

Behavioral Coinductive Rewriting

Grigore Roşu*

Department of Computer Science & Engineering
University of California at San Diego

1 Introduction

Specifying and verifying behavioral properties in addition to strict properties of systems is becoming an appropriate approach to the actual software engineering. The Internet facilitates distributed cooperative projects and implicitly distributed cooperative software systems whose behavior is difficult to catch using only strict equalities of states. At the author's knowledge, *coinduction* [11, 22, 18, 13, 12, 15] and *context induction* [16, 6, 1] are among the most frequent techniques to prove behavioral equivalences of states and they both require human intervention. Proving automatically behavioral equivalences is a useful feature of specification languages. CafeOBJ [3, 4] has implemented *behavioral rewriting*² to make behaviorally sound reductions of terms. In this report, we propose a new technique which combines behavioral rewriting and coinduction.

Equational logic is not sound for behavioral satisfaction since there might be operations which are not behaviorally congruent, that is, which do not preserve the behavioral equivalence. We modified the congruence rule of equational logic accordingly in Section 3 (see also [21, 14]), obtaining a sound five rule system for hidden algebra. Behavioral rewriting is for these rules what standard rewriting is for equational logic and (many sorted) algebra. In particular, in section 4 we show that if the behavioral rewriting relation is confluent, then an equation $(\forall X) t = t'$ is derivable by the five rule inference system iff t and t' rewrite to a common term.

Induction is a well established technique to prove strict equalities of terms (and not only) in the framework of abstract data types. Induction's *basis*, that is, a set of operations or derived operations generating all the terms, play a major role in inductive proofs and good choices of them can enormously simplify the proofs³. Dually, coinduction is a technique to prove behavioral equalities of terms in the framework of object types. We introduce the notion of *cobasis* as a generalization of a similar concept in [21], which is a set of operations which (co)generate the behavioral equivalence on terms, and provide a method to obtain cobases using a syntactic criterion. Given a cobasis, coinduction can be applied automatically; we capture that in a new inference rule called Δ -*coinduction*.

Behavioral coinductive rewriting is a method built on behavioral rewriting, its purpose being to prove automatically more behavioral equalities than behavioral rewriting alone can do. Unlike behavioral rewriting which rewrites well formed terms, behavioral coinductive rewriting rewrites sets of pairs of terms, called *goals*. Each rewriting step is either a many step behavioral rewriting or a coinduction step applied to a goal, that is, a replacement of that goal with a finite set of goals generated by putting the operations in the cobasis on top of both terms in that goal. Two terms t and t' are behaviorally equivalent if the goal $\{(t, t')\}$ can be reduced to a set of pairs of equal terms (called trivial goals). Confluence and completeness (with regards to the six inference rules) are investigated, and syntactic criteria for termination and completeness of an algorithm based on behavioral coinductive rewriting are given in subsection 6.1.

* On leave from Fundamentals of Computer Science, Faculty of Mathematics, University of Bucharest, Romania.

² Actually a subrelation of it; see Section 4

³ In Chapter 8 of [7], there is a nice inductive proof of the Euler formula in which the natural numbers are generated by prime numbers and multiplications with prime numbers, so an infinite basis.

2 Hidden Algebra

Hidden algebra [8, 9] appeared to give algebraic semantics for the object paradigm. A comprehensive survey for the case in which operations are all behavioral and have at most one hidden argument can be found in [13, 12]. We remind the reader the basic definitions and results of hidden algebra in its slightly extended form:

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

- 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.

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 $A|_{\Psi} = D$.

Definition 2. Given a hidden Σ -algebra A and any equivalence \sim on A , an operation σ in $\Sigma_{s_1 \dots s_n, s}$ is **congruent for \sim** iff $A_{\sigma}(a_1, \dots, a_n) \sim A_{\sigma}(a'_1, \dots, a'_n)$ whenever $a_i \sim a'_i$ for $i = 1 \dots n$. A **hidden Γ -congruence on A** is an equivalence on A which is identity on visible sorts and such that each operation in Γ is congruent for it.

The following result proved in [21] generalizes a similar result in [13] to operations that can have more than one hidden argument.

Theorem 3. *Given a hidden subsignature Γ of Σ and a hidden Σ -algebra A , there exists a largest hidden Γ -congruence on A .*

There might be readers thinking that our extension to behavioral operations with many hidden arguments leads to serious foundational problems, such as the non-consistency of behavioral specifications (non-existence of models). We claim that this problem is not different by any means in our extended framework from the similar problem in basic hidden algebra, because by Theorem 3, any hidden algebra comes together with a largest hidden Γ -congruence, which allows defining the behavioral satisfaction (as it is shown below), exactly as in the case of basic hidden algebra. The only inconvenience we are aware of so far is the possible lack of a final model, but fortunately, this is not crucial neither to giving semantics to the modular composition of behavioral specifications nor to the soundness of proof techniques such as coinduction, as the theorem above suggests. Undoubtedly, criteria to insure consistency of specifications in our extended framework as the one in [13] for basic hidden specifications would be an interesting and challenging topic of research, but it is beyond the goal of the present paper.

Definition 4. The largest hidden Γ -congruence in Theorem 3 is called the Γ -**behavioral equivalence** and it is denoted by \equiv_{Σ}^* . A hidden Σ -algebra A **Γ -behaviorally satisfies** a Σ -equation $e = (\forall X) t = t'$ iff $\theta(t) \equiv_{\Sigma}^* \theta(t')$ for each $\theta: X \rightarrow A$; in this case we write $A \models_{\Sigma}^* e$. If E is a set of Σ -equations, we write $A \models_{\Sigma}^* E$ 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_{Σ}^* and \models_{Σ}^* , respectively. We say that A **behaviorally satisfies** (or **is a model of**) a behavioral specification $\mathcal{B} = (\Sigma, \Gamma, E)$ iff $A \models_{\Sigma}^* E$, and in this case we write $A \models \mathcal{B}$; we write $\mathcal{B} \models e$ whenever $A \models \mathcal{B}$ implies $A \models_{\Sigma}^* e$. An operation σ is **Γ -behaviorally congruent for A** iff σ is congruent for \equiv_{Σ}^* ; we will often say just “congruent” instead of “behaviorally congruent”⁴. An operation $\sigma \in \Sigma$ is **behaviorally congruent for \mathcal{B}** if σ is behaviorally congruent for every $A \models \mathcal{B}$.

⁴ A similar notion has been given by Padawitz [20].

Up to the author’s knowledge, the compatibility of behavioral equivalence with functions and/or predicates was for the first time investigated in [19], and then it has been further explored in [3, 5, 2], where it is called *behavioral coherence*, and in [20]⁵.

2.1 Examples

In this paper we use a CafeOBJ-like syntax since, at the author’s knowledge, it is the only specification language that explicitly distinguishes between behavioral and non-behavioral operations and implements behavioral rewriting as operational engine to automatically deduce behavioral equalities. However, the code presented in this paper is not entirely executable on the actual CafeOBJ platform because of the lack of the current implementation to handle many-hidden-sorted behavioral operations.

Example 1. Nondeterministic Stack: The following example after [13] motivates non-behavioral operations. We specify a random number generator for a distributed system, as a separate process that places generated numbers on a stack (for simplicity, assuming infinite storage), so that other processes can take numbers from the stack, each consumed by exactly one call from one process, since multiple access to a single number is wrong. We consider two stack states equivalent if they have the same numbers in the same order; then `top` and `pop` are behavioral with respect to this equivalence, but `push` is not, since its behavior should not be determined by what is on the stack.

```
mod* NDSTACK { pr(NAT)
  *[ Stack ]*
  op empty : -> Stack      ** hidden constant
  op push  : Stack -> Stack ** method
  bop top   : Stack -> Nat  ** attribute
  bop pop   : Stack -> Stack ** method
  var S : Stack
  beq pop(empty) = empty .
  beq pop(push(S)) = S . }
```

An implementation might be a function $f : Nat \rightarrow Nat$ where $f(n)$ is the n th randomly generated number. To ensure that n changes with each new random number generation, we can keep it as a variable with the stack, incremented whenever a new number is pushed. Such an implementation is equivalent to the following model: Let $A_{Nat} = \omega$, where ω is the natural numbers, let $A_{Stack} = \omega \times \omega^*$, where ω^* is lists of naturals. Using [head,tail] list notation with $[]$ for the empty list, let $A_{empty} = (0, [])$, $A_{top}((n, [])) = 0$, $A_{top}((n, [h, t])) = h$, $A_{pop}((n, [])) = (n, [])$, $A_{pop}((n, [h, t])) = (n, t)$, and $A_{push}((n, l)) = (n + 1, [f(n), l])$. Then two states are behaviorally equivalent iff for every sequence of pops followed by a top they give the same number, that is, they store the same elements in the same order; in other words, $(n, l) \equiv (n', l')$ iff $l = l'$. Note that `push` is not behaviorally congruent for this model, because $f(n)$ can be different from $f(n')$.

Example 2. Sets:

```
mod SET1 { pr(NAT)
  *[ Set ]*
  op empty : -> Set
  op {_}   : Nat -> Set
  bop _in_ : Nat Set -> Bool
  bop _U_  : Set Set -> Set
  bop _&_  : Set Set -> Set
  bop not_ : Set -> Set
  vars N N' : Nat   vars S S' : Set
  eq N in empty = false .
  eq N in { N' } = (N == N') . }
```

⁵ We should mention that the notion of behavioral congruence is presented in a more general context in [19, 20].

```

eq N in S U S' = (N in S) or (N in S') .
eq N in S & S' = (N in S) and (N in S') .
eq N in not S = not (N in S) . }

```

All operations, except the hidden constants which might also be behavioral, are behavioral as they preserve the intended equivalence between sets (two sets are equivalent iff they have the same elements). Later on, we will see that `in` is a cobasis for sets, that is, it (co)generates the behavioral equivalence on sets.

Notice that the union has two hidden sorted arguments and this is correct within our extended framework. In the actual implementation of CafeOBJ, which does not admit such operations, the only way to deal with many-hidden-argument operations is to declare them congruent using the attribute *coherent*. The fact that a specification in which the congruent operations are declared behavioral is by all means semantically equivalent to the specification in which they are declared *coherent* [21, 14], in addition to the operational argument that the behavioral rewriting does not distinguish between behavioral and congruent operations, suggests that by admitting many-hidden-argument behavioral operations the attribute *coherent* is not necessary anymore, making the definition of the language simpler without any loss of conceptual clarity.

Example 3. Streams: The following is a behavioral specification for infinite streams of natural numbers. There is a `zero` stream containing only zeros, an operation `cons` that adds a natural number in front of a stream, operations `head` and `tail` giving the head and the tail of a stream, and an operation `zip` which zips two streams, that is, `zip(0000..., 1111...)` is the stream `01010101...`:

```

mod STREAM1 { pr(NAT)
  *[ Stream ]*
  op zero : -> Stream
  bop cons : Nat Stream -> Stream
  bop head_ : Stream -> Nat
  bop tail_ : Stream -> Stream
  bop zip : Stream Stream -> Stream
  var N : Nat      vars S S' : Stream
  eq head zero = 0 .
  beq tail zero = zero .

  eq head cons(N, S) = N .
  beq tail cons(N, S) = S .

  eq head zip(S, S') = head(S) .
  beq tail zip(S, S') = zip(S', tail(S)) . }

```

The intuition is that two streams are equivalent iff they have the same elements in the same order. As all operations preserve this intuitive behavior, we declare them behavioral. One can consider at one's discretion `zero` as behavioral or not, and also the visible sorted equations as behavioral or not.

In order to simplify writing, given an operation $\sigma \in \Sigma_{s_1 \dots s_n, s}$ and a set of variables $W = \{w_1 : s_1, \dots, w_n : s_n\}$, we write $\sigma(W)$ instead of $\sigma(w_1, \dots, w_n)$; when interested in only one argument of σ , say the j th, we often write it as the last argument; moreover, if $W = \{w_1 : s_1, \dots, w_{j-1} : s_{j-1}, w_{j+1} : s_{j+1}, \dots, w_n : s_n\}$ and $x : s_j$ then $\sigma(W, x)$ is the term $\sigma(w_1, \dots, w_{j-1}, x, w_{j+1}, \dots, w_n)$.

Given a hidden sort h (or a term of sort h), an operation $\sigma \in \Sigma$ is *appropriate for h* (or t) iff σ has at least one argument of sort h . When we say that a property holds for all appropriate σ , we also mean for all arguments h of σ . For example, given two hidden sorted terms $t, t' \in T_{\Sigma, h}(X)$, the assertion “ $(\sigma(W, t), \sigma(W, t')) \in G$ for all appropriate $\sigma \in \Sigma$ ”, means that $(\sigma(w_1, \dots, w_{j-1}, t, w_{j+1}, \dots, w_n), \sigma(w_1, \dots, w_{j-1}, t', w_{j+1}, \dots, w_n)) \in G$ for all $\sigma \in \Sigma_{s_1 \dots s_n, s}$ and all $j \in 1, \dots, n$ such that $s_j = h$.

3 Five Sound Inference Rules for Hidden Algebra

As observed in [3, 5], equational deduction is not sound for behavioral satisfaction. This is because some operations might not be behaviorally congruent, so the congruence inference rule is not sound anymore. Since behavioral equivalence is identity on visible sorts, we can modify the congruence rule accordingly, obtaining the following inference rules; given $\mathcal{B} = (\Sigma, \Gamma, E)$, we 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 : $\frac{}{(\forall X) t = t}$
- (2) Symmetry : $\frac{(\forall X) t = t'}{(\forall X) t' = t}$
- (3) Transitivity : $\frac{(\forall X) t = t', (\forall X) t' = t''}{(\forall X) t = t''}$
- (4) Substitution : $\frac{(\forall Y) l = r \in E, \theta : Y \rightarrow T_\Sigma(X)}{(\forall X) \theta(l) = \theta(r)}$
- (5) Congruence : $\left\{ \begin{array}{l} a) \frac{(\forall X) t = t', \text{sort}(t, t') \in V}{(\forall X, W) \sigma(W, t) = \sigma(W, t'), \text{ for each } \sigma \in \Sigma} \\ b) \frac{(\forall X) t = t', \text{sort}(t, t') \in H}{(\forall X, W) \delta(W, t) = \delta(W, t'), \text{ for each congruent } \delta \in \Sigma} \end{array} \right\}$

If all operations are behaviorally congruent then the above rules become the well-known equational deduction system. The five inference rules are sound for behavioral satisfaction, that is, the following result generalizes the observation in [3, 5] that equational deduction is sound whenever all operations are congruent:

Proposition 5. $\mathcal{B} \Vdash (\forall X) t = t'$ implies $\mathcal{B} \equiv (\forall X) t = t'$.

Notice that *b*) can be applied only for congruent operations. Consequently, the more congruent operations are, the more powerful the inference system is, and implicitly the more behavioral equalities can be proved. Fortunately, all behavioral operations are behaviorally congruent [21], so there already are some cases in which *b*) can be applied. Additionally, techniques to prove congruence of operations, including an easy to check syntactic criterion, are provided in [21].

Another result in [21] says that a behavioral specification for which a certain operation is congruent, is actually equivalent (has the same models with the same behavioral equivalences) to a behavioral specification obtained from the first one just taking that operation behavioral. The consequence is that all congruent operations can be declared behavioral. If the user is faithful to this methodology then *b*) can be simplified to consider only behavioral operations.

On the other hand, the fewer behavioral operations are, the easier coinduction is (because the task of proving hidden congruence requires fewer subtasks). Therefore, there seems to be a tradeoff between taking either more or fewer behavioral operations. In Section 5, we show that actually there is no tradeoff, the notion of *cobasis* playing a central role.

4 Behavioral Rewriting

Behavioral rewriting is for the five inference rules presented in the previous section what standard term rewriting is for equational logic. Since we are not aware of any reference introducing the concept of behavioral rewriting in its full generality as we see it, we do it in this section.

Given two relations $R \subseteq A \times B$ and $S \subseteq B \times C$, their composition is the relation $R; S \subseteq A \times C$ defined as $a(R; S)c$ iff there are some $b \in B$ such that aRb and bRc . Since functions are special relations, one can have compositions between functions and relations as well.

Definition 6. A Σ -rewriting rule is a triple written $(\forall X) l \rightarrow r$, where X is a set of variables and $l, r \in T_\Sigma(X)$. A **behavioral** (or **hidden**) Σ -rewriting system is a triple (Σ, Γ, R) , where Σ is a hidden signature, Γ is a hidden subsignature of Σ , and R is a set of Σ -rewriting rules.

Framework: From now on in the paper, suppose that $\mathcal{R} = (\Sigma, \Gamma, R)$ is a behavioral Σ -rewriting system and $\mathcal{B} = (\Sigma, \Gamma, E)$ is the associated behavioral specification, that is, $E = \{(\forall X) l = r \mid (\forall X) l \rightarrow r \in R\}$.

Ordinary term rewriting is not sound for the behavioral satisfaction anymore. This is because non-behavioral operations might not preserve the behavioral equivalence. Consequently, it needs to be modified as follows:

Definition 7. The **behavioral (term) rewriting relation** associated to the behavioral rewriting system \mathcal{R} is the smallest relation \Rightarrow such that:

- For each $(\forall Y) l \rightarrow r$ in \mathcal{R} and each $\theta: Y \rightarrow T_\Sigma(X)$, $\theta(l) \Rightarrow \theta(r)$,
- If $t \Rightarrow t'$ and $\text{sort}(t, t') \in V$ then $\sigma(W, t) \Rightarrow \sigma(W, t')$ for all $\sigma \in \Sigma$,
- If $t \Rightarrow t'$ and $\text{sort}(t, t') \in H$ then $\delta(W, t) \Rightarrow \delta(W, t')$ for all $\delta \in \Gamma$.

Behavioral rewriting modifies the standard term rewriting as follows: each time a hidden redex is found, apply the rewriting rule when there are only congruent operations on the path from that redex going toward the root until *eventually* a visible sort is found. If a visible sort is not found, the rewriting is still applied if all operations on the path from the redex to the root are congruent. CafeOBJ successfully implements only a subrelation of behavioral rewriting (used within the command `reduce`). More exactly, the rewriting rule is applied when there are only congruent operations on the path from the redex going toward the root until a visible sort is found. If no visible sort is found on that path, the rewriting does not take place. Therefore, CafeOBJ's behavioral rewriting does not consider the case when all operations on the path to the redex are congruent and there is no visible sorted operation on that path. We believe that the behavioral rewriting might be implemented in its full generality within the CafeOBJ's `behavioral-reduce` command [3].

Proposition 8. If $t \Rightarrow^* t'$ then $\Vdash (\forall X) t = t'$, where $X = \text{var}(t) \cup \text{var}(t')$.

Definition 9. \mathcal{R} is **confluent** iff \Rightarrow is confluent, and \mathcal{R} is **terminating** iff \Rightarrow is terminating. \mathcal{R} is **canonical** iff it is both confluent and terminating.

Confluence and/or termination criteria of a behavioral rewriting system may be an interesting area of research, but we are not interested in it in this paper. However, notice that the termination of \mathcal{R} as a non-behavioral rewriting system produces the termination of \mathcal{R} as a behavioral rewriting system (because \Rightarrow is a subrelation of the ordinary rewriting); thus, any termination criterion for ordinary rewriting is still applicable for behavioral rewriting.

Lemma 10. *The following hold:*

1. $\Rightarrow^*; \Leftarrow^*$ is reflexive,
2. $\Rightarrow^*; \Leftarrow^*$ is symmetric,
3. If \Rightarrow is confluent then $\Rightarrow^*; \Leftarrow^*$ is transitive,
4. If $t \Rightarrow^*; \Leftarrow^* t'$ then
 - if $\text{sort}(t, t') \in V$ then $\sigma(x, t) \Rightarrow^*; \Leftarrow^* \sigma(x, t')$ for all $\sigma \in \Sigma$,
 - if $\text{sort}(t, t') \in H$ then $\delta(x, t) \Rightarrow^*; \Leftarrow^* \delta(x, t')$ for all $\delta \in \Gamma$.

Theorem 11. Let $(\forall X) t = t'$ be an equation. Then

1. $t \Rightarrow^*; \Leftarrow^* t'$ implies $\Vdash (\forall X) t = t'$, and
2. If \Rightarrow is confluent then $\Vdash (\forall X) t = t'$ implies $t \Rightarrow^*; \Leftarrow^* t'$.

Proof. 1. It follows by Proposition 8.

2. Since $\llbracket _ \rrbracket$ is the smallest relation closed under the five inference rules in the previous section, the result follows by Lemma 10.

If \Rightarrow is canonical then we let $nf(t)$ denote the normal form of t .

Corollary 12. *If \Rightarrow is confluent then $\llbracket _ \rrbracket (\forall X) t = t'$ iff $t \Rightarrow^*$; $\Leftarrow^* t'$, and if \Rightarrow is canonical then $\llbracket _ \rrbracket (\forall X) t = t'$ iff $nf(t) = nf(t')$.*

Example 4. Since all operations in **STREAM1** are behavioral, \Rightarrow is nothing else than the standard term rewriting, so one can easily prove by rewriting properties like $\mathbf{tail}^{28}(\mathbf{zip}(S, S')) \Rightarrow \mathbf{zip}(\mathbf{tail}^{14}(S), \mathbf{tail}^{14}(S'))$ and $\mathbf{tail}^{29}(\mathbf{zip}(S, S')) \Rightarrow \mathbf{zip}(\mathbf{tail}^{15}(S'), \mathbf{tail}^{14}(S))$, or by induction properties like $\mathbf{tail}^{2k}(\mathbf{zip}(S, S')) \Rightarrow \mathbf{zip}(\mathbf{tail}^k(S), \mathbf{tail}^k(S'))$ and also $\mathbf{tail}^{2k+1}(\mathbf{zip}(S, S')) \Rightarrow \mathbf{zip}(\mathbf{tail}^{k+1}(S'), \mathbf{tail}^k(S))$ for all $k \geq 0$, where \mathbf{tail}^k is a shorthand for the composition of \mathbf{tail} for k times.

However, properties like $(\forall S) \mathbf{cons}(\mathbf{head}(S), \mathbf{tail}(S)) = S$ and $(\forall N, S, S') \mathbf{zip}(\mathbf{cons}(N, S), S') = \mathbf{cons}(N, \mathbf{zip}(S', S))$ cannot be directly proved by behavioral rewriting. We will see later in the paper how such properties can automatically be proved by behavioral coinductive rewriting.

5 Cobases and Δ -Coinduction

Induction is a well established technique to prove strict equalities of terms in abstract data types. The *basis* of induction, i.e., some operations or derived operations generating all the terms, play a major role in inductive proofs. Dually, coinduction is a technique to prove behavioral equalities of terms. We introduce the notion of *cobasis* as a set of operations (or derived operations or even more generally, operations in a conservative extension) which (co)generate the behavioral equivalence of terms:

Definition 13. Given $\mathcal{B}' = (\Sigma', \Gamma', E')$ a conservative extension of $\mathcal{B} = (\Sigma, \Gamma, E)$ and $\Delta \subseteq \Sigma'$, then Δ is a **cobasis of \mathcal{B}** iff for each hidden sorted terms $t, t' \in T_{\Sigma, h}(X)$, $\mathcal{B}' \models (\forall W, X) \delta(W, t) = \delta(W, t')$ for all appropriate $\delta \in \Delta$ implies $\mathcal{B} \models (\forall X) t = t'$.

Perhaps the most trivial example of cobasis is when Δ is the (infinite) set of visible contexts of Σ and Σ' is the derived signature of Σ (see the definition below).

Another very important example of cobasis is when Δ contains exactly the behavioral operations (see the proposition below).

A more concrete definition of cobasis has been given in [21] (Definition 11; see also Theorem 12). Since the cobasis in [21] is a generalization of the notion of “complete set of Σ_{Obs} -contexts” in [2], one can conclude that the complete set of contexts is also a special case of cobasis in its general form in the above definition, so our results are also applicable to the framework of “observational logic” [17] (notice that the fact that the fixed data algebra D is protected by all models is not used at all in our proofs).

Definition 14. Given a (not necessary hidden) signature Σ , a **derived operation** $\gamma : s_1 \dots s_n \rightarrow s$ of Σ is a term in $T_{\Sigma, s}(\{z_1, \dots, z_n\})$, where z_1, \dots, z_n are special variables of sorts s_1, \dots, s_n . For any Σ -algebra A , the interpretation of γ in A is the map $\gamma_A : A_{s_1} \times \dots \times A_{s_n} \rightarrow A_s$ defined as $\gamma_A(a_1, \dots, a_n) = \theta^*(\gamma)$, where $\theta : \{z_1, \dots, z_n\} \rightarrow A$ takes z_i to a_i for all $i = 1 \dots n$. We let $Der(\Sigma)$ denote the signature of all derived operations of Σ .

It can be easily seen that $\mathcal{B}' = (Der(\Sigma), \Gamma, E)$ is a conservative extension of \mathcal{B} . The following result reformulates a result in [21]:

Proposition 15. *Given a behavioral specification $\mathcal{B} = (\Sigma, \Gamma, E)$, then $\Delta = \Gamma$ is a cobasis of \mathcal{B} , where \mathcal{B}' can be either \mathcal{B} or $(Der(\Sigma), \Gamma, E)$.*

The notion of equivalent behavioral specifications has been introduced in [21] to formalize the intuitive process of reducing the number of behavioral operations:

Definition 16. $\mathcal{B}_1 = (\Sigma, \Gamma_1, E_1)$ and $\mathcal{B}_2 = (\Sigma, \Gamma_2, E_2)$ are **equivalent** iff they have the same models (hidden algebras) with the same behavioral equivalence. We write it $\mathcal{B}_1 \sim \mathcal{B}_2$.

Notice that \sim is indeed an equivalence relation on behavioral specifications, that \mathcal{B}' is a conservative extension of \mathcal{B} whenever $\mathcal{B} \sim \mathcal{B}'$, and that \mathcal{B}'' is a conservative extension of \mathcal{B} whenever \mathcal{B}' is a conservative extension of \mathcal{B} and $\mathcal{B}' \sim \mathcal{B}''$.

Proposition 17. *If $\mathcal{B}_1 = (\Sigma, \Gamma_1, E)$ and $\mathcal{B}_2 = (\Sigma, \Gamma_2, E)$ are behavioral specifications such that $\Gamma_2 \subseteq \Gamma_1$ and all operations in $\Gamma_1 - \Gamma_2$ are behaviorally congruent for \mathcal{B}_2 , then:*

1. $\mathcal{B}_1 \sim \mathcal{B}_2$, and
2. Γ_2 is a cobasis of \mathcal{B}_1 .

Proof. 1. A more general result is proved in [21], namely, given $\mathcal{B}_1 = (\Sigma, \Gamma_1, E)$ and $\mathcal{B}_2 = (\Sigma, \Gamma_2, E)$ with $\Gamma_2 \subseteq \Gamma_1$ such that all operations in $\Gamma_1 - \Gamma_2$ are behaviorally congruent for \mathcal{B}_2 and there are no conditional equations with hidden sorted conditions in E , then $\mathcal{B}_1 \sim \mathcal{B}_2$.

Our result follows immediately now, because we assumed that there are no conditional equations in E .

2. Since $\mathcal{B}_1 \sim \mathcal{B}_2$, one gets that \mathcal{B}_2 is a conservative extension of \mathcal{B}_1 . By Proposition 15, Γ_2 is a cobasis of \mathcal{B}_2 , so by 1., one can easily get that Γ_2 is a cobasis of \mathcal{B}_1 .

Therefore, fewer behavioral operations are enough to generate the behavioral equivalence whenever the congruence of the others can be proved. However, proving the behavioral congruence of operations is not easy and might require coinduction as in Example 6. Fortunately, congruence criteria which work in all practical situations we are aware of so far have been developed [2, 21]; the following theorem seems to be a key result (Theorem 16 in [21]):

Theorem 18. *If Δ is a cobasis of \mathcal{B} and for each appropriate $\delta: s_1 \dots s_n \rightarrow s$ in Δ there is some γ in $T_{\star}(Z_j \cup W)$ such that⁶ $\mathcal{B}' \models (\forall Z_j, W) \delta(Z_j, \sigma(W)) = \gamma$ for $j = 1, \dots, n$, then σ is behaviorally congruent for \mathcal{B} .*

The following criterion [21] covers most of the situations and it follows easily from the theorem above:

Congruence Criterion: Let $\mathcal{B} = (\Sigma, \Gamma, E)$ be a hidden specification and σ be an operation in Σ . If for each appropriate δ in Γ there is some γ in $T_{\star}(Z_j \cup W)$ such that the Σ -equation $(\forall Z_j, W) \delta(Z_j, \sigma(W)) = \gamma$ is in E (modulo renaming of variables), then σ is behaviorally congruent for \mathcal{B} .

Example 5. Sets: The difference between the next behavioral specification and SET1 is that in is the only behavioral operation:

```

mod SET2 { pr(NAT)
  * [ Set ] *
  op empty : -> Set
  op { _ } : Nat -> Set
  bop _in_ : Nat Set -> Bool
  op _U_ : Set Set -> Set
  op _&_ : Set Set -> Set
  op not_ : Set -> Set

  vars N N' : Nat   vars S S' : Set

  eq N in empty = false .
  eq N in { N' } = (N == N') .
  eq N in S U S' = (N in S) or (N in S') .
  eq N in S & S' = (N in S) and (N in S') .
  eq N in not S = not (N in S) . }

```

⁶ Z_j is the set of variables $\{z_1 : s_1, \dots, z_{j-1} : s_{j-1}, z_{j+1} : s_{j+1}, \dots, z_n : s_n\}$ and $\delta(Z_j, t)$ is the term $\delta(z_1, \dots, z_{j-1}, t, z_{j+1}, \dots, z_n)$.

The congruence criterion implies that $_U_$, $_&_$ and not are behaviorally congruent for **SET2**. Consequently, by Proposition 17, **SET1** and **SET2** are equivalent and $\Delta = \{\text{in}\}$ is a cobasis of **SET1**.

Example 6. Streams: Let us consider the following behavioral specification of streams which differs from **STREAM1** in Example 3 in that only **head** and **tail** are behavioral:

```

mod STREAM2 { pr(NAT)
  * [ Stream ] *
  op zero : -> Stream
  op cons  : Nat Stream -> Stream
  bop head_ : Stream -> Nat
  bop tail_ : Stream -> Stream
  op zip    : Stream Stream -> Stream

  var N : Nat      vars S S' : Stream
  eq head zero = 0 .
  beq tail zero = zero .

  eq head cons(N, S) = N .
  beq tail cons(N, S) = S .

  eq head zip(S, S') = head(S) .
  beq tail zip(S, S') = zip(S', tail(S)) . }

```

By the congruence criterion above, **cons** is behaviorally congruent for **STREAM2**. The criterion cannot be applied for **zip** because the last equation contains **zip** (which is not declared behavioral) in both terms. However, we can show that **zip** is also behaviorally congruent. Let Σ_1 and Σ_2 be the hidden signatures of **STREAM1** and **STREAM2**, respectively, let Γ_1 and Γ_2 be the hidden subsignatures of Σ_1 and Σ_2 containing the behavioral operations $\{\text{cons}, \text{head}, \text{tail}, \text{zip}\}$ and $\{\text{head}, \text{tail}\}$, respectively, and let A be a model (hidden Σ_2 -algebra) of **STREAM2**. Notice that for any streams a, a' in A , $a \equiv_{\Sigma_2}^* a'$ iff $A_{\text{head}}(A_{\text{tail}}^n(a)) = A_{\text{head}}(A_{\text{tail}}^n(a'))$ for all $n \geq 0$ (it can be proved eventually using Theorem 3). Now, suppose that $a_1 \equiv_{\Sigma_2}^* a'_1$ and $a_2 \equiv_{\Sigma_2}^* a'_2$. It can be shown by induction that $A_{\text{tail}}^{2k}(A_{\text{zip}}(a_1, a_2)) \equiv_{\Sigma_2}^* A_{\text{zip}}(A_{\text{tail}}^k(a_1), A_{\text{tail}}^k(a_2))$ and $A_{\text{tail}}^{2k+1}(A_{\text{zip}}(a_1, a_2)) \equiv_{\Sigma_2}^* A_{\text{zip}}(A_{\text{tail}}^{k+1}(a_2), A_{\text{tail}}^k(a_1))$, and similarly for a'_1 and a'_2 , for all $k \geq 0$. Hence $A_{\text{head}}(A_{\text{tail}}^{2k}(A_{\text{zip}}(a_1, a_2))) = A_{\text{head}}(A_{\text{tail}}^k(a_1))$ and $A_{\text{head}}(A_{\text{tail}}^{2k+1}(A_{\text{zip}}(a_1, a_2))) = A_{\text{head}}(A_{\text{tail}}^k(a_2))$, and similarly for a'_1 and a'_2 . Thus $A_{\text{head}}(A_{\text{tail}}^n(A_{\text{zip}}(a_1, a_2))) = A_{\text{head}}(A_{\text{tail}}^n(A_{\text{zip}}(a'_1, a'_2)))$ for all $n \geq 0$, that is, $A_{\text{zip}}(a_1, a_2) \equiv_{\Sigma_2}^* A_{\text{zip}}(a'_1, a'_2)$. Therefore, **zip** is behaviorally congruent for **STREAM2**. By Proposition 17, **STREAM1** and **STREAM2** are equivalent and $\Delta = \{\text{head}, \text{tail}\}$ is a cobasis of **STREAM1**.

Therefore, there are examples of congruent operations which do not fall under the above criterion. Fortunately, a more general criterion is introduced in [2] which works on the above example. We do not discuss this criterion in the present paper, but it is worth noting that it follows (modulo the lemma of Theorem 4.5 in [2]) as a consequence of Theorem 18, applied for the cobasis containing exactly the “observable contexts”.

Given a cobasis Δ of \mathcal{B} , the following rule called Δ -coinduction is sound for behavioral satisfaction:

$$(6) \Delta - \text{Coinduction} : \frac{(\forall W, X) \delta(W, t) = \delta(W, t') \text{ for all appropriate } \delta \in \Delta}{(\forall X) t = t'}$$

The fewer operations Δ has, the easier the Δ -coinduction is. Notice that this rule cannot be applied when Δ has an infinite number of operations. We call it Δ -coinduction because a special coinduction is intended when $\mathcal{B}' = \mathcal{B}$ and $\Delta = \Gamma$. More precisely, given a hidden Σ -algebra A , consider all relations R on A having the property that $a R a'$ iff $\delta(a_1, \dots, a_n, a) R \delta(a_1, \dots, a_n, a')$ for all appropriate $\delta \in \Delta (= \Gamma)$ and $a_1, \dots, a_n \in A$. Obviously, all operations in Δ are congruent for all relations R as above, so R is a hidden Γ -congruence and by Theorem 3, $R \subseteq \equiv_{\Sigma}^*$. Notice that \equiv_{Σ}^* is itself an R as above.

Proposition 19. Define \Vdash_{Δ} by $\mathcal{B} \Vdash_{\Delta} (\forall X) t = t'$ iff $(\forall X) t = t'$ is derivable from \mathcal{B} under the rules (1)–(6). Then \Vdash_{Δ} is sound for behavioral satisfaction if Δ is a cobasis of \mathcal{B} .

Notice that \Vdash_{Δ} 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*. Kumo [10] implements this special case, and will soon implement the coinduction rule (6) in its general form.

Example 7. Sets: Since $\Delta = \{\text{in}\}$ is a cobasis of SET1 (see Example 5), in order to prove that two sets are behaviorally equivalent it suffices to show that they cannot be distinguished under in . In this way, one can prove the idempotency of union and intersection, their commutativity, associativity, distributivity and De Morgan laws. For example, let us show the distributivity of intersection, that is, $(\forall S, S', S'') S \ \& \ (S' \cup S'') = (S \ \& \ S') \cup (S \ \& \ S'')$. We can show that a natural number is in the left hand side set iff it is in the right hand side set with the following CafeOBJ proof score:

```
mod DISTR { pr(SET2)  op n : -> Nat  ops s s' s'' : -> Set }

open DISTR .
  eq n in s = true .
  red n in s & (s' U s'') == n in (s & s') U (s & s'') . ** true
close

open DISTR .
  eq n in s = false .
  red n in s & (s' U s'') == n in (s & s') U (s & s'') . ** true
close
```

We will see in the next section that this proof (and many others) can be done automatically by behavioral coinductive rewriting.

Example 8. Streams: Now we can prove that STREAM1 (see Example 3) satisfies behaviorally $(\forall S) \text{cons}(\text{head}(S), \text{tail}(S)) = S$ and $(\forall N, S, S') \text{zip}(\text{cons}(N, S), S') = \text{cons}(N, \text{zip}(S', S))$. We saw in Example 6 that $\Delta = \{\text{head}, \text{tail}\}$ is a cobasis of STREAM1.

By the first five inference rules, $\text{STREAM1} \Vdash_{\Delta} (\forall S) \text{head}(\text{cons}(\text{head}(S), \text{tail}(S))) = \text{head}(S)$ and $\text{STREAM1} \Vdash_{\Delta} (\forall S) \text{tail}(\text{cons}(\text{head}(S), \text{tail}(S))) = \text{tail}(S)$. Then by Δ -coinduction, $\text{STREAM1} \Vdash_{\Delta} (\forall S) \text{cons}(\text{head}(S), \text{tail}(S)) = S$.

Similarly, $\text{STREAM1} \Vdash_{\Delta} (\forall S) \text{head}(\text{zip}(\text{cons}(N, S), S')) = \text{head}(\text{cons}(N, \text{zip}(S, S')))$ and also $\text{STREAM1} \Vdash_{\Delta} (\forall S) \text{tail}(\text{zip}(\text{cons}(N, S), S')) = \text{tail}(\text{cons}(N, \text{zip}(S, S')))$. Then by Δ -coinduction, $\text{STREAM1} \Vdash_{\Delta} (\forall N, S, S') \text{zip}(\text{cons}(N, S), S') = \text{cons}(N, \text{zip}(S', S))$.

We will see in the next section that these properties can be proved automatically by behavioral coinductive rewriting.

6 Behavioral Coinductive Rewriting

Unlike standard term rewriting or behavioral rewriting, behavioral coinductive rewriting rewrites finite sets of pairs of terms, called *goals*, instead of just terms. Let T be the set of terms with variables, and let $Goal$ be the set of finite subsets of $T \times T$. Also, let \square be the diagonal relation on $Goal$, containing exactly the pairs (G, G) for all goals G . A goal is *trivial* iff it contains only pairs of equal terms.

To make the presentation simpler, we consider that Δ contains only derived operations over Γ , that is, from now on in the paper we work under the following

Assumption: Δ is a cobasis of \mathcal{B} with $\mathcal{B}' = (\text{Der}(\Sigma), \Gamma, E)$ and $\Delta \subseteq \text{Der}(\Gamma)$.

Now we define two relations on $Goal$:

Rewrite: $G \rightrightarrows_r G'$ iff G' replaces each $(t, t') \in G$ by some (s, s') such that $t \rightrightarrows^* s$ and $t' \rightrightarrows^* s'$

Expand: $G \rightrightarrows_e G'$ iff $G' = \{(\delta(W, t), \delta(W, t')) \mid \delta \text{ appropriate in } \Delta\} \cup (G - \{(t, t')\})$,
where $(t, t') \in G$ and $\text{sort}(t, t') \in H$.

Definition 20. Let \rightrightarrows be the relation $(\rightrightarrows_r \cup \rightrightarrows_e)^*$. If $G \rightrightarrows G'$ then we say that G **behaviorally coinductively rewrites to** G' .

Proposition 21. *The following hold:*

1. $\rightrightarrows_r; \rightrightarrows_r = \rightrightarrows_r$,
2. $\rightrightarrows_r; \rightrightarrows_e \subseteq \rightrightarrows_e; \rightrightarrows_r$,
3. \rightrightarrows is equal to $\rightrightarrows_e^*; \rightrightarrows_r$,
4. $\leftarrow_e; \rightrightarrows_r \subseteq \rightrightarrows_r; \leftarrow_e$,
5. $\leftarrow_e; \rightrightarrows_e \subseteq \square \cup (\rightrightarrows_e; \leftarrow_e)$,
6. $\leftarrow_e; \rightrightarrows_e \subseteq (\square \cup \rightrightarrows_e); \leftarrow_e$,
7. If \rightrightarrows is confluent then \rightrightarrows_r is strongly confluent, that is, $\leftarrow_r; \rightrightarrows_r \subseteq \rightrightarrows_r; \leftarrow_r$,
8. If \rightrightarrows is confluent then $\leftarrow_e; \rightrightarrows_r \subseteq \rightrightarrows_r; \leftarrow_e$.

Proof. 1. It follows by the transitivity of \rightrightarrows^* .

2. Suppose that $G \rightrightarrows_r G_1 \rightrightarrows_e G'$ with $G' = \{(\delta(W, s), \delta(W, s')) \mid \delta \in \Delta\} \cup (G_1 - \{(s, s')\})$, where $(s, s') \in G_1$ such that $t \rightrightarrows^* s$ and $t' \rightrightarrows^* s'$ for some $(t, t') \in G$. Then take $G_2 = \{(\delta(W, t), \delta(W, t')) \mid \delta \in \Delta\} \cup (G - \{(t, t')\})$ and notice that $G_2 \rightrightarrows_r G'$. Hence $G (\rightrightarrows_e; \rightrightarrows_r) G'$.
3. Obviously, $\rightrightarrows_e^*; \rightrightarrows_r \subseteq (\rightrightarrows_e \cup \rightrightarrows_r)^* = \rightrightarrows$. On the other hand, we can first prove by induction that $(\rightrightarrows_e \cup \rightrightarrows_r)^n \subseteq (\square \cup \rightrightarrows_e)^n; \rightrightarrows_r$. It is immediate for $n = 0$. Suppose that it is true for n and let us prove it for $n + 1$. By 1, 2, and induction hypothesis,

$$\begin{aligned}
(\rightrightarrows_e \cup \rightrightarrows_r)^{n+1} &\subseteq (\square \cup \rightrightarrows_e)^n; \rightrightarrows_r; (\rightrightarrows_e \cup \rightrightarrows_r) \\
&= (\square \cup \rightrightarrows_e)^n; (\rightrightarrows_r; \rightrightarrows_e \cup \rightrightarrows_r) \\
&\subseteq (\square \cup \rightrightarrows_e)^n; (\rightrightarrows_e; \rightrightarrows_r \cup \rightrightarrows_r) \\
&= (\square \cup \rightrightarrows_e)^{n+1}; \rightrightarrows_r
\end{aligned}$$

Therefore, $\rightrightarrows = (\rightrightarrows_e \cup \rightrightarrows_r)^* \subseteq (\square \cup \rightrightarrows_e)^*; \rightrightarrows_r$. But $(\square \cup \rightrightarrows_e)^* = \rightrightarrows_e^*$.

4. Suppose that $G \rightrightarrows_e G_1$ and $G \rightrightarrows_r G_2$, with $G_1 = \{(\delta(W, t), \delta(W, t')) \mid \delta \in \Delta\} \cup (G - \{(t, t')\})$ where $(t, t') \in G$, and $t \rightrightarrows^* s$, $t' \rightrightarrows^* s'$, and $(s, s') \in G_2$. Take $G' = \{(\delta(W, s), \delta(W, s')) \mid \delta \in \Delta\} \cup (G_2 - \{(s, s')\})$ and notice that $G_1 \rightrightarrows_r G'$ and $G_2 \rightrightarrows_e G'$.
5. Suppose that $G_1 \leftarrow_e G \rightrightarrows_e G_2$ with $G_1 = \{(\delta_1(W, t_1), \delta_1(W, t'_1)) \mid \delta_1 \in \Delta\} \cup (G - \{(t_1, t'_1)\})$ and $G_2 = \{(\delta_2(W, t_2), \delta_2(W, t'_2)) \mid \delta_2 \in \Delta\} \cup (G - \{(t_2, t'_2)\})$, where $(t_1, t'_1), (t_2, t'_2) \in G$. If $(t_1, t'_1) = (t_2, t'_2)$ then $G_1 = G_2$, that is, $(G_1, G_2) \in \square$. If $(t_1, t'_1) \neq (t_2, t'_2)$ then let G' be the goal containing $\{(\delta_1(W, t_1), \delta_1(W, t'_1)) \mid \delta_1 \in \Delta\}$, $\{(\delta_2(W, t_2), \delta_2(W, t'_2)) \mid \delta_2 \in \Delta\}$ and $(G - \{(t_1, t'_1)\} - \{(t_2, t'_2)\})$. Now, notice that $G_1 \rightrightarrows_e G' \leftarrow_e G_2$.
6. We first prove by induction that $\leftarrow_r; (\square \cup \leftarrow_e)^n; \rightrightarrows_e \subseteq (\square \cup \rightrightarrows_e); \leftarrow_r; (\square \cup \leftarrow_e)^n$. If $n = 0$ then it follows by 4. Suppose that it is true for n and prove it for $n + 1$. By 5 and induction hypothesis,

$$\begin{aligned}
\leftarrow_r; (\square \cup \leftarrow_e)^{n+1}; \rightrightarrows_e &= \leftarrow_r; (\square \cup \leftarrow_e)^n; (\rightrightarrows_e \cup \leftarrow_e; \rightrightarrows_e) \\
&\subseteq \leftarrow_r; (\square \cup \leftarrow_e)^n; (\rightrightarrows_e \cup \square \cup \rightrightarrows_e; \leftarrow_e) \\
&= \leftarrow_r; (\square \cup \leftarrow_e)^n; \rightrightarrows_e; (\square \cup \leftarrow_e) \cup \leftarrow_r; (\square \cup \leftarrow_e)^n \\
&\subseteq (\square \cup \rightrightarrows_e); \leftarrow_r; (\square \cup \leftarrow_e)^{n+1} \cup \leftarrow_r; (\square \cup \leftarrow_e)^n \\
&= ((\square \cup \rightrightarrows_e) \cup \square); \leftarrow_r; (\square \cup \leftarrow_e)^n; ((\square \cup \leftarrow_e) \cup \square) \\
&= (\square \cup \rightrightarrows_e); \leftarrow_r; (\square \cup \leftarrow_e)^{n+1}.
\end{aligned}$$

Thus, $\leftarrow_r; (\square \cup \leftarrow_e)^*; \rightrightarrows_e \subseteq (\square \cup \rightrightarrows_e); \leftarrow_r; (\square \cup \leftarrow_e)^*$. But $(\square \cup \leftarrow_e)^* = \leftarrow_e^*$. Since by 3, $\leftarrow_e = \leftarrow_r; \leftarrow_e^*$, one gets $\leftarrow_e; \rightrightarrows_e \subseteq (\square \cup \rightrightarrows_e); \rightrightarrows$.

7. It is immediate, because \rightrightarrows confluent implies \rightrightarrows^* strongly confluent.

8. It can be easily seen that 4 implies $\Leftarrow_e^*; \Rightarrow_r \subseteq \Rightarrow_r; \Leftarrow_e^*$. Then by 7,

$$\begin{aligned} \Leftarrow; \Rightarrow_r &= \Leftarrow_r; \Leftarrow_e^*; \Rightarrow_r \\ &\subseteq \Leftarrow_r; \Rightarrow_r; \Leftarrow_e^* \\ &\subseteq \Rightarrow_r; \Leftarrow_r; \Leftarrow_e^* \\ &= \Rightarrow_r; \Leftarrow. \end{aligned}$$

Theorem 22. *If \Rightarrow is confluent then \Rightarrow is strongly confluent.*

Proof. Notice that 6 in Proposition 21 yields $\Leftarrow; \Rightarrow_e^* \subseteq (\square \cup \Rightarrow_e)^*; \Leftarrow$. Since $(\square \cup \Rightarrow_e)^* = \Rightarrow_e^*$, one gets that $\Leftarrow; \Rightarrow_e^* \subseteq \Rightarrow_e^*; \Leftarrow$. By 3 and 8 in Proposition 21,

$$\begin{aligned} \Leftarrow; \Rightarrow &= \Leftarrow; \Rightarrow_e^*; \Rightarrow_r \\ &\subseteq \Rightarrow_e^*; \Leftarrow; \Rightarrow_r \\ &\subseteq \Rightarrow_e^*; \Rightarrow_r; \Leftarrow \\ &= \Rightarrow; \Leftarrow, \end{aligned}$$

that is, \Rightarrow is strongly confluent.

Definition 23. Let \Downarrow be the relation on terms defined as $t \Downarrow t'$ iff $\{(t, t')\} \Rightarrow \text{triv}$ for some trivial goal triv . Also, let \Downarrow_n be the subrelation of \Downarrow defined as $t \Downarrow_n t'$ iff $\{(t, t')\} ((\square \cup \Rightarrow_e)^n; \Rightarrow_r) \text{triv}$ for some trivial goal triv .

In other words, $t \Downarrow_n t'$ iff the goal $\{(t, t')\}$ can be reduced to a trivial goal doing at most n expansions. Obviously, $\Downarrow_n \subseteq \Downarrow_{n+1}$ for all $n \geq 0$.

Proposition 24. *Given two hidden terms t and t' :*

1. *If $t \Downarrow_n t'$ for some $n \geq 1$ then there is some $m < n$ such that $\delta(W, t) \Downarrow_m \delta(W, t')$ for all appropriate $\delta \in \Delta$,*
2. *$t \Downarrow t'$ iff $\delta(W, t) \Downarrow \delta(W, t')$ for all appropriate behavioral operations $\delta \in \Delta$.*

Proof. 1. Let $n_0 \leq n$ be the smallest number with $t \Downarrow_{n_0} t'$. If $n_0 = 0$ then $t(\Rightarrow^*; \Leftarrow^*)t'$. Then by Definition 7, $\delta(W, t)(\Rightarrow^*; \Leftarrow^*)\delta(W, t')$, that is, $\delta(W, t) \Downarrow_0 \delta(W, t')$. If $n_0 > 0$ then $\{(t, t')\} \Rightarrow_e G ((\square \cup \Rightarrow_e)^{n_0-1}; \Rightarrow_r) \text{triv}$ for some trivial goal triv , where G is the goal containing all pairs $(\delta(W, t), \delta(W, t'))$ for all appropriate $\delta \in \Delta$. Hence $\delta(W, t) \Downarrow_{n_0-1} \delta(W, t')$ for each appropriate $\delta \in \Delta$. Now take $m = n_0 - 1$.

2. If $t \Downarrow t'$ then there is some $n \geq 1$ such that⁷ $t \Downarrow_n t'$. By 1, there is some $m < n$ such that $\delta(W, t) \Downarrow_m \delta(W, t')$ for each appropriate $\delta \in \Delta$. Therefore, $\delta(W, t) \Downarrow \delta(W, t')$ for all appropriate $\delta \in \Delta$.

If $\delta(W, t) \Downarrow \delta(W, t')$ for all appropriate $\delta \in \Delta$, then $G \Rightarrow \text{triv}$ for some trivial goal triv , where G is the goal defined in 1. Since $\{(t, t')\} \Rightarrow_e G$, one gets that $\{(t, t')\} \Rightarrow \text{triv}$, that is, $t \Downarrow t'$.

Example 9. The inferences in Examples 7 and 8 say that $S \ \& \ (S' \cup S'') \Downarrow_1 (S \ \& \ S') \cup (S \ \& \ S'')$, $\text{cons}(\text{head}(S), \text{tail}(S)) \Downarrow_1 S$ and that $\text{zip}(\text{cons}(N, S), S') \Downarrow_1 \text{cons}(N, \text{zip}(S', S))$.

Lemma 25. *The following hold:*

1. *\Downarrow is reflexive,*
2. *\Downarrow is symmetric,*
3. *If \Rightarrow is confluent then \Downarrow is transitive,*
4. *If $t \Downarrow t'$ then*
 - *if $\text{sort}(t, t') \in V$ then $\sigma(W, t) \Downarrow \sigma(W, t')$ for all appropriate $\sigma \in \Sigma$,*
 - *if $\text{sort}(t, t') \in H$ then $\delta(W, t) \Downarrow \delta(W, t')$ for all appropriate $\delta \in \Delta$,*
5. *If $\delta(W, t) \Downarrow \delta(W, t')$ for all appropriate $\delta \in \Delta$, then $t \Downarrow t'$.*

Proof. 1. A goal $\{(t, t)\}$ is already trivial.

⁷ Notice that $\Downarrow_0 \subseteq \Downarrow_1$, so we can consider $n \geq 1$.

2. It is obvious because \Rightarrow_r and \Rightarrow_e do not make any distinction between the left term and the right term in any pair of terms.
3. Suppose that $t_1 \Downarrow t_2$ and $t_2 \Downarrow t_3$. Then there are some goals G_1 and G_2 and some trivial goals $triv_1$ and $triv_2$, such that $\{(t_1, t_2)\} \Rightarrow_e^* G_1 \Rightarrow_r triv_1$ and $\{(t_2, t_3)\} \Rightarrow_e^* G_1 \Rightarrow_r triv_2$. Let C_1 and C_2 be the sets of all Δ -terms $\gamma = \delta_1(W_1, \dots, \delta_k(W_k, z) \dots)$ with $k \geq 0$ and $\delta_1, \dots, \delta_k$ appropriate behavioral operations, such that $G_1 = \{(\gamma[t_1], \gamma[t_2]) \mid \gamma \in C_1\}$ and $G_2 = \{(\gamma[t_1], \gamma[t_2]) \mid \gamma \in C_2\}$, respectively, and let $G'_1 = \{(\gamma[t_2], \gamma[t_2]) \mid \gamma \in C_1\}$ and $G'_2 = \{(\gamma[t_2], \gamma[t_2]) \mid \gamma \in C_2\}$. Notice that $G'_1 \Leftarrow_e^* \{(t_2, t_2)\} \Rightarrow^* G'_2$, so by 4 in Proposition 21, there is some G'_3 such that $G'_1 \Rightarrow_e^* G'_3$ and $G'_2 \Rightarrow_e^* G'_3$. Let C_3 be the sets of all Δ -terms $\gamma = \delta_1(W_1, \dots, \delta_k(W_k, z) \dots)$ such that $G'_3 = \{(\gamma[t_2], \gamma[t_2]) \mid \gamma \in C_3\}$, and let $G_3 = \{(\gamma[t_1], \gamma[t_3]) \mid \gamma \in C_3\}$, $G_4 = \{(\gamma[t_1], \gamma[t_2]) \mid \gamma \in C_3\}$, and $G_5 = \{(\gamma[t_2], \gamma[t_3]) \mid \gamma \in C_3\}$. Notice that $G_1 \Rightarrow_e^* G_4$, $G_2 \Rightarrow_e^* G_5$, and $\{(t_1, t_3)\} \Rightarrow_e^* G_3$. By 3 in Proposition 21, there are some trivial goals $triv_4$ and $triv_5$ such that $G_4 \Rightarrow_r^* triv_4$ and $G_5 \Rightarrow_r^* triv_5$. Then, for each $\gamma \in C_3$ there are some s_1 and s_2 such that $\gamma[t_1] \Rightarrow^* s_1$, $\gamma[t_2] \Rightarrow^* s_2$, and $\gamma[t_2] \Rightarrow^* s_2$, $\gamma[t_3] \Rightarrow^* s_2$. Since \Rightarrow is confluent, there is some s_3 such that $s_1 \Rightarrow^* s_3$ and $s_2 \Rightarrow^* s_3$. Thus, $\gamma[t_1] \Rightarrow^* s_3$ and $\gamma[t_3] \Rightarrow^* s_3$, that is, there is some trivial goal $triv_3$ such that $G_3 \Rightarrow_r triv_3$. Therefore, $\{(t_1, t_3)\} \Rightarrow_e^* G_3 \Rightarrow_r triv_3$, that is, $t_1 \Downarrow t_3$.
4. If $sort(t, t') \in V$ then \Rightarrow_e cannot be applied at all in the reduction of $\{(t, t')\}$ to a trivial goal, so $t \Downarrow_0 t'$. Therefore, there is some t'' such that $t \Rightarrow^* t''$ and $t' \Rightarrow^* t''$. Since $sort(t, t', t'')$ is visible, by Definition 7, $\sigma(W, t) \Rightarrow^* \sigma(W, t'')$ and $\sigma(W, t') \Rightarrow^* \sigma(W, t'')$ for all $\sigma \in \Sigma$, so $\sigma(W, t) \Downarrow_0 \sigma(W, t')$.
If $sort(t, t') \in H$ then it follows by 2 in Proposition 24.
5. It follows by 2 in Proposition 24.

Theorem 26. *Let $(\forall X) t = t'$ be an equation. Then*

1. $t \Downarrow t'$ implies $\Vdash_{\Delta} (\forall X) t = t'$, and
2. If \Rightarrow is confluent then $\Vdash_{\Delta} (\forall X) t = t'$ implies $t \Downarrow t'$.

Proof. 1. We can prove a more general result, namely, if $G \Rightarrow G'$ and every pair (s, s') in G' is derivable (that is, $\Vdash_{\Delta} (\forall X) s = s'$), then every equation in G is derivable. To do that, it suffices to prove that the pairs in G are derivable if the pairs in G' are derivable, whenever $G \Rightarrow_r G'$ or $G \Rightarrow_e G'$. Suppose that $G \Rightarrow_r G'$ and that $\Vdash_{\Delta} (\forall X) s = s'$ for all $(s, s') \in G'$. By Proposition 8, $\Vdash (\forall X) t = s$ and $\Vdash (\forall X) t' = s'$, where $(t, t') \in G$ with $t \Rightarrow^* s$ and $t' \Rightarrow^* s'$. Since $\Vdash \subseteq \Vdash_{\Delta}$ and \Vdash_{Δ} is transitive, one gets that $\Vdash_{\Delta} (\forall X) t = t'$ for all $(t, t') \in G$.

2. It follows easily by induction on the length of the derivation of $(\forall X) t = t'$, using Lemma 25, observing that $\theta(l) \Downarrow \theta(r)$ whenever $(\forall Y) l = r$ is an equation in E .

Theorem 26 and Example 9 give other proofs for the fact that SET1 and STREAM1 satisfy behaviorally the properties in Examples 7 and 8, respectively.

Corollary 27. *If \Rightarrow is confluent then $\Vdash_{\Delta} (\forall X) t = t'$ iff $t \Downarrow t'$.*

6.1 Proving Behavioral Properties Automatically

In this section, we give an algorithm for proving behavioral equalities of terms. The algorithm takes two terms as input and if it terminates then it returns either *YES* or a goal. If it returns *YES* then it means that it succeeded in proving the behavioral equivalence of the two terms. If it returns a goal, it means that it didn't succeed in proving the behavioral equivalence of the two terms, but it reduced it to other behavioral equivalences. The user can then try to prove them as lemmas.

We first show the correctness of the algorithm, in the sense that if it returns *YES* for a pair of terms, then the two terms are indeed behaviorally equivalent. Then we investigate conditions under which the algorithm terminates, and finally conditions under which it is complete for \Vdash_{Δ} . There is little hope to find a complete algorithm for behavioral satisfaction in the general case; there haven't been found even complete deduction systems for hidden algebra so far.

Before we present the main algorithm, we should mention that Definition 23 suggests the following trivial algorithm: for each $n \geq 0$, check if $t \Downarrow_n t'$ by doing all n expansions followed by rewritings of all terms in all pairs to normal forms, then followed by removing all pairs of equal terms; stop if the goal becomes empty. By 1 in Theorem 26, this algorithm yields behavioral equivalence whenever it stops. But unfortunately, besides its inefficiency, it doesn't stop if the two input terms are not behaviorally equivalent. However, by 2 in Theorem 26 and soundness of \Vdash_Δ , it stops if \Rightarrow is canonical and the equation of the two terms is derivable with the six inference rules. Therefore, if \Rightarrow is canonical, then this algorithm is semi-decidable for \Vdash_Δ and there is little hope to improve it in this way.

Assumption: Δ is a cobasis of \mathcal{B} with $\Delta \subseteq \Gamma$.

Now, let us consider the following:

ALGORITHM $\mathcal{A}(t, t')$

1. initialize G with the pair $\{(t, t')\}$
2. rewrite all terms in all pairs in G to normal forms (\Rightarrow_r)
3. remove from G all pairs containing equal terms
4. **if** ($G = \emptyset$) **then return YES**
5. **if** (G has no hidden term rooted in $\Sigma - \Delta$) **then return G**
6. expand a pair in G containing a hidden term rooted in $\Sigma - \Delta$ (\Rightarrow_e)
7. **goto** 2.

Proposition 28. *If $\mathcal{A}(t, t')$ returns YES then $\Vdash_\Delta (\forall X) t = t'$.*

Proof. If $\mathcal{A}(t, t')$ returns YES, then there is some trivial goal $triv$ such that $(\{(t, t')\} \Rightarrow_r; (\Rightarrow_e; \Rightarrow_r)^* triv$. By 2 and 1 in Proposition 21, $\Rightarrow_r; (\Rightarrow_e; \Rightarrow_r)^* \subseteq \Rightarrow_e^*; \Rightarrow_r$, so by 3 in Proposition 21, $t \Downarrow t'$. Then by Theorem 26, $\Vdash_\Delta (\forall X) t = t'$.

Because of the soundness of \Vdash_Δ , \mathcal{A} is correct, in the sense that t and t' are behaviorally equivalent whenever $\mathcal{A}(t, t')$ returns YES. The following proposition gives a simple termination criterion for \mathcal{A} .

Proposition 29. Termination Criterion: *If \mathcal{R} terminates and each rewriting rule $(\forall X) l \rightarrow r$ has the property that r is a Δ -term and l is not a term $\delta(W)$ with $W \subseteq X$ and $\delta \in \Delta$, then \mathcal{A} terminates.*

Proof. We need the following

Lemma 30. *In the same context, if s is a normal form and $\delta(W', s) \Rightarrow^+ u$ for some $\delta \in \Delta$ and appropriate W' , then each subterm of u rooted in $\Sigma - \Delta$ is a proper subterm of s .*

Proof. The proof can be done by induction on the length of the derivation. If $\delta(W', s) \Rightarrow u$ then the only position where the rewriting can be done is at the top. Since the right hand side of every rewrite rule contains only behavioral operations, every subterm of u rooted in $\Sigma - \Delta$ is actually a subterm of s . On the other hand, s cannot be a subterm of u because there is no rewriting rule having $\delta(W)$ as its left hand side term.

Now, suppose that $\delta(W', s) \Rightarrow^+ u$, each subterm of u rooted in $\Sigma - \Delta$ is a proper subterm of s , and that $u \Rightarrow u'$. Since s is a normal form and each subterm of u rooted in $\Sigma - \Delta$ is a proper subterm of s , the position of u where the rewriting to u' is applied cannot be inside a subterm of u rooted in $\Sigma - \Delta$. Therefore, there are only behavioral operations on the path from that position to the root of u . Since the rewriting rule applied has only behavioral operations in the right hand side, it is easy to observe that all subterms of u' rooted in $\Sigma - \Delta$ are subterms of u . Thus, each subterm of u' rooted in $\Sigma - \Delta$ is a proper subterm of s .

Given a pair of terms (t, t') , $\mathcal{A}(t, t')$ terminates either when $G = \emptyset$ (step 4) or when both terms in any pair in G are distinct normal forms that have behavioral operations as roots (step 5). Suppose that $\mathcal{A}(t, t')$ does not terminate. Then for each $n \geq 1$, there is a sequence $(t_1, t'_1), (s_1, s'_1), \dots, (t_n, t'_n), (s_n, s'_n)$ such that $(t_1, t'_1) = (t, t')$, $(t_{i+1}, t'_{i+1}) = (\delta_i(W_i, s_i), \delta_i(W_i, s'_i))$ for some $\delta_i \in \Delta$, and s_i and s'_i are normal forms of t_i and t'_i such that at least one of s_i and s'_i is rooted in $\Sigma - \Delta$. By Lemma 30, there is an infinite sequence of normal forms rooted in $\Sigma - \Delta$, say s_{i_1}, s_{i_2}, \dots , such that $s_{i_{j+1}}$ is a proper subterm of s_{i_j} for all natural numbers j . Since any term has only a finite number of subterms, we deduce that $\mathcal{A}(t, t')$ terminates.

Example 10. Notice that Δ can contain hidden constants. This is even recommended in some situations to make the previous proposition applicable. For example, a Δ for NDSTACK in Example 1 for which the algorithm \mathcal{A} terminates is $\{\text{top}, \text{pop}, \text{empty}\}$.

Since $\Delta = \{\text{in}\}$ is a cobasis of SET1 (see Example 7) and all hypotheses in Proposition 29 are fulfilled, the algorithm \mathcal{A} for SET1 terminates.

Proposition 29 cannot be applied for STREAM1 with $\Delta = \{\text{head}, \text{tail}\}$ as zip appears in both terms of the last equation. However, the algorithm \mathcal{A} does not terminate for the input $(\text{zip}(S, S'), \text{zip}(S', S))$ as it generates infinite number of goals: $(\text{zip}(S', \text{tail}(S)), \text{zip}(S, \text{tail}(S')))$, followed by $(\text{zip}(\text{tail}(S), \text{tail}(S')), \text{zip}(\text{tail}(S'), \text{tail}(S)))$, and so on. Notice that \mathcal{A} does terminate if $\Delta = \{\text{head}, \text{tail}, \text{zip}\}$.

The next proposition provides a completeness criterion for \mathcal{A} with respect to the six rule inference system that generates \Vdash_{Δ} :

Proposition 31. Completeness Criterion: *Let \mathcal{R} be a behavioral rewriting system verifying the hypotheses in Proposition 29. If in addition,*

- \Rightarrow is confluent,
- for each $\delta, \delta' \in \Delta$ (not necessarily distinct) there is some $\delta_1 \in \Delta$ such that there is no rewriting rule with $\delta_1(W_1, \delta(W))$ or $\delta_1(W_1, \delta'(W))$ as its left hand side term,
- there is no rewriting rule $(\forall X) \delta(W, \delta'(t_1, \dots, t_n)) \rightarrow r$ with $W \subseteq X$ and $\delta, \delta' \in \Delta$ such that at least one of t_1, \dots, t_n is different from variables,

then $\Vdash_{\Delta} (\forall X) t = t'$ implies $\mathcal{A}(t, t')$ returns YES.

Proof. \mathcal{A} terminates by Proposition 29. If $\Vdash_{\Delta} (\forall X) t = t'$ then by 2 in Theorem 26, $t \Downarrow t'$. Suppose that $\mathcal{A}(t, t')$ does not return YES, that is, the returned goal contains some pairs of distinct normal forms (s, s') such that both s and s' are rooted in Δ . More precisely, there is a sequence $(t_1, t'_1), (s_1, s'_1), \dots, (t_n, t'_n), (s_n, s'_n)$ such that $(t_1, t'_1) = (t, t')$ and $(s_n, s'_n) = (s, s')$, s_i and s'_i are normal forms of t_i and t'_i , respectively, and for each $i = 0, \dots, n-1$, $(t_{i+1}, t'_{i+1}) = (\delta_i(W_i, s_i), \delta_i(W_i, s'_i))$ for some $\delta_i \in \Delta$.

Lemma 32. *If \Rightarrow is confluent, $t \Rightarrow^* s$ and $t' \Rightarrow^* s'$, then $t \Downarrow t'$ implies $s \Downarrow s'$.*

Proof. We can prove by well-founded induction on n that: for each $t \Rightarrow^* s$ and $t' \Rightarrow^* s'$, $t \Downarrow_n t'$ implies $s \Downarrow s'$. Suppose that the assertion is true for all $m < n$. Let $t \Rightarrow^* s$ and $t' \Rightarrow^* s'$ and $t \Downarrow_n t'$. If $n = 0$ then there is some s'' such that $t \Rightarrow^* s''$ and $t' \Rightarrow^* s''$. Since \Rightarrow is confluent, there are some terms s_1 and s_2 such that $s \Rightarrow^* s_1$, $s'' \Rightarrow^* s_1$, and $s' \Rightarrow^* s_2$, $s'' \Rightarrow^* s_2$. Furthermore, there is some term s_3 such that $s_1 \Rightarrow^* s_3$ and $s_2 \Rightarrow^* s_3$. Therefore, $s \Rightarrow^* s_3$ and $s' \Rightarrow^* s_3$, that is $s \Downarrow_0 s'$. Thus $s \Downarrow s'$. If $n \geq 1$ then by 1 in Proposition 24, there is some $m < n$ such that $\delta(W, t) \Downarrow_m \delta(W, t')$ for each appropriate $\delta \in \Delta$. By the induction hypothesis, $\delta(W, s) \Downarrow \delta(W, s')$, and by 2 in Proposition 24, $s \Downarrow s'$.

Since $t \Downarrow t'$, by Lemma 32, $s \Downarrow s'$; suppose that $s \Downarrow_n s'$. Let δ and δ' be the roots of s and s' , respectively, and let δ_1 be a behavioral operation in Δ such that $\delta_1(W_1, \delta(W))$ and $\delta_1(W_1, \delta'(W'))$ do not appear as left hand side terms of any rewriting rule in \mathcal{R} . Since s and s' are normal forms and there is no rewriting rule with $\delta_1(W_1, \delta(t_1, \dots, t_n))$ or $\delta_1(W_1, \delta'(t'_1, \dots, t'_n))$ as left hand side, $\delta_1(W_1, s)$ and $\delta_1(W_1, s')$ are also normal forms. By 1 in Proposition 24, there is some $n_1 < n$ such that

$\delta_1(W_1, s) \Downarrow_{n_1} \delta_1(W_1, s')$. Similarly, there is some $\delta_2 \in \Delta$ and $n_2 < n_1$ such that $\delta_2(W_2, \delta_1(W_1, s))$ and $\delta_2(W_2, \delta_1(W_1, s'))$ are normal forms and $\delta_2(W_2, \delta_1(W_1, s)) \Downarrow_{n_2} \delta_2(W_2, \delta_1(W_1, s'))$. Iterating this method, there are some $\delta_1, \dots, \delta_k \in \Delta$ such that γ_k and γ'_k are normal forms and $\gamma_k \Downarrow_0 \gamma'_k$, where $\gamma_k = \delta_k(W_k, \dots, \delta_1(W_1, s))$ and $\gamma'_k = \delta_k(W_k, \dots, \delta_1(W_1, s'))$. Since γ_k and γ'_k are already normal forms, one gets that $\gamma_k = \gamma'_k$, that is, $s = s'$. This is a contradiction because s and s' were supposed to be distinct. Consequently, $\mathcal{A}(t, t')$ returns *YES*.

Definition 33. We say that a behavioral rewriting system is **reasonable** iff it verifies the easy to check syntactic conditions in Proposition 29 and 31, that is,

- for each rewriting rule $(\forall X) l \rightarrow r$,
 - r is a Δ -term,
 - l is *not* a term $\delta(W)$ with $\delta \in \Delta$ and $W \subseteq X$,
 - l is *not* a term $\delta(W, \delta'(t_1, \dots, t_n))$ with $\delta, \delta' \in \Delta$ and the terms t_1, \dots, t_n are not all variables,
- for each $\delta, \delta' \in \Delta$ (not necessarily distinct), there is some $\delta_1 \in \Delta$ such that neither $\delta_1(W_1, \delta(W))$ nor $\delta_1(W_1, \delta'(W'))$ does appear as left hand side term of rewriting rules.

The algorithm \mathcal{A} described above terminates and is sound and complete for \Downarrow_{Δ} , whenever \mathcal{R} is reasonable and canonical:

Theorem 34. *If \mathcal{R} is a reasonable canonical behavioral rewriting system, then*

1. \mathcal{A} terminates, and
2. $\Downarrow_{\Delta} (\forall X) t = t'$ iff $\mathcal{A}(t, t') = \text{YES}$.

7 Conclusion and Future Work

Behavioral coinductive rewriting and its confluence, completeness and termination has been investigated in this paper. However, there is still much work left to do, including:

- find more general syntactic criteria for termination and completeness of the algorithm \mathcal{A} in subsection 6.1;
- allow both behavioral and strict equations;
- study the confluence and termination of the behavioral rewriting relation \Rightarrow in section 4;
- allow conditional equations;
- extend the results to general cobases in conservative extensions.

Acknowledgments

I would like to warmly thank Professor Joseph Goguen for his precious suggestions regarding previous drafts of this work. Special thanks to my colleagues Kai Lin and Bogdan Warinschi for useful and fruitful discussions on hidden subjects.

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-Verlag, 1998.
2. Michel Bidoit and Rolf Hennicker. Observer complete definitions are behaviourally coherent. In Kokichi Futatsugi, Joseph Goguen, and José Meseguer, editors, *OBJ/CafeOBJ/Maude at Formal Methods '99*, pages 83–94. Theta, 1999. Proceedings of a workshop held in Toulouse, France, 20th and 22nd September 1999.
3. 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.

4. Răzvan Diaconescu and Kokichi Futatsugi. Logical foundations of CafeOBJ, 1999. Submitted to Theoretical Computer Science.
5. Răzvan Diaconescu and Kokichi Futatsugi. Behavioral coherence in object-oriented algebraic specification. *Journal of Universal Computer Science*, 6(1):74–96, 2000. Also Japan Advanced Institute for Science and Technology Technical Report number IS-RR-98-0017F, 1998.
6. Marie-Claude Gaudel and Igor Privara. Context induction: an exercise. Technical Report 687, LRI, Université de Paris-Sud, 1991.
7. Joseph Goguen. *Theorem Proving and Algebra*. MIT, ?? to appear.
8. 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.
9. 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.
10. 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. Numazu, Japan, April 1998.
11. Joseph Goguen and Grant Malcolm. Proof of correctness of object representation. In A. William Roscoe, editor, *A Classical Mind: Essays in Honour of C.A.R. Hoare*, pages 119–142. Prentice-Hall, 1994.
12. Joseph Goguen and Grant Malcolm. Hidden coinduction: Behavioral correctness proofs for objects. *Mathematical Structures in Computer Science*, 9(3):287–319, 1999.
13. Joseph Goguen and Grant Malcolm. A hidden agenda. *Theoretical Computer Science*, to appear. Also UCSD Dept. Computer Science & Eng. Technical Report CS97-538, May 1997.
14. Joseph Goguen and Grigore Roşu. Hiding more of hidden algebra. In *FM'99 – Formal Methods*, pages 1704–1719. Springer, 1999. Lecture Notes in Computer Sciences, Volume 1709, Proceedings of World Congress on Formal Methods, Toulouse, France.
15. Lutz Hamel. *Behavioural Verification and Implementation of an Optimizing Compiler for OBJ3*. PhD thesis, Oxford University Computing Lab, 1996.
16. Rolf Hennicker. Context induction: a proof principle for behavioral abstractions. *Formal Aspects of Computing*, 3(4):326–345, 1991.
17. Rolf Hennicker and Michel Bidoit. Observational logic. In *Algebraic Methodology and Software Technology (AMAST'98)*, volume 1548 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 1999.
18. 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, 1997.
19. Peter Padawitz. Swinging data types: Syntax, semantics, and theory. In *Proceedings, WADT'95*, volume 1130 of *Lecture Notes in Computer Science*, pages 409–435. Springer, 1996.
20. Peter Padawitz. Towards the one-tiered design of data types and transition systems. In *Proceedings, WADT'97*, volume 1376 of *Lecture Notes in Computer Science*, pages 365–380. Springer, 1998.
21. Grigore Roşu and Joseph Goguen. Hidden congruent deduction. In Ricardo Caferra and Gernot Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, pages 252–267. Springer, 2000. Lecture Notes in Artificial Intelligence, Volume 1761; papers from a conference held in Vienna, November 1998.
22. J.J.M.M. Rutten. Universal coalgebra: a theory of systems. Technical Report CS-R9652, CWI, 1996.