ \texttt{in}\ x'_1$	(1), (4), (5)
2.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall N : \texttt{Nat})\ N\ \texttt{in}\ x_2 = N\ \texttt{in}\ x'_2$	(1), (4), (5)
3.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall N : \texttt{Nat})\ (N\ \texttt{in}\ x_1)\ \texttt{or}\ (N\ \texttt{in}\ x_2) = (N\ \texttt{in}\ x'_1)\ \texttt{or}\ (N\ \texttt{in}\ x'_2)$	(5)
4.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall N : \texttt{Nat})\ N\ \texttt{in}\ (x_1\ \texttt{U}\ x_2) = (N\ \texttt{in}\ x_1)\ \texttt{or}\ (N\ \texttt{in}\ x_2)$	(4)
5.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall N : \texttt{Nat})\ N\ \texttt{in}\ (x'_1\ \texttt{U}\ x'_2) = (N\ \texttt{in}\ x'_1)\ \texttt{or}\ (N\ \texttt{in}\ x'_2)$	(4)
6.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall N : \texttt{Nat})\ N\ \texttt{in}\ (x_1\ \texttt{U}\ x_2) = N\ \texttt{in}\ (x'_1\ \texttt{U}\ x'_2)$	(2), (3)
7.	$\texttt{SET}' \Vdash_{\{\texttt{in}\}} (\forall\emptyset)\ x_1\ \texttt{U}\ x_2 = x'_1\ \texttt{U}\ x'_2$	(6)

The rest follows by the soundness of the six rule inference system. The congruences of _&_ and `neg` follows similarly. □

A similar approach was used in proving the congruence of the operations in the previous example. We capture it in the following method for proving the congruence of an operation

$\sigma : wh_1...h_k \to s$ for a hidden specification $\mathcal{B} = (\Sigma, \Gamma, E)$:

METHOD FOR PROVING CONGRUENCE: • **Step 1:** Choose a suitable $\Delta \subseteq \Sigma$ and show that it is a cobasis. Usually Δ is just Γ, in which case it is automatically a cobasis. • **Step 2:** Introduce appropriate new hidden constants $x_1, x'_1, ..., x_k, x'_k$, and new equations $(\forall\emptyset)\ x_1 = x'_1$,...,$(\forall\emptyset)\ x_k = x'_k$. Let $\mathcal{B}'$ denote the new hidden specification. • **Step 3:** Show $\mathcal{B}' \Vdash_\Delta (\forall Y)\ \sigma(Y, x_1, ..., x_k) = \sigma(Y, x'_1, ..., x'_k)$, where Y is a set of appropriate visible variables.

The correctness of this method follows from Proposition 9, Theorem 10 and Fact 11. Let's see how it works on another example:

Example 20 The following is a CafeOBJ behavioral theory of stacks of natural numbers:

```
mod* STACK { *[Stack]* pr(NAT)
bop top  : Stack -> Nat          ** attribute
bop pop  : Stack -> Stack        ** method
op  push : Nat Stack -> Stack    ** method
var N : Nat  var X : Stack
eq top(push(N,X)) = N .
beq pop(push(N,X)) = X . }
```

Let us prove the congruence of push using the strategy described above:

- **Step 1:** Let Δ be the signature of NAT together with top and pop. Then Δ is a cobasis for STACK because it contains exactly the data signature and the behavioral operations.
- **Step 2:** Introduce two hidden constants x and x' and the equation $(\forall\emptyset)\ x = x'$. Let STACK$'$ be the new hidden specification.
- **Step 3:** Prove $(\forall N : \mathtt{Nat})\ \mathtt{push}(N, x) = \mathtt{push}(N, x')$. One natural proof might be:
  1. $\mathtt{STACK}' \Vdash_{\{\mathtt{top},\mathtt{pop}\}} (\forall N : \mathtt{Nat})\ \mathtt{top}(\mathtt{push}(N, x)) = N$ (4)
  2. $\mathtt{STACK}' \Vdash_{\{\mathtt{top},\mathtt{pop}\}} (\forall N : \mathtt{Nat})\ \mathtt{top}(\mathtt{push}(N, x')) = N$ (4)
  3. $\mathtt{STACK}' \Vdash_{\{\mathtt{top},\mathtt{pop}\}} (\forall N : \mathtt{Nat})\ \mathtt{top}(\mathtt{push}(N, x)) = \mathtt{top}(\mathtt{push}(N, x'))$ (2), (3)
  4. $\mathtt{STACK}' \Vdash_{\{\mathtt{top},\mathtt{pop}\}} (\forall N : \mathtt{Nat})\ \mathtt{pop}(\mathtt{push}(N, x)) = x$ (4)
  5. $\mathtt{STACK}' \Vdash_{\{\mathtt{top},\mathtt{pop}\}} (\forall N : \mathtt{Nat})\ \mathtt{pop}(\mathtt{push}(N, x')) = x'$ (4)
  6. $\mathtt{STACK}' \Vdash_{\{\mathtt{top},\mathtt{pop}\}} (\forall N : \mathtt{Nat})\ x = x'$ (4)
  7. $\mathtt{STACK}' \Vdash_{\{\mathtt{top},\mathtt{pop}\}} (\forall N : \mathtt{Nat})\ \mathtt{pop}(\mathtt{push}(N, x)) = \mathtt{pop}(\mathtt{push}(N, x'))$ (2), (3)
  8. $\mathtt{STACK}' \Vdash_{\{\mathtt{top},\mathtt{pop}\}} (\forall N : \mathtt{Nat})\ \mathtt{push}(N, x) = \mathtt{push}(N, x')$ (6)

□

4.1 A Congruence Criterion

Let $\mathcal{B} = (\Sigma, \Gamma, E)$ be a hidden specification and let $\sigma : v_1...v_m h_1...h_k \to h$ be an operation in Σ, where $v_1, ..., v_m$ are visible sorts and $h_1, ..., h_k, h$ are hidden sorts. If $W = \{y_1 : v_1, ..., y_m : v_m, x_1 : h_1, ..., x_k : h_k\}$ is a set of variables then $\sigma(W)$ denotes the term $\sigma(y_1, ..., y_m, x_1, ..., x_k)$. Then

Theorem 21 If Δ is a cobasis for $\mathcal{B}$ in a conservative extension $\mathcal{B}' = (\Sigma', \Gamma', E')$ of $\mathcal{B}$ and if for each appropriate $\delta\colon s_1...s_n \to s$ in Δ, there is some γ in $T_{\Gamma'}(Z_j \cup W)$ such that[6] $\mathcal{B}' \models (\forall Z_j, W)\ \delta(Z_j, \sigma(W)) = \gamma$ for $j = 1, ..., n$, then σ is behaviorally congruent for $\mathcal{B}$.

Proof: By Proposition 9, the Theorem of Hidden Constants (Theorem 10) and Fact 11, it suffices to show that $\mathcal{B}_{X,X'} \models (\forall Y)\ \sigma(Y, x_1, ..., x_k) = \sigma(Y, x'_1, ..., x'_k)$, where $\mathcal{B}_{X,X'} = (\Sigma \cup X \cup X', \Gamma, E \cup \{(\forall\emptyset)\ x_1 = x'_1, ..., (\forall\emptyset)\ x_k = x'_k\})$. Let $\mathcal{B}'_{X,X'}$ be the hidden spec-

[6] We use the same notational conventions as in Lemma 15.

ification $(\Sigma' \cup X \cup X', \Gamma', E' \cup \{(\forall\emptyset)\ x_1 = x'_1, ..., (\forall\emptyset)\ x_k = x'_k\})$. It is straightforward that $\mathcal{B}'_{X,X'}$ is a conservative extension of $\mathcal{B}_{X,X'}$ and that Δ is also a cobasis for $\mathcal{B}_{X,X'}$. We claim that $\mathcal{B}'_{X,X'} \models (\forall Z_j, Y)\ \delta(Z_j, \sigma(Y, x_1, ..., x_k)) = \delta(Z_j, \sigma(Y, x'_1, ..., x'_k))$. Indeed, it is not difficult to observe that $\mathcal{B}'_{X,X'}$ behaviorally satisfies $(\forall Z_j, W)\ \delta(Z_j, \sigma(W)) = \gamma$, so by 4 of Proposition 12, $\mathcal{B}'_{X,X'}$ satisfies $(\forall Z_j, Y)\ \delta(Z_j, \sigma(Y, x_1, ..., x_k)) = \gamma_x$ and $\mathcal{B}'_{X,X'}$ satisfies $(\forall Z_j, Y)\ \delta(Z_j, \sigma(Y, x'_1, ..., x'_k)) = \gamma_{x'}$, where γ_x and $\gamma_{x'}$ are γ in which each variable x_i in X is replaced by the corresponding constants x_i and x'_i, respectively, and since γ contains only operations in Γ' (which are behaviorally congruent for $\mathcal{B}'_{X,X'}$), by 4 of Proposition 12, we get $\mathcal{B}'_{X,X'} \models (\forall Z_j, Y)\ \gamma_x = \gamma_{x'}$. Then by Lemma 15, $\mathcal{B}_{X,X'} \models (\forall Y)\ \sigma(Y, x_1, ..., x_k) = \sigma(Y, x'_1, ..., x'_k)$, that is, σ is behaviorally congruent for $\mathcal{B}$. □

Corollary 22 Congruence Criterion: If for each appropriate $\delta\colon s_1 ... s_n \to s$ in Γ and each $j = 1, ..., n$ such that $s_j = s$, there is some γ in $T_\Gamma(Z_j \cup W)$ such that the Σ-equation $(\forall Z_j, W)\ \delta(Z_j, \sigma(W)) = \gamma$ is in[7] E, then σ is behaviorally congruent for $\mathcal{B}$. □

Most examples fall under this easy to check criterion, including every example in this paper, and it would be easy to implement the criterion in a system like CafeOBJ.

5 Reducing the Behavioral Operations

The fewer operations Δ has, the easier it is to apply the Δ-coinduction rule. Most often, Δ contains the data signature and only behavioral operations, either all or only part of them. Therefore, it is important to have as few behavioral operations as possible in a hidden specification.

Definition 23 Hidden specifications $\mathcal{B}_1 = (\Sigma, \Gamma_1, E_1)$ and $\mathcal{B}_2 = (\Sigma, \Gamma_2, E_2)$ over the same hidden signature are **equivalent** iff for any hidden Σ-algebra A, $A \models \mathcal{B}_1$ iff $A \models \mathcal{B}_2$ and in this case, $\equiv^{\Gamma_1}_\Sigma = \equiv^{\Gamma_2}_\Sigma$ on A. □

Assumption: $\mathcal{B}_1 = (\Sigma, \Gamma_1, E)$ and $\mathcal{B}_2 = (\Sigma, \Gamma_2, E)$ are two hidden specifications over the same signature with the same equations and with $\Gamma_1 \subseteq \Gamma_2$; also the Σ-equations in E have no conditions of hidden sorts.

Fact 24 $\mathcal{B}_1$ and $\mathcal{B}_2$ are equivalent iff $A \models \mathcal{B}_1$ implies $\equiv^{\Gamma_1}_\Sigma \subseteq \equiv^{\Gamma_2}_\Sigma$ for every hidden Σ-algebra A; moreover, $\mathcal{B}_1$ is a conservative extension of $\mathcal{B}_2$. □

Theorem 25 $\mathcal{B}_1$ and $\mathcal{B}_2$ are equivalent iff all operations in Γ_2 are behaviorally congruent for $\mathcal{B}_1$.

Proof: If $\mathcal{B}_1$ and $\mathcal{B}_2$ are equivalent then $\equiv^{\Gamma_1}_\Sigma = \equiv^{\Gamma_2}_\Sigma$ for every hidden Σ-algebra A with $A \models \mathcal{B}_1$. Since the operations in Γ_2 are congruent for $\equiv^{\Gamma_2}_\Sigma$ (see Theorem 4), they are also congruent for $\equiv^{\Gamma_1}_\Sigma$, so they are behaviorally congruent for $\mathcal{B}_1$.

Conversely, suppose that all operations in Γ_2 are behaviorally congruent for $\mathcal{B}_1$ and let A be a hidden Σ-algebra such that $A \models \mathcal{B}_1$. Then for every $a, a' \in A_h$ such that $a \equiv^{\Gamma_1}_{\Sigma,h} a'$, for every Γ_2-context c and for every $\theta\colon var(c) \to A$, we get $A_c(a, \theta) = A_c(a', \theta)$, that is, $a \equiv^{\Gamma_2}_{\Sigma,h} a'$. Therefore $\equiv^{\Gamma_1}_\Sigma \subseteq \equiv^{\Gamma_2}_\Sigma$, so by Fact 24, $\mathcal{B}_1$ and $\mathcal{B}_2$ are equivalent. □

Example 26 The restriction on conditional equations cannot be neglected: Consider the following CafeOBJ behavioral theory:

```
mod* B1 { *[ S ]*
bop f : S -> Bool      ** attribute
op  g : S -> Bool      ** attribute
vars X X' : S
bceq g(X) = g(X') if X == X' . }
```

[7] Modulo renaming of variables.

Notice that g is congruent for B1. Now let us consider another CafeOBJ behavioral theory in which g is also behavioral:

```
mod* B2 { *[ S ]*
bop f : S -> Bool      ** attribute
bop g : S -> Bool      ** attribute
vars X X' : S
bceq g(X) = g(X') if X == X' . }
```

Let Σ be the (common) signature of B1 and B2, containing the operations on the booleans plus `f : S -> Bool` and `g : S -> Bool`. Then for any hidden Σ-algebra A, $A \models\!\equiv$ B1 iff $A_f(a) = A_f(a')$ implies $A_g(a) = A_g(a')$ for any $a, a' \in A_s$, and $A \models\!\equiv$ B2 under no restrictions. Therefore B1 and B2 are not equivalent because there exist hidden Σ-algebras satisfying B2 which do not satisfy B1. Because B1 and B2 satisfy all the hypotheses in Theorem 25 except the one regarding the conditional equations, it follows that this restriction cannot be omitted. □

Example 27 SET1 of Example 2 and SET of Example 19 are equivalent, because all behavioral operations in SET1 are congruent for SET. Similarly, STACK of Example 20 is equivalent to the behavioral specification where push is also behavioral. □

Proposition 28 If Γ_1 is context complete[8] for $\mathcal{B}_2$ then $\mathcal{B}_1$ and $\mathcal{B}_2$ are equivalent.

Proof: Let A be any hidden Σ-algebra such that $A \models\!\equiv \mathcal{B}_1$, and let $a \equiv^{\Gamma_1}_{\Sigma,h} a'$. Since for every Γ_2-context c over z of sort h there is some γ in $T_{\Gamma_1}(\{z\} \cup var(c))$ such that $\mathcal{B}_1 \models\!\equiv (\forall z, var(c))\ c = \gamma$, we get that $A_c = A_\gamma$ as functions $A_h \times A^{var(c)} \to D$, where A_γ is defined similarly to A_c, that is, $A_\gamma(a,\theta) = a^*_\theta(\gamma)$. As γ has visible sort and contains only operations in Γ_1 (which are congruent for $\equiv^{\Gamma_1}_{\Sigma}$), we get $A_\gamma(a,\theta) = A_\gamma(a',\theta)$ for any $\theta\colon var(c) \to A$. Therefore $A_c(a,\theta) = A_c(a',\theta)$ for any θ, that is, $a \equiv^{\Gamma_2}_{\Sigma,h} a'$. Therefore $\equiv^{\Gamma_1}_{\Sigma} \subseteq \equiv^{\Gamma_2}_{\Sigma}$, and so by Fact 24, $\mathcal{B}_1$ and $\mathcal{B}_2$ are equivalent. □

Example 29 Let LIST be the following behavioral specification:

```
mod* LIST { *[ List ]* pr(NAT)
bop car  : List -> Nat
bop cdr  : List -> List
bop cons : Nat List -> List
bop _in_ : Nat List -> Bool
vars N N' : Nat  var L : List
eq  car(cons(N,L)) = N .
beq cdr(cons(N,L)) = L .
eq N' in cons(N,L) = (N == N') or (N in L) . }
```

If Ψ is its data signature (natural numbers and booleans), and Σ and E are its hidden signature and equations, then the spec is $(\Sigma, \Psi \cup \{\mathtt{car}, \mathtt{cdr}, \mathtt{cons}, \mathtt{in}\}, E)$. By the congruence criterion (Corrolary 22), cons is congruent for LIST1 $= (\Sigma, \Psi \cup \{\mathtt{car}, \mathtt{cdr}, \mathtt{in}\}, E)$, and so Theorem 25 implies that LIST and LIST1 are equivalent. They have many models, including the standard finite lists (a reachable model) and infinite lists (an unreachable model). Note that car and in can behave unexpectedly on the unreachable states of some models.

Now let LIST2 be the behavioral specification $(\Sigma, \Psi \cup \{\mathtt{in}\}, E)$. Again by the congruence criterion, cons is behaviorally congruent for LIST2. One model for LIST2 is the Σ-algebra of finite lists (with any choice for car(nil) and cdr(nil), such as 0 and nil), in which two lists are behaviorally equivalent iff they contain the same natural numbers (without regard to their order and number of occurrences). Therefore car and cdr are not behaviorally congruent for LIST2.

[8] This makes sense becasue $\mathcal{B}_1$ is a conservative extension of $\mathcal{B}_2$.

Another interesting behavioral specification is $\texttt{LIST3} = (\Sigma, \Psi \cup \{\texttt{car}, \texttt{cdr}\}, E)$, for which `cons` is also behaviorally congruent, but `in` is not necessarily congruent, because it can be defined in almost any way on unreachable states. □

References

[1] Narjes Berregeb, Adel Bouhoula, and Michaël Rusinowitch. Observational proofs with critical contexts. In *Fundamental Approaches to Software Engineering*, volume 1382 of *Lecture Notes in Computer Science*, pages 38–53. Springer, 1998.

[2] Michael Bidoit and Rolf Hennicker. Behavioral theories and the proof of behavioral properties. *Theoretical Computer Science*, 165:3–55, 1996.

[3] Rod Burstall and Răzvan Diaconescu. Hiding and behaviour: an institutional approach. In Andrew William Roscoe, editor, *A Classical Mind: Essays in Honour of C.A.R. Hoare*, pages 75–92. Prentice-Hall, 1994.

[4] Răzvan Diaconescu. Behavioural coherence in object-oriented algebraic specification. Technical Report IS–RR–98–0017F, Japan Advanced Institute for Science and Technology, June 1998. Submitted for publication.

[5] Răzvan Diaconescu and Kokichi Futatsugi. *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification.* World Scientific, 1998. AMAST Series in Computing, volume 6.

[6] Marie-Claude Gaudel and Igor Privara. Context induction: an exercise. Technical Report 687, LRI, Université de Paris-Sud, 1991.

[7] Joseph Goguen. Types as theories. In George Michael Reed, Andrew William Roscoe, and Ralph F. Wachter, editors, *Topology and Category Theory in Computer Science*, pages 357–390. Oxford, 1991. Proceedings of a Conference held at Oxford, June 1989.

[8] Joseph Goguen and Răzvan Diaconescu. Towards an algebraic semantics for the object paradigm. In Hartmut Ehrig and Fernando Orejas, editors, *Proceedings, Tenth Workshop on Abstract Data Types*, pages 1–29. Springer, 1994. Lecture Notes in Computer Science, Volume 785.

[9] Joseph Goguen, Kai Lin, Akira Mori, Grigore Roşu, and Akiyoshi Sato. Tools for distributed cooperative design and validation. In *Proceedings, CafeOBJ Symposium.* Japan Advanced Institute for Science and Technology, 1998. Nomuzu, Japan, April 1998.

[10] Joseph Goguen and Grant Malcolm. Proof of correctness of object representation. In Andrew William Roscoe, editor, *A Classical Mind: Essays in Honour of C.A.R. Hoare*, pages 119–142. Prentice-Hall, 1994.

[11] Joseph Goguen and Grant Malcolm. *Algebraic Semantics of Imperative Programs.* MIT, 1996.

[12] Joseph Goguen and Grant Malcolm. A hidden agenda. *Theoretical Computer Science*, to appear 1999. Also UCSD Dept. Computer Science & Eng. Technical Report CS97–538, May 1997.

[13] Rolf Hennicker. Context induction: a proof principle for behavioural abstractions. *Formal Aspects of Computing*, 3(4):326–345, 1991.

[14] Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the European Association for Theoretical Computer Science*, 62:222–259, June 1997.

[15] Michihiro Matsumoto and Kokichi Futatsugi. Test set coinduction: Toward automated verification of behavioural properties. In *Proceedings of the Second International Workshop on Rewriting Logic and its Applications*, Electronic Notes in Theoretical Computer Science. Elsevier Science, to appear 1998.

[16] Peter Padawitz. Towards the one-tiered design of data types and transition systems. In *Proceedings, WADT'97*, pages 365–380. Springer, 1998. Lecture Notes in Computer Science, Volume 1376.

[17] Horst Reichel. An approach to object semantics based on terminal co-algebras. *Mathematical Structures in Computer Science*, 5:129–152, 1995.