Circular Coinduction*

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

Abstract. Circular coinduction is a new technique for behavioral reasoning that extends coinduction to specifications with circularities. We show that a congruence criterion due to Bidoit and Hennicker follows easily from circular coinduction, and we give some natural examples of circular coinductive proofs. A notation, called BOBJ, appropriate for our style of behavioral specification is also sketched. Finally, everything is conducted in a general framework that in a sense is the gcd of previous behavioral frameworks.

1 Introduction

This paper gives a new inference rule for behavioral reasoning. We call this rule circular Δ-coinduction because it handles some examples with circularities (i.e., infinite recursions) that could not be handled by our previous rules in [16, 9, 10, 15]; we will also call it $\Delta^{\circlearrowleft}$-coinduction.

In addition, this paper tries to provide a "greatest common divisor" (gcd) or "international" version of hidden algebra that captures what is most common among approaches developed in France, Japan, Germany and California, which are respectively called "observational logic" [12, 2, 1], "coherent hidden algebra" [3, 4, 13], "swinging types" [14], and "hidden algebra" [16, 9, 10, 15]. We will call this common approach "gcd hidden algebra" when being precise, and just "hidden algebra" for short. The fact that all major results hold in this approach helps to explain the remarkable fact that more or less the same results seem to hold in every approach.

Similarly, we suggest "behavioral equational logic" or "behavioral algebra" for a "least common multiple" (lcm) approach, i.e., for an umbrella name that can include all the variants. But because research in behavioral algebra is now very active, we cannot regard the meanings of technical terms in the various approaches, such as "observational," "coherent," "hidden," "behavioral," as entirely fixed, but should rather see them as evolving, often through beneficial mutual interactions.

After exploring how to prove the congruence of operations in [16], we became convinced that this does not differ essentially from proving other behavioral properties, except perhaps that it is usually easier. Also certain "coinductive patterns" that appeared in specifying operations inspired a congruence criterion

* Supported by NSF grant CCR-9901002.

** Also Fundamentals of Computing, Faculty of Mathematics, University of Bucharest, Romania.

that could automatically decide whether an operation is congruent [16]; moreover, this criterion followed from the Δ-coinduction rule that was strong enough for all proofs we knew at that time. But the fact that congruence of the `zip` operation of the `STREAM` example (in Section 3) does not follow from this criterion, suggests that more powerful deduction rules are needed.

Results in this paper arose in part through discussions with Michel Bidoit, Răzvan Diaconescu, Kokichi Futatsugi, and Rolf Hennicker at WDS'99 in Iaşi, WADT'99 in Bonas, and FM'99 in Toulouse. Bidoit and Hennicker [2] gave a general congruence criterion from which the congruence of `zip` followed easily, and so influenced by the relation between coinduction rules and congruence criteria found in [16], we sought a general inference rule from which the criterion in [2] would follow as naturally as our criterion in [16] followed from Δ-coinduction, and which could prove behavioral properties not provable by Δ-coinduction. The result of this search was circular Δ-coinduction[3].

2 GCD Hidden Algebra

Briefly, gcd hidden algebra is the logic (or better, the institution [7]) having equations as sentences, and algebras with behavioral equivalences as models, related by behavioral satisfaction. Unfortunately, the details are not so easy, for example because there are two choices for how to declare behavioral operations: as part of the signature; and as separate sentences. We considered both choices in [9], showing that there is an interesting morphism between the corresponding institutions, and also that each has certain disadvantages. In particular, the institution where behavioral operations are given in the signature and models are algebras contradicts the intuition that signatures should determine the syntax of terms, and is more complex technically, whereas the institution where each model has its own equivalence relation includes models that are not relevant. However, all approaches to behavioral algebra agree that specifications should have a signature with designated hidden sorts, some equations over that signature, and some "special" operations, variously called *observational* [12, 2], *behavioral* [3, 4, 9], or *destructor* [14]. This motivates the following:

Definition 1. A (**gcd**) **hidden signature** is a triple (S, H, Σ), often written just Σ, where S is a set of **sorts**, Σ is an S-sorted signature, and H is a subset of S of **hidden** sorts; call $V = S - H$ the **visible** sorts.

A (**gcd**) **hidden specification** is a triple (Σ, Γ, E), where Σ is a hidden signature, Γ is a subsignature of Σ, and E is a set of Σ-equations. We will use calligraphic letters $\mathcal{B}, \mathcal{B}_1$, etc. for behavioral specifications.

Given a Σ-algebra A with a binary relation $\sim$, an operation $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ is **congruent** (or **compatible** or **coherent**) for $\sim$ iff $A_\sigma(a_1, ..., a_n) \sim A_\sigma(a'_1, ..., a'_n)$ whenever $a_1 \sim a'_1$, ..., $a_n \sim a'_n$. Given $\Gamma \subseteq \Sigma$, a relation $\sim$ on A is a Γ-**congruence** iff each $\sigma \in \Gamma$ is congruent for $\sim$, and a Γ-congruence $\sim$ on A is called **hidden** iff it is the identity on visible sorts.

[3] We hope readers will find this an inspiring example of how science evolves, and of how scientific interaction can improve a subject.

The following very basic result, which seems to have first appeared in an early ancestor of [8], extends to gcd hidden algebra, with essentially the same proof as that given in [16], since nothing there relies on the fixed data algebra assumed in the framework of that paper.

Theorem 2. *Given $\Gamma \subseteq \Sigma$, there is a largest hidden Γ-congruence on any Σ-algebra A, called* Γ-**behavioral equivalence** *and denoted $\equiv^\Gamma_\Sigma$.*

The construction of $\equiv^\Gamma_\Sigma$ is natural: an (appropriate) Γ-**context**[4] of sort s is a term in $T_\Gamma(\{\star\} \cup Z)$ with exactly one occurrence of a special variable ("special" means it is different from any other variable in this situation) $\star$ of sort s, where Z is an infinite set of special variables. A **visible context** is a context with a visible result. Let $C_\Gamma[\star : s]$ denote the set of all visible Γ-contexts of sort s, and $var(c)$ the finite set of variables in c, excluding $\star$. Given a subsignature Γ of Σ and a Σ-algebra A, a Γ-context c determines a map $A_c\colon\ A_s \to (A^{var(c)} \to A)$ 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 $\theta(z')$. Then $\equiv^\Gamma_\Sigma$ is given by $a \equiv^\Gamma_\Sigma a'$ iff $A_c(a) = A_c(a')$ for all visible Γ-contexts c.

Perhaps the most definitive feature of behavioral algebra is its distinction between hidden and visible sorts, going back to [5]. It is this feature that supports the existence of the largest Γ-congruence of Theorem 2, and hence the notion of behavior, and it is mainly for this reason that we choose the name "hidden algebra" for our gcd approach.

Definition 3. An operation σ is Γ-**behaviorally congruent** for a Σ-algebra A iff σ is congruent for $\equiv^\Gamma_\Sigma$; we will often say just "congruent" instead of "behaviorally congruent". The algebra A Γ-**behaviorally satisfies** a Σ-equation $e = (\forall X)\ t = t'$ iff $\theta(t) \equiv^\Gamma_\Sigma \theta(t')$ for each $\theta\colon\ X \to A$; in this case we just write $A \models^\Gamma_\Sigma e$. If E is a set of Σ-equations, we write $A \models^\Gamma_\Sigma E$ iff A Γ-behaviorally satisfies each equation in E. When Σ, Γ and A are clear from context, we may write just $\equiv$ and $\models$ instead of $\equiv^\Gamma_\Sigma$ and $\models^\Gamma_\Sigma$ respectively. We say that A **behaviorally satisfies** (or **is a model of**) a behavioral specification $\mathcal{B} = (\Sigma, \Gamma, E)$ iff $A \models^\Gamma_\Sigma E$, and in this case we write $A \models \mathcal{B}$; also, $\mathcal{B} \models e$ means $A \models \mathcal{B}$ implies $A \models^\Gamma_\Sigma e$.

Among the various current approaches to behavioral algebra, observational logic is perhaps closest to gcd hidden algebra; the main difference is that the "observers" of observational logic (which correspond to our behavioral operations) are pairs (σ, i) with $\sigma\colon\ s_1 \ldots s_i \ldots s_n \to s$, which are declared in the signature. Because pairs (σ, i) and (σ, j) are allowed for $i \neq j$, there is greater expressive power; however we prefer the simpler notion until we have compelling examples where the extra generality is needed[5]. Coherent hidden algebra signatures differ

[4] Intuitively, contexts can be thought of as "experiments" on a system, consisting of a number of operations with hidden result followed by a "measurement" using some visible operation.

[5] However, we do have an example (the lambda calculus) where having this capability in cobases (see Section 2.2 for this concept) would reduce the proof obligations.

from those in observational logic and gcd hidden algebra in that only operations with *exactly* one hidden argument can be declared behavioral; however, we find this restriction unnecessary. The gcd hidden approach differs from our previous hidden algebra in that a fixed data algebra is no longer assumed; we felt more comfortable with this loss after we realized that the assumption was not needed for the soundness of Δ-coinduction, even though the distinctness of visible elements is needed in certain correctness proofs (e.g., that $1 \neq 0$ in the alternating bit protocol).

2.1 Behavioral Deduction

The following five inference rules from [16] are derived from ordinary equational logic, and all extend to gcd hidden algebra:

(1) Reflexivity : $\dfrac{}{(\forall X)\ t = t}$

(2) Symmetry : $\dfrac{(\forall X)\ t = t'}{(\forall X)\ t' = t}$

(3) Transitivity : $\dfrac{(\forall X)\ t = t',\ (\forall X)\ t' = t''}{(\forall X)\ t = t''}$

(4) Substitution : $\dfrac{(\forall Y)\ l = r \in E,\ \theta\colon Y \to T_\Sigma(X)}{(\forall X)\ \theta(l) = \theta(r)}$

(5) Congruence : $\left\{ \begin{array}{ll} a) & \dfrac{(\forall X)\ t = t',\ sort(t,t') \in V}{(\forall X, W)\ \sigma(W,t) = \sigma(W,t'),\ \text{for each } \sigma \in Der(\Sigma)} \\ b) & \dfrac{(\forall X)\ t = t',\ sort(t,t') \in H}{(\forall X, W)\ \delta(W,t) = \delta(W,t'),\ \text{for each congruent } \delta \in \Sigma} \end{array} \right\}$

If all operations are congruent then these rules become the usual rules of equational deduction. Notice that $(5b)$ only applies to congruent operations. Therefore, the more congruent operations there are, the more powerful the inference system becomes, and the more behavioral equalities can be proved. Because all behavioral operations are behaviorally congruent [16], we already have cases where $(5b)$ applies. Use of this method is facilitated by techniques for proving that operations are congruent, including easy to check syntactic criteria, given in [2, 9] and Section 4 below.

As a result, we suggest that specifications should declare as *many* operations as possible to be behavioral, preferably all of them. We believe the fact that when doing proofs, one wants as *few* behavioral operations as possible generating the intended behavioral equivalence should be considered a verification issue rather than a specification issue (see Section 2.2).

We also consider this approach easier to use in practice, and hence more suitable for ordinary software engineers and system designers. Behavioral speci-

fications are intended to capture and analyze the behavior of systems, or more exactly, their behavioral properties. Before writing a specification, there should be a pretty good feeling for what the system is intended to do and what it means for two states to be equivalent, e.g., "two sets are equivalent iff they have the same elements," or "order and multiple occurrences of elements do not matter," or "two streams are equivalent iff they have the same elements in the same order." Then the system designer should check whether the declared operations preserve this intended behavioral equivalence, which we believe is usually done informally in practice. The rare cases of non-behavior preserving operations should be marked with the attribute "`nbop`".

If it is not clear whether or not a certain operation should be behavioral, it is safer to let it be behavioral. The risk of letting an operation be behavioral instead of non-behavioral is that models will have a smaller behavioral equivalence, so that the specification will have fewer models, which is safer. There are two further possibilities: either the intended implementation is among those models, the fortunate case in which more properties can be proved about the system than if there were more models; or else the intended implementation is not among those models, in which case a careful rethinking is required to get a better understanding of the intended behavioral equivalence, and make some operations non-behavioral.

Therefore, by "behavioral operation" we mean an operation that preserves (or is compatible with) the intended behavior (or behavioral equivalence). We consider this approach simpler than the (actually technically equivalent) "indistinguishable under experiments" which suggests chosing a minimal set of "observers" and then distinguishing the behavior preserving (or coherent) operations from the non-behavior preserving ones; for us, this seems an unnecessary complication. In our terminology, both observers and coherent operations are behavioral.

A second motivation comes from the analogy (by duality) with standard equational specifications and induction, where for the same specification, different constructors (bases for induction) can be chosen depending on the proof goal. For example, induction schemas with zero and successor, or with zero, one and double successor, or with prime numbers and multiplication, are possible for the natural numbers. Similarly, different coinduction schemes can be chosen for the same behavioral specification. For example, in the case of infinite streams, there are at least the following two interesting cobases, $\Delta = \{\texttt{head}, \texttt{tail}\}$, and $\Delta' = \{\texttt{head}, \texttt{odd}, \texttt{even}\}$ (see Section 3). Therefore a specification should not impose an arbitrary choice of just one cobasis, but should rather allow any cobasis that is convenient. If the user is faithful to the methodology of declaring as many operations as possible to be behavioral, then (5*b*) can be modified to say "for each $\delta \in \Gamma$" instead of "for each congruent δ".

2.2 Cobases and Coinduction

The notion of cobasis was introduced in [16], but we soon realized it could be generalized (a common phenomenon in new areas of science) [9, 10, 15]. For this

reason, we will add the phrase "in the old sense" if we need to distinguish the cobases of [16] from the new ones.

Definition 4. Given a conservative extension $\mathcal{B}' = (\Sigma', \Gamma', E')$ of $\mathcal{B} = (\Sigma, \Gamma, E)$, and a subsignature Δ of Σ', let $T(\Gamma', \Delta; \star{:}h, Z)$ be the indexed set of all $(\Gamma' \cup \Delta)$-terms γ with variables in $\{\star\} \cup Z$, such that each subterm of γ rooted in any operation δ in Δ has exactly one occurrence of $\star$ which is an argument of δ, and these are the only occurrences of $\star$ in γ. We let $var(\gamma)$ denote the set of variables different from $\star$ in γ. Then Δ is a **cobasis for $\mathcal{B}$ in the old sense** iff for any Γ-context c over $\star$ of hidden sort h there is some γ in $T(\Gamma', \Delta; \star{:}h, var(c))$ such that $\mathcal{B}' \models (\forall \star, var(c))\ c = \gamma$.

This seemingly complex formulation includes all notions currently in use. We recall that a **complete set of observers** (in the sense of [2]) for Σ is a set of Γ-contexts, say Δ, such that for each visible Γ-context c there is some context $\delta \in \Delta$ which is a subcontext of c.

Proposition 5. *The following are all cobases (in the old sense):*

1. *The infinite set of all visible contexts*[6].
2. *Γ, the set of behavioral operations.*
3. *any complete set of observers in the sense of [2].*

Proof. The conservative extension $\mathcal{B}'$ of $\mathcal{B} = (\Sigma, \Gamma, E)$ is respectively:

1. $(Der(\Sigma), \Gamma, E)$. Notice that all visible contexts belong to $T(\Gamma', \Delta; \star{:}h, Z)$ in this case.
2. Either $\mathcal{B}$ or $(Der(\Sigma), \Gamma, E)$.
3. $(Der(\Sigma), \Gamma, E)$ is easy.

The main conceptual difference between cobasis in the old sense and the complete set of observers is that our notion relates to a behavioral specification rather than just a signature, which means that the equations can be also used in deducing that "for each visible Γ-context c there is some context $\delta \in \Delta$ which is a subcontext of c"; in addition, the form of equations in a specification might suggest getting rid of some contexts from the complete set of observers. Our more general notion of cobasis (see also [9, 10, 15]) is as follows:

Definition 6. If $\mathcal{B}' = (\Sigma', \Gamma', E')$ is a conservative extension of $\mathcal{B} = (\Sigma, \Gamma, E)$ and if $\Delta \subseteq \Sigma'$, then Δ is a **cobasis for** $\mathcal{B}$ iff for all hidden sorted terms $t, t' \in T_{\Sigma,h}(X)$, if $\mathcal{B}' \models (\forall W, X)\ \delta(W, t) = \delta(W, t')$ for all appropriate $\delta \in \Delta$ then $\mathcal{B} \models (\forall X)\ t = t'$.

The following result is proved in [16] (Theorem 12):

Theorem 7. *Every cobasis in the old sense is a cobasis.*

[6] This is the case of so- called context induction.

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

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

However, we show in the next section that these six rules are not powerful enough to prove certain simple behavioral properties, thus motivating the introduction of circular coinduction.

2.3 Congruent Operations and Equivalent Behavioral Specifications

The common interest in proving that operations are congruent [3, 2, 16] has a variety of motivations, depending on the methodological approach. Our approach emphasizes removing congruent operations from cobases, so that coinduction proofs can be as efficient as possible. We now remind the reader of the notion of equivalent behavioral specifications and some related results (for more detail, see [9, 10, 16]):

Definition 8. Behavioral specifications $\mathcal{B}_1 = (\Sigma, \Gamma_1, E_2)$ and $\mathcal{B}_2 = (\Sigma, \Gamma_2, E_2)$ are **behaviorally equivalent** iff they have the same models, each with the same behavioral equivalence; in this case, we write $\mathcal{B}_1 \equiv \mathcal{B}_2$.

Notice that $\equiv$ is an equivalence relation on behavioral specifications.

Theorem 9. *If $\mathcal{B}_1 = (\Sigma, \Gamma_1, E)$ and $\mathcal{B}_2 = (\Sigma, \Gamma_2, E)$ are behavioral specifications such that $\Gamma_1 \subseteq \Gamma_2$, and if E contains no equation with hidden sorted conditions, then $\mathcal{B}_1 \equiv \mathcal{B}_2$ iff all operations in $\Gamma_2 - \Gamma_1$ are behaviorally congruent for $\mathcal{B}_2$, and in this case, Γ_1 is a cobasis for $\mathcal{B}_2$.*

Therefore, if one suspects that a certain subset of behavioral operations $\Delta \subseteq \Gamma$ is a cobasis for $\mathcal{B} = (\Sigma, \Gamma, E)$, the following can be done:

1. Let $\mathcal{B}_\Delta = (\Sigma, \Delta, E)$.
2. Show that all operations in $\Gamma - \Delta$ are congruent for $\mathcal{B}_\Delta$.
3. Conclude that Δ is a cobasis for $\mathcal{B}$.

We will apply these steps implicitly in proofs below, without citing Theorem 9.

2.4 Specification Notation

The CafeOBJ language [4] was the first, and so far is the only, language to implement a version of behavioral algebra, and in particular, is the only language to support behavioral verification and the declaration of behavioral operations. We used CafeOBJ in several of our previous papers, but were criticized for doing so in a way that conflicted with the official CafeOBJ methodology. As a consequence of this and of the continuing evolution of our own ideas, we hope

it will be less confusing if we use a notation that corresponds more directly to our own current approach, and which in particular allows the declaration of behavioral operations with multiple hidden sorts.

The choice of syntax is often a sensitive issue, due in part to the difficulties of learning a new notation, and of shifting back and forth between notations. As a result, established user communities tend to be rather conservative about syntax. On the other hand, researchers have a natural desire to establish their own territories, and these are often marked by syntactic innovations. For reasons based in semiotics [6], as well perhaps as sentiment, we prefer a conservative approach with respect to the syntax of OBJ3 [11], and wish to depart from the conventions of OBJ3 only when there is a clear necessity for so doing. We call the resulting notation "BOBJ," from "behavioral OBJ." In BOBJ, behavioral specifications have the form

```
bth <NAME> is protecting DATA
  ...
end
```

where `DATA` is an imported theory of visible sorts, and all new sorts declared within the pattern are considered hidden. Operations are declared with the keyword `op`, and are considered behavioral, i.e., in Γ, unless they are given the attribute "`nbop`." This is justified by the design principle that the most common cases should be the simplest, noting that non-congruent operations are rare in behavioral theories. All equations are considered behavioral. Final periods are not needed if we assume that the user has taken some extra effort to prevent confusing the parser, for example, by using parentheses to separate the two terms of an equation if there are operations named `_eq_` or `_=_`. We can illustrate this syntax with the following behavioral specification for nondeterministic stacks of bits, based on [16]:

```
bth NDSTACK is protecting BIT
  sort Stack
  op top   : Stack -> Bit
  op pop   : Stack -> Stack
  op empty : -> Stack
  op push  : Stack -> Stack  [nbop]
  var S : Stack
  eq pop(empty)   = empty
  eq pop(push(S)) = S
end
```

3 Limitations of Ordinary Coinduction

We give some specifications where the six rules of Section 2.1 are not enough to prove certain simple properties. Section 5 shows that these properties can be proved with circular Δ-coinduction. The first example specifies infinite bit streams, with an operation that reverses each bit. The data theory for bits is given first; here sorts and operations are considered visible.

```
dth BIT is
  sort Bit
  ops 0 1 : -> Bool
  op not : Bool -> Bool
  eq not(1) = 0
  eq not(0) = 1
end

bth REV is protecting BIT
  sort Stream
  op head : Stream -> Bool
  op tail : Stream -> Stream
  op _&_  : Bool Stream -> Stream
  op rev  : Stream -> Stream
  var B : Bool  var S : Stream
  eq head(B & S) = B
  eq tail(B & S) = S
  eq head(rev(S)) = not(head(S))
  eq tail(rev(S)) = rev(tail(S))
end
```

We can apply a congruence criterion from [16] (see also Corollary 15) to see that `_&_` is congruent for the behavioral specification with only `head` and `tail` considered behavioral. Similarly, a more general congruence criterion from [2] (see also Corollary 14) implies that `rev` is congruent too. Thus Theorem 9 implies that $\Delta = \{$`head,tail`$\}$ is a cobasis for `REV`. Now it is easy to use Δ-coinduction to prove properties like `rev(B& S) = not(B) & rev(S)`, since it is immediate that `head(rev(B & S))` and `head(not(B) & rev(S))` are both equal to `not(B)`, and that `tail(rev(B & S))` and `tail(not(B) & rev(S))` are both equal to `rev(S)`.

On the other hand, Δ-coinduction plus the six rules of Section 2.1 cannot prove the behavioral equality `rev(rev(S)) = S`, because `tail(rev(rev(S))) = tail(S)`, and `tail(tail(rev(rev(S)))) = tail(tail(S))`, and so on through an infinite recurrence. However, circular Δ-coinduction *can* prove this property, and much more, including operation congruence properties that do not follow from the congruence criterion of [2]. Moreover, all operations that are congruent by the criteria of [2, 16] can be proved congruent by $\Delta^{\circlearrowleft}$-coinduction (see Corollaries 14 and 15).

The following behavioral specification of infinite streams of natural numbers is interesting not only because natural properties of it cannot be proved by coinduction, but also because it admits two unrelated cobases:

```
bth STREAM is protecting NAT
  sort Stream
  op head : Stream -> Bool
  op tail : Stream -> Stream
  op _&_  : Nat Stream -> Stream
  op odd  : Stream -> Stream
```

```
  op even : Stream -> Stream
  op zip  : Stream Stream -> Stream

  var N : Nat  var S S' : Stream
  eq head(N & S) = B                      *** 1
  eq tail(N & S) = S                      *** 2
  eq head(odd(S)) = head(S)               *** 3
  eq tail(odd(S)) = even(tail(S))         *** 4
  eq head(even(S)) = head(tail(S))        *** 5
  eq tail(even(S)) = even(tail(tail(S)))  *** 6
  eq head(zip(S,S')) = head(S)            *** 7
  eq tail(zip(S,S')) = zip(S',tail(S))    *** 8
end
```

As usual, `head`, `tail` and `_&_` give the first element, the rest but the first element, and add an element to the front of a stream, respectively, while `odd` and `even` give the streams formed by the elements in the odd and even positions, respectively, and `zip` interleaves two streams. For example, `odd(1 2 3 4 5 6 7 8 9 ...)` is `1 3 5 7 9 ...`, while `even(1 2 3 4 5 6 7 8 9 ...)` is `2 4 6 8 ...`, and `zip(1 3 5 7 9 ..., 2 4 6 8 ...)` is `1 2 3 4 5 6 7 8 9 ...`.

Notice that all operations are behavioral, because they preserve the intended behavioral equivalence, which is "two streams are equivalent iff they have the same elements in the same order." However, given a model, there are at least two interesting ways to generate this behavioral equivalence on that model. One uses observations (or contexts) built with `head` and `tail`, and the other uses observations built with `head`, `odd` and `even`. For example, both

`head(tail(tail(tail(tail(S)))))`
`head(even(odd(odd(S))))`

"observe" the fifth element of `S`, while the term

`head(even(even(odd(even(odd(S))))))`

"observes" the 27th element:

$$\begin{aligned}
\texttt{S} &= a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8\ a_9\ \cdots \\
\texttt{odd(S)} &= a_1\ a_3\ a_5\ a_7\ a_9\ a_{11}\ a_{13}\ a_{15}\ \cdots \\
\texttt{even(odd(S))} &= a_3\ a_7\ a_{11}\ a_{15}\ a_{19}\ a_{23}\ a_{27}\ a_{31}\ \cdots \\
\texttt{odd(even(odd(S)))} &= a_3\ a_{11}\ a_{19}\ a_{27}\ a_{35}\ a_{43}\ a_{51}\ a_{59}\ \cdots \\
\texttt{even(odd(even(odd(S))))} &= a_{11}\ a_{27}\ a_{43}\ a_{59}\ \cdots \\
\texttt{even(even(odd(even(odd(S)))))} &= a_{27}\ a_{59}\ \cdots \\
\texttt{head(even(even(odd(even(odd(S))))))} &= a_{27}
\end{aligned}$$

There are reasons to consider the second cobasis better than the first. For example, a stream's elements can be reached more quickly (e.g., the 27th element can be observed in 6 steps instead of 27), so that less computation is needed for testing. Also properties like `zip(odd(S),even(S)) = S` have much easier proofs with {`head, odd, even`}-coinduction, but seem impossible using {`head,tail`} as observers without circular Δ-coinduction. Section 5 shows that indeed $\Delta =$ {`head,tail`} and $\Delta' =$ {`head,odd,even`} are both valid cobases.

Adding circular Δ-coinduction to Δ-coinduction and the five rules adapted from equational reasoning yields an inference system for behavioral properties which allows us to prove everything we know so far, but of course there is no guarantee that new inference rules will not be needed for more exotic examples. In fact, our experience so far with behavioral deduction leads us to the following, which again parallels the situation for ordinary induction, and means that no finite set of rules can be fully adequate:

Conjecture 1 *Behavioral equational logic is incomplete.*

4 Circular Coinduction

Let $\mathcal{B} = (\Sigma, \Gamma, E)$ be a behavioral specification that is fixed within this section, and let Δ be a complete set of observers, i.e., a cobasis in the sense of Proposition 5 and Theorem 7. To simplify notation, we consider all equations to be quantified by exactly the variables that occur in their two terms, and omit them whenever possible; also write $t \equiv t'$ instead of $\mathcal{B} \models (\forall X)\ t = t'$.

Definition 10. Substitutions $\theta_1, \theta_2 \colon X \to T_\Gamma(Y)$ are **behaviorally equivalent**, written $\theta_1 \equiv \theta_2$, iff $\theta_1(x) \equiv \theta_2(x)$ for every $x \in X$. Terms t_1 and t_2 are **strongly behaviorally equivalent**, written $t_1 \mathrel{\tilde{\equiv}} t_2$, iff for any $\mathcal{B}$-algebra A and any $\tau_1, \tau_2 \colon X \to A$ with $\tau_1(x) \equiv^\Gamma_\Sigma \tau_2(x)$ for each $x \in X$, $\tau_1(t_1) \equiv^\Gamma_\Sigma \tau_2(t_2)$.

Notice that $\mathrel{\tilde{\equiv}}$ is symmetric and transitive but may not be reflexive, since, for example, terms of the form $\sigma(x_1, ..., x_n)$ are not strongly equivalent to any term if σ is not congruent.

Proposition 11. *The following assertions hold:*

1. $t_1 \mathrel{\tilde{\equiv}} t_2$ *implies* $t_1 \equiv t_2$;
2. $t \mathrel{\tilde{\equiv}} u$ *iff* $t \equiv u$, *whenever* u *is a* Γ*-term*[7];
3. $t_1 \mathrel{\tilde{\equiv}} t_2$ *iff* $\gamma[t_1] \mathrel{\tilde{\equiv}} \gamma[t_2]$ *for all appropriate visible* Γ*-contexts* γ;
4. $t_1 \mathrel{\tilde{\equiv}} t_2$ *and* $\theta_1 \equiv \theta_2$ *imply* $\theta_1(t_1) \mathrel{\tilde{\equiv}} \theta_2(t_2)$;
5. σ *is congruent iff* $\sigma(x_1, ..., x_n) \mathrel{\tilde{\equiv}} \sigma(x_1, ..., x_n)$.

For the rest of the section, we assume some well-founded partial order $<$ on Γ-contexts which is preserved by the operations in Γ. For example, one such order is the depth of contexts.

Definition 12. Terms t_1 and t_2 are **$\Delta^\circlearrowleft$-coinductively equivalent**, written $t_1 \equiv^\circlearrowleft_\Delta t_2$, iff for each appropriate $\delta \in \Delta$, either $\delta[t_1] \equiv \delta[t_2] \equiv u$ for some Γ-term u, or $\delta[t_1] \equiv \theta_1(c[t_1])$ and $\delta[t_2] \equiv \theta_2(c[t_2])$ for some $\theta_1 \equiv \theta_2$ and $c < \delta$.

Theorem 13. $t_1 \equiv^\circlearrowleft_\Delta t_2$ *implies* $t_1 \mathrel{\tilde{\equiv}} t_2$.

[7] We write "Γ-terms" for simplicity, but all results hold for terms built with congruent operations.

Proof. We first show by well-founded induction that for every appropriate visible context γ, $\gamma[t_1] \mathrel{\widetilde{\equiv}} \gamma[t_2]$. Let γ be any visible context and assume that $\gamma'[t_1] \mathrel{\widetilde{\equiv}} \gamma'[t_2]$ for all visible contexts $\gamma' < \gamma$. Since Δ is a complete set of observers, there is some context γ'' such that $\gamma = \gamma''[\delta]$ for some $\delta \in \Delta$. If there is some Γ-term u such that $\delta[t_1] \equiv \delta[t_2] \equiv u$ then $\gamma[t_1] \equiv \gamma[t_2] \equiv \gamma''[u]$ and $\gamma''[u]$ is a Γ-term, so by 2 of Proposition 11, $\gamma[t_1] \mathrel{\widetilde{\equiv}} \gamma[t_2]$. On the other hand, if $\delta[t_1] \equiv \theta_1(c[t_1])$ and $\delta[t_2] \equiv \theta_2(c[t_2])$ for some $\theta_1 \equiv \theta_2$ and $c < \delta$, then since the variables appearing in contexts are assumed to be always different from the other variables, one gets that $\gamma[t_1] = \theta_1(\gamma''[c[t_1]])$ and $\gamma[t_2] = \theta_2(\gamma''[c[t_2]])$, and so by the induction hypothesis for $\gamma' = \gamma''[c] < \gamma''[\delta] = \gamma$ and 4 of Proposition 11, $\gamma[t_1] \mathrel{\widetilde{\equiv}} \gamma[t_2]$. The rest follows by 3 of Proposition 11.

Therefore, the following inference rule is sound for behavioral satisfaction:

$$(7)\ \Delta^{\circlearrowleft}\text{-Coinduction}: \quad \frac{t_1 \equiv^{\circlearrowleft}_{\Delta} t_2}{(\forall X)\ t_1 = t_2}$$

The following congruence criterion, which we will call the **BH criterion**, is the essence of that in [2]:

Corollary 14. *Given a complete set Δ of observers and given $\sigma \in \Sigma$ such that for each $\delta \in \Delta$, either $\delta[\sigma(x_1, ..., x_n)] \equiv u$ for some Γ-term u, or else $\delta[\sigma(x_1, ..., x_n)] = c[\sigma(t_1, ..., t_n)]$ for some Γ-terms $t_1, ..., t_n$ and $c < \delta$, then σ is congruent.*

Proof. Theorem 13 with $t_1 = t_2 = \sigma(x_1, ..., x_n)$ and $\theta_1 = \theta_2 = \theta$ with $\theta(x_i) = t_i$ for all $1 \leq i \leq n$, gives $\sigma(x_1, ..., x_n) \mathrel{\widetilde{\equiv}} \sigma(x_1, ..., x_n)$. Then 5 of Proposition 11 gives congruence of σ.

The following simpler but common congruence criterion, which we here call the **RG criterion**, was presented in [16] together with the suggestion that it could be easily implemented in a system like CafeOBJ:

Corollary 15. *Given an operation $\sigma \in \Sigma$ such that for each $\delta \in \Gamma$, if the equation $\delta[\sigma(x_1, ..., x_n)] = u$ for some Γ-term u is in E, then σ is congruent.*

Proof. This is the special case of the BH criterion where $\Delta = \Gamma$ and there is no circularity (i.e., recurrence) in the definition of σ.

5 Further Examples

Section 3 argued that Δ-coinduction plus equational reasoning cannot prove `rev(rev(S)) = S` for the `REV` specification. We now show that $\Delta^{\circlearrowleft}$-coinduction is adequate for this problem. First notice that $\Delta = \{$`head,tail`$\}$ is a cobasis for `REV`. It is easy to show that `head(rev(rev(S)))` $\equiv$ `head(S)` and that `tail(rev(rev(S)))` $\equiv$ `rev(rev(tail(S)))`. Then from Definition 12, where for $\delta =$ `tail` one takes $c = \star$ (the trivial context) and θ_1(`S`) $= \theta_2$(`S`) $=$ `tail(S)`,

we get that `rev(rev(S))` $\equiv^{\circlearrowleft}_{\Delta}$ `S`, and so Theorem 13 and 1 in Proposition 11 give `rev(rev(S)) = S`.

We now consider a number of properties of the `STREAM` specification of Section 3. First, we show that $\Delta = \{$`head,tail`$\}$ really is a cobasis for `STREAM`. Notice that `_&_` is congruent by the RG criterion, that `even` and `zip` are congruent by the BH criterion, but that the congruence of `odd` does not follow by the BH criterion as formulated in [2], because a different non-observational operation is used in the right hand term, namely `even`; however its congruence does follow from Corollary 14 with the remark that there is no distinction between Γ-terms and "terms built with congruent operations" (see footnote 7).

The equalities of visible sort `Nat` in the list below are easy, and are left as exercises; similarly, when proving behavioral equalities `t` $\equiv$ `t'` by $\{$`head, tail`$\}$-coinduction, the interesting subtask is `tail(t)` $\equiv$ `tail(t')`, and the other subtask is skipped. Since all operations are behavioral, equational reasoning is sound.

- `head(tail(S))` $\equiv$ `head(even(S))`.
- `odd(tail(S))` $\equiv$ `even(S)`. We use $\{$`head,tail`$\}$-coinduction. By equation 4, `tail(odd(tail(S)))` $\equiv$ `even(tail(tail(S)))`; by 6, `tail(even(S))` $\equiv$ `even(tail(tail(S)))`; therefore `tail(odd(tail(S)))` $\equiv$ `tail(even(S))`.
- `even(tail(S)` $\equiv$ `tail(odd(S))`. By equation 4.
- `head(N & S)` $\equiv$ `N`.
- `odd(N & S)` $\equiv$ `N & even(S)`. We use $\{$`head,tail`$\}$-coinduction. By equations 4 and 2, `tail(odd(N & S))` $\equiv$ `even(S)`; by 2, `tail(N & even(S))` $\equiv$ `even(S)`; therefore `tail(odd(N & S))` $\equiv$ `tail(N & even(S))`.
- `even(N & S)` $\equiv$ `odd(S)`. We use $\{$`head,tail`$\}$-coinduction. It follows from equations 6, 2, and 4 that `tail(even(N & S))` $\equiv$ `tail(odd(S))`.
- `head(zip(S,S'))` $\equiv$ `head(S)`.
- `even(zip(S,S'))` $\equiv$ `S'`. By equations 6 and 8, `tail(even(zip(S,S')))` $\equiv$ `even(zip(tail(S),tail(S')))`. Now we use $\{$`head,tail`$\}^{\circlearrowleft}$-coinduction. Definition 12 where for $\delta =$ `tail` one takes $c = \star$ (the trivial context) and $\theta_1(\mathtt{S}) = \theta_2(\mathtt{S}) =$ `tail(S)`, gives `even(zip(S,S'))` $\equiv^{\circlearrowleft}_{\Delta}$ `S'`, so by Theorem 13 and 1 of Proposition 11, `even(zip(S,S'))` $\equiv$ `S'`.
- `odd(zip(S,S'))` $\equiv$ `S`. We use $\{$`head,tail`$\}$-coinduction. By equations 4 and 8 and the previous behavioral equality, `tail(odd(zip(S,S')))` $\equiv$ `tail(S')`.

Therefore the congruence criterion (Corollary 14) and Theorem 9 give that $\Delta' = \{$`head,odd,even`$\}$ is another cobasis for `STREAM`, and actually, we can prove that `STREAM` is equivalent to the following specification, since every equation of `STREAM` can be proved in `STREAM'` equationally and/or by $\{$`head,odd,even`$\}$-coinduction.

```
bth STREAM' is protecting NAT
 sort Stream
 op head : Stream -> Bool
 op tail : Stream -> Stream
 op _&_  : Nat Stream -> Stream
 op odd  : Stream -> Stream
```

```
  op even : Stream -> Stream
  op zip  : Stream Stream -> Stream

  var N : Nat  var S S' : Stream
  eq head(tail(S)) = head(even(S))        *** 1
  eq  odd(tail(S)) = even(S)              *** 2
  eq even(tail(S)) = tail(odd(S))         *** 3
  eq head(N & S) = B                      *** 4
  eq  odd(N & S) = N & even(S)            *** 5
  eq even(N & S) = odd(S)                 *** 6
  eq head(zip(S,S')) = head(S)            *** 7
  eq  odd(zip(S,S')) = S                  *** 8
  eq even(zip(S,S')) = S'                 *** 9
end
```

The behavioral equality

$\texttt{zip(odd(S),even(S))} \equiv \texttt{S}$

follows easily by {head,odd,even}-coinduction, but it *cannot* be proved by {head,tail}-coinduction plus equational reasoning, because of the infinite chain of subtasks that is created,

$\texttt{zip(odd(tail(S)),even(tail(S)))} \equiv \texttt{tail(S)}$

$\texttt{zip(odd(tail(tail(S))),even(tail(tail(S))))} \equiv \texttt{tail(tail(S))}$

.....

However, this equality *can* be proved by $\{\texttt{head, tail}\}^{\circlearrowleft}$-coinduction, taking $c = \star$ (the trivial context) and $\theta_1(\texttt{S}) = \theta_2(\texttt{S}) = \texttt{tail(S)}$.

References

1. Gilles Bernot, Michael Bidoit, and Teodor Knapik. Observational specifications and the indistinguishability assumption. *Theoretical Computer Science*, 139(1-2):275–314, 1995. Submitted 1992.
2. Michael Bidoit and Rolf Hennicker. Observer complete definitions are behaviourally coherent. Technical Report LSV-99-4, Ecole Normale Superior de Cachan, 1999.
3. 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.
4. 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.
5. 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.
6. Joseph Goguen. An introduction to algebraic semiotics, with applications to user interface design. In Chrystopher Nehaniv, editor, *Computation for Metaphors, Analogy and Agents*, pages 242–291. Springer, 1999. Lecture Notes in Artificial Intelligence, Volume 1562.

7. Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, January 1992.
8. Joseph Goguen and Grant Malcolm. A hidden agenda. Technical Report CS97–538, UCSD, Dept. Computer Science & Eng., May 1997. To appear in special issue of *Theoretical Computer Science* on Algebraic Engineering, edited by Chrystopher Nehaniv and Masamo Ito. Extended abstract in *Proc., Conf. Intelligent Systems: A Semiotic Perspective, Vol. I*, ed. J. Albus, A. Meystel and R. Quintero, Nat. Inst. Science & Technology (Gaithersberg MD, 20–23 October 1996), pages 159–167.
9. Joseph Goguen and Grigore Roşu. Hiding more of hidden algebra. In Jeannette Wing, Jim Woodcock, and Jim Davies, editors, *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.
10. Joseph Goguen and Grigore Roşu. A protocol for distributed cooperative work. In *Proceedings, Workshop on Distributed Systems, 1999*. Springer, to appear 1999. (Iaşi, Romania) Electronic Lecture Notes in Computer Science, Volume 28.
11. Joseph Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. In Joseph Goguen and Grant Malcolm, editors, *Software Engineering with OBJ: Algebraic Specification in Action*. Kluwer, to appear. Also Technical Report SRI-CSL-88-9, August 1988, SRI International.
12. 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.
13. Shusaku Iida, Michihiro Matsumoto, Răzvan Diaconescu, Kokichi Futatsugi, and Dorel Lucanu. Concurrent object composition in CafeOBJ. Technical Report IS–RR–96–0024S, Japan Advanced Institute for Science and Technology, 1997.
14. Peter Padawitz. Swinging types = functions + relations + transition systems, 1999. `http://issan.informatik.uni-dortmund.de/~peter`. Submitted to *Theoretical Computer Science*.
15. Grigore Roşu. Behavioral coinductive rewriting. In Kokichi Futatsugi, Joseph Goguen, and José Meseguer, editors, *OBJ/CafeOBJ/Maude at Formal Methods '99*, pages 179–196. Theta (Bucharest), 1999. Proceedings of a workshop held in Toulouse, France, 20 and 22 September 1999.
16. Grigore Roşu and Joseph Goguen. Hidden congruent deduction. In Ricardo Caferra and Gernot Salzer, editors, *Proceedings, 1998 Workshop on First Order Theorem Proving*, pages 213–223. Technische Universität Wien, 1998. (Schloss Wilhelminenberg, Vienna, November 23-25, 1998). Full version to appear in *Lecture Notes in Artificial Intelligence*, Springer, 1999.

This article was processed using the LaTeX macro package with LLNCS style