Circular Coinduction –A Proof Theoretical Foundation–

Grigore Roşu[1] and Dorel Lucanu[2]

[1] Department of Computer Science
University of Illinois at Urbana-Champaign, USA, `grosu@illinois.edu`
[2] Faculty of Computer Science
Alexandru Ioan Cuza University, Iaşi, Romania, `dlucanu@info.uaic.ro`

Abstract. Several algorithmic variants of circular coinduction have been proposed and implemented during the last decade, but a proof theoretical foundation of circular coinduction in its full generality is still missing. This paper gives a three-rule proof system that can be used to formally derive circular coinductive proofs. This three-rule system is proved behaviorally sound and is exemplified by proving several properties of infinite streams. Algorithmic variants of circular coinduction now become heuristics to search for proof derivations using the three rules.

1 Introduction

Circular coinduction at a glance. Circular coinduction is a generic name associated to a series of algorithmic techniques to prove behavioral equivalence mechanically. A variant of circular coinduction was published for the first time in [15] in March 2000, and then other more general or more operational variants not too long afterwards first in [13] and then in [5, 6]; the system BOBJ implemented an earlier variant of circular coinduction back in 1998. The name "circular coinduction" was inspired from how it operates: it systematically searches for circular behaviors of the terms to prove equivalent. More specifically, it derives the behavioral equational task until one obtains, on every path, either a truth or a cycle.

$$\boxed{\begin{array}{l} \mathit{zeros} = 0 : \mathit{zeros} \\ \mathit{ones} = 1 : \mathit{ones} \\ \mathit{blink} = 0 : 1 : \mathit{blink} \\ \mathit{zip}(B : S, S') = B : \mathit{zip}(S', S) \end{array}}$$

Fig. 1. Stream definitions

Let us intuitively discuss how circular coinduction operates, by means of an example; rigorous definitions and proofs are given later in the paper. The usual *zeros* and *ones* infinite streams containing only 0 and 1 bits, respectively, as well as a *blink* stream of alternating 0 and 1 bits and the *zip* binary operation on streams, can be defined equationally as in Fig. 1. Lazy evaluation languages like Haskell support such stream definitions. One can prove $\mathit{zip}(\mathit{zeros}, \mathit{ones}) = \mathit{blink}$ by circular coinduction as follows: (1) Check that the two streams have the same head, namely 0; (2) Take the tail of the two streams and generate new goal $\mathit{zip}(\mathit{ones}, \mathit{zeros}) = \mathit{1{:}blink}$; this becomes the next task; (3) Check that the two new streams have the same head, 1 in this case; (4) Take the tail of the two new streams; after simplification one gets the new goal *zip*(*zeros*, *ones*) = *blink*, which is nothing but the original proof task; (5) Conclude that *zip*(*zeros*, *ones*) = *blink* holds. This is also shown in Fig. 2.

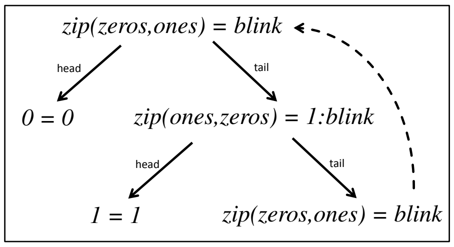

Fig. 2. Intuitive proof by circular coinduction of *zip*(*zeros*, *ones*) = *blink*

The intuition for the above "proof" is that the two streams have been exhaustively tried to be distinguished by iteratively deriving them, that is, checking their heads and taking their tails. Ending up in circles (we obtained the same new proof task as the original one) means that the two streams are indistinguishable with experiments involving stream heads and tails, so they are equal. In fact, the correctness of circular coinduction can be informally argued in several ways; we found that colleagues with different backgrounds prefer different arguments: (1) Since each derived path ends up in a cycle, it means that there is no way to show the two original terms behaviorally different, so they must be equal; (2) The obtained circular graph structure can be used as a backbone to "consume" any possible experiment applied on the two original terms, rolling each experiment until it becomes a visible equality, so the graph is a witness that the two terms are "indistinguishable under experiments"; (3) The equalities that appear as nodes in the obtained graph can be regarded as lemmas inferred in order to prove the original task, and the graph itself as a dependence relation among these lemmas; one can take all these and produce a parallel proof by context induction [9] for all of them; (4) When/if it stabilizes, it "discovers" a binary relation which is compatible with the derivatives and is the identity on data, so the stabilized set of equations, which includes the original task, is included in the behavioral equivalence (because the latter is the largest with that property); (5) It incrementally completes a given equality into a bisimulation relation on terms, so the two original terms must be behaviorally equivalent (bisimilar).

At our knowledge, variants of circular coinduction have been implemented in three systems so far: in a behavioral extension of OBJ called BOBJ [13] (not maintained anymore), in Isabelle/HOL for CoCasl [8], and in CIRC [11].

Novel proof system. The definition and proof of correctness of the original circular coinduction variant in [15, 13] were rather complex. Moreover, the setting in those and other papers on circular coinduction was model theoretical, with hidden algebras as models, making one wonder whether the applicability of circular coinduction was limited to hidden algebra models or not. Here we adopt a general proof theoretical approach, without fixing any particular models (and implicitly any particular model-based notions of behavioral equivalence) and any particular base deductive system. Instead, our approach here is parametric in a given base *entailment relation* $\vdash$, which one can use to discard "obvious" proof tasks, and in a set of *derivatives* Δ, which are special terms with a hole (i.e., contexts) allowing us to define a syntactic notion of *behavioral equivalence* on terms without referring to any model but only to $\vdash$. This syntactic notion of behavioral equivalence is so natural that it is straightforwardly sound in all models that we are aware of. More importantly, it allows for more succinct and clean definitions and proofs for circular coinduction, independent of the underlying models.

Fig. 3 shows our novel formulation of circular coinduction as a three-rule proof system deriving pairs of the form $\mathcal{H} \Vvdash^{\circlearrowleft} \mathcal{G}$, where $\mathcal{H}$ (*hypotheses*) and $\mathcal{G}$ (*goals*) are sets of equations. Some equations can be *frozen* and written in a box (e.g., $\boxed{e}$), with the intuition that those equations cannot be used in contextual reasoning (i.e., the congruence rule of equational logic cannot be applied on them), but only at the top; this is easily achieved by defining the box as a wrapper operator of a fresh sort result, say *Frozen*. $\mathcal{G}$ contains only frozen equations, while $\mathcal{H}$ may contain both frozen (denoted by $\mathcal{F}$) and normal (denoted by $\mathcal{B}$), or "unfrozen" equations; the unfrozen equations are those in the original behavioral specification $\mathcal{B}$. The first rule says that we are done when there are no goals left. The second rule discards those goals which can be proved with the base entailment system. The third rule is the distinguished rule of circular coinduction that enables circular reasoning and it essentially says the following, where the involved equations are all frozen:

$$\frac{\cdot}{\mathcal{B} \cup \mathcal{F} \Vvdash^{\circlearrowleft} \emptyset}$$

$$\frac{\mathcal{B} \cup \mathcal{F} \Vvdash^{\circlearrowleft} \mathcal{G}, \quad \mathcal{B} \cup \mathcal{F} \vdash \boxed{e}}{\mathcal{B} \cup \mathcal{F} \Vvdash^{\circlearrowleft} \mathcal{G} \cup \{\boxed{e}\}}$$

$$\frac{\mathcal{B} \cup \mathcal{F} \cup \{\boxed{e}\} \Vvdash^{\circlearrowleft} \mathcal{G} \cup \boxed{\Delta[e]}}{\mathcal{B} \cup \mathcal{F} \Vvdash^{\circlearrowleft} \mathcal{G} \cup \{\boxed{e}\}, \ e \text{ hidden}}$$

Fig. 3. Circular coinduction

To prove behavioral property e, assume it and prove its derivatives $\Delta[e]$; or, metaphorically, "if one cannot show it wrong then it is right".

Let us discuss the circular coinductive proof system for infinite streams, whose derivatives Δ are the head and tail operations. The third rule in Fig. 3 says that in order to prove an equality of two infinite streams s_1 and s_2 (let us informally write it $s_1 \sim s_2$ for the time being), just assume $s_1 \sim s_2$ as a hypothesis and then prove $head(s_1) = head(s_2)$ and $tail(s_1) \sim tail(s_2)$. To prove the latter, one may need to apply the third rule again, assuming $tail(s_1) \sim tail(s_2)$ and then proving $head(tail(s_1)) = head(tail(s_2))$ and $tail(tail(s_1)) \sim tail(tail(s_2))$, and so on and so forth. The proof system therefore attempts to incrementally perform a bisimulation, or behavioral closure of the property to prove. Practice and experience tell us that naive attempts to perform such a closure, without an appropriate reasoning infrastructure, will easily lead to nontermination in many cases. The strength of circular coinduction comes from the fact that it uses the available knowledge, i.e. both the equations in $\mathcal{B}$ and the accumulated hypotheses in $\mathcal{F}$, during this closure process; however, using the available knowledge soundly and effectively is a much more complex task than it may appear.

There are several reasoning aspects encapsulated in the second condition of the second rule in Fig. 3, namely in $\mathcal{B} \cup \mathcal{F} \vdash \boxed{e}$, that are key to termination and therefore core to circular coinduction. For example, can one freely use the original equations in $\mathcal{B}$, including those between infinite streams, to simplify the terms s_1 and s_2 in a new proof task $s_1 \sim s_2$? The answer is *yes*. Also, can one freely use the accumulated hypotheses in $\mathcal{F}$? This time the answer is *no*, because otherwise the new tasks $\Delta[e]$ added by the third rule in Fig. 3 would hold vacuously by the congruence rule of equational reasoning. Freezing inhibits the applications of congruence, so one needs to find other means to eliminate the derivative operations. It turns out that the other rules of equational deduction are sound in combination with the accumulated hypotheses in $\mathcal{F}$, including substitution and transitivity, but why that is the case is not immediate.

A way to understand freezing by means of context induction [9] intuitions is to regard it as the application of an arbitrary but fixed context C to an equation e; with this, the third rule in Fig. 3 says that in order to prove e, assume $C[e]$ for an arbitrary but fixed context C and prove $C[\delta[e]]$ for any $\delta \in \Delta$. This view does not explain the need for freezing e in $\mathcal{G} \cup \{\boxed{e}\}$ in the third rule in Fig. 3, but it gives an intuitive argument why congruence is not sound in combination with the accumulated hypotheses. This way of applying context induction is reminiscent to the inductive proof techniques in [2], but there C is a variable of sort *context* and one needs to do an explicit induction on it. The use of freezing relieves the user of our proof system from performing explicit induction on contexts; in fact, the user of our proof system needs not be aware of any contexts at all (except for the derivatives), nor of induction on contexts. Nevertheless, the soundness of our proof system is complex and makes use of a well-founded induction on contexts (in Theorem 2), but the proof system itself is simple and easy to use.

The circular coinductive proof system in Fig. 3 is the result of many years of experimentation with BOBJ [13], a behavioral variant of OBJ implementing an early version of circular coinduction, and with CIRC [11], a behavioral extension of Maude [3] optimized for automated inductive and coinductive proving. The operational simplicity of our proof system may, however, be deceiving. For example, a crucial detail may have passed unnoticed to the hasty reader: the third rule in Fig. 3 does *not* split the proof of $\mathcal{G} \cup \{\boxed{e}\}$ into a proof of $\mathcal{G}$ and a proof of $\Delta[\boxed{e}]$. Instead, it generates a proof task for $\mathcal{G} \cup \Delta[\boxed{e}]$. This allows one to move some equations in the set of goals $\mathcal{G}$ as hypotheses in $\mathcal{F}$ and use them to prove other, completely unrelated equations in $\mathcal{G}$! This allows us, in particular, to prove a set of mutually related behavioral equalities in parallel (an example is given later in the paper). The soundness of our proof system (Theorem 3, which makes use of Theorem 2) is a monolithic and difficult result, depending upon the existence of a *complete* derivation proof for $\mathcal{B} \Vdash^{\circlearrowleft} \boxed{G}$. In other words, its soundness does not follow from the soundness of each derivation rule, as it is commonly done in soundness proofs; in fact, the third rule is not sound in isolation. We are not aware of any other similar proof systems in the literature.

A question is whether the proposed proof system can prove all behavioral equalities. Unfortunately, the answer is negative even for the particular case of streams. Indeed, [14] shows that stream behavioral equivalence is a Π_2^0 complete problem, which means in particular that there is no complete proof system for behavioral equivalence. Thus, the best we can do is to algorithmically approximate behavioral equivalence, aiming at covering more practical situations, but there will always be room for better proof systems. In fact, the latest version of CIRC [10] already implements an extended and more complex variant of the discussed proof system. Nevertheless, the circular coinductive proof system presented in this paper is simple, powerful and relatively efficiently implementable.

Implementation in CIRC. The proposed circular coinductive reasoning system has been implemented and extensively evaluated in the latest version of CIRC. Even though versions of CIRC have been available for download and online experimentation at `http : //fsl.cs.uiuc.edu/circ` for more than 3 years, this is the first paper about its foundations from a proof theoretical rather than algorithmic perspective, explaining how its automated coinductive engine works and why it is sound. Since CIRC can automatically prove using the discussed technique all the behavioral properties in this paper (see its website above) and since its syntax minimally extends that of Maude, which we assume known, from here on we give our running specification, that of streams, using CIRC:

```
op hd : Stream -> Bit .          derivative hd(*:Stream) .
op tl : Stream -> Stream .       derivative tl(*:Stream) .
```

Declaring `hd` (head) and `tl` (tail) as *derivatives* tells CIRC that one can use them to derive (or observe) the task to prove. One can define several operations on streams. For example, `not(S)` reverses each bit in stream `S` and is trivial to define; `odd`, `even` and `zip` are standard and can be defined as follows (we use now the definitional style of CIRC, explicitly using derivatives in equations):

```
ops odd even : Stream -> Stream .  op zip : Stream Stream -> Stream .
eq hd(odd(S)) = hd(S) .            eq hd(zip(S,S')) = hd(S) .
eq tl(odd(S)) = even(tl(S)) .      eq tl(zip(S,S')) = zip(S',tl(S)) .
eq even(S) = odd(tl(S)) .
```

One can now ask CIRC to prove $\mathtt{zip(odd(S),even(S))} = \mathtt{S}$. Using the circular coinductive proof system, CIRC proves it automatically as follows: (1) assume $\mathtt{zip(odd(S),even(S))} = \mathtt{S}$ as frozen hypothesis; (2) add its derivatives $\mathtt{hd(zip(odd(S),even(S)))} = \mathtt{hd(S)}$ and $\mathtt{tl(zip(odd(S),even(S)))} = \mathtt{tl(S)}$ as new proof tasks; (3) prove first task using the stream equations; (4) prove second task by first reducing it to $\mathtt{zip(odd(tl(S)),even(tl(S)))} = \mathtt{tl(S)}$ using stream equations and then using frozen hypothesis (1) at its top with substitution $\mathtt{S} \mapsto \mathtt{tl(S)}$.

Sometimes it is easier to prove more tasks at the same time than each task separately, because they can help each other. Here is an example. The famous Thue-Morse sequence (see, e.g., [1]) has several interesting properties and can be defined as a stream in several ways (we give one definition below). Interestingly, `morse` is a fixed point of the operation `f` also defined below:

```
--- Thue-Morse sequence:                   --- morse is a fixed point of f
op morse : -> Stream .                     op f : Stream -> Stream .
eq hd(morse) = 0 . eq hd(tl(morse)) = 1 .  eq hd(f(S)) = hd(S) .
eq tl(tl(morse))                           eq hd(tl(f(S))) = not(hd(S)) .
   = zip(tl(morse), not(tl(morse))) .      eq tl(tl(f(S))) = f(tl(S)) .
```

If we try to prove $\mathtt{f(morse)} = \mathtt{morse}$ (either manually or using CIRC), then we see that larger and larger frozen equations are derived and the process does not terminate. Analyzing the new equations (as humans), we see that the source of problems is the lack of an equation $\mathtt{f(S)} = \mathtt{zip(S,not(S))}$. We can then either prove it first as a lemma, or simply add it to the set of goals and then restart the coinductive proving process with the goal $\{\mathtt{f(morse)} = \mathtt{morse},\ \mathtt{f(S)} = \mathtt{zip(S,not(S))}\}$, this time succeeding in ten steps (see Example 4 in Section 5).

Paper structure. Section 2 introduces basic notions and notations. A key notion is that of a Δ-contextual entailment system; one such system is assumed given and acts as a parameter of our generic techniques developed in the rest of the paper. Section 3 recasts behavioral satisfaction in our generic setting. Section 4 is dedicated to defining behavioral equivalence and coinduction in our generic setting, showing that behavioral equivalence remains the largest relation closed under derivatives. Our novel circular coinductive proof system is presented in Section 5, together with its behavioral soundness result.

2 Preliminaries

We assume the reader familiar with basics of many sorted algebraic specifications and only briefly recall our notation. An algebraic specification, or simply a *specification*, is a triple (S, Σ, E), where S is a set of *sorts*, Σ is a $(S^* \times S)$-*signature* and E is a set of Σ-*equations* of the form $(\forall X)\, t = t'$ `if` $\wedge_{i \in I}\, u_i = v_i$ with t, t', u_i, and v_i Σ-terms with variables in X, $i = 0, \ldots, n$; the two terms appearing in any equality in an equation, that is the terms t, t' and each pair u_i, v_i for each $i \in I$, have, respectively, the same sort. If the sort of t and t' is s we may say that the sort of the equation is also s. When $i = 0$ we call the equation unconditional and omit the condition (i.e., write it $(\forall X)\, t = t'$). When $X = \emptyset$ we drop the quantifier and call the equation *ground*.

If Σ is a many sorted signature, then a Σ-*context* C is a Σ-term which has one occurrence of a distinguished variable $*{:}s$ of sort s; to make this precise, we may write $C[*{:}s]$ instead of just C. When Σ is understood, a Σ-context may be called just a *context*. When the sort s of $*$ is important, we may call $C[*{:}s]$ a *context for sort* s; also, when the sort of a context C (regarded as a term), say s', is important, C may be called a *context of sort* s'. If $C[*{:}s]$ is a context for sort s of sort s' and t is a term of sort s, then $C[t]$ is the term of sort s' obtained by replacing t for $*{:}s$ in C. A Σ-context $C[*{:}s]$ induces a partially defined *equation transformer* $e \mapsto C[e]$: if e is an equation $(\forall X)\, t = t'$ `if` c of sort s, then $C[e]$ is the equation $(\forall X \cup Y)\, C[t] = C[t']$ `if` c, where Y is the set of non-star variables occurring in $C[*{:}s]$. Moreover, if $\mathbf{C}$ is a set of contexts and E a set of equations, then $\mathbf{C}[e] = \{C[e] \mid C \in \mathbf{C}\}$, $C[E] = \{C[e] \mid e \in E\}$ and $\mathbf{C}[E] = \bigcup_{e \in E} \mathbf{C}[e]$.

The theoretical results in this paper will be parametric in a given entailment relation $\vdash$ on many sorted equational specifications, which may, but is not enforced to, be the usual equational deduction relation [7]. For instance, it can also be the "rewriting" entailment relation ($E \vdash t = t'$ iff t and t' rewrite to the same term using E as a rewrite system), or some behavioral entailment system, etc. We need though some properties of $\vdash$, which we axiomatize here by adapting to our context the general definition of entailments system as given in [12]. Fix a signature Σ (in a broader setting, $\vdash$ could be regarded as a set $\{\vdash_\Sigma \mid \Sigma \text{ signature}\}$ indexed by the signature; however, since we work with only one signature in this paper and that signature is understood, for simplicity we fix the signature).

Definition 1. *If Δ is a set of Σ-contexts, then a* ***Δ-contextual entailment system*** *is an (infix) relation $\vdash$ between sets of equations and equations, with:* ***(reflexivity)*** *$\{e\} \vdash e$;* ***(monotonicity)*** *If $E_1 \supseteq E_2$ and $E_2 \vdash e$ then $E_1 \vdash e$;*

(transitivity) *If $E_1 \vdash E_2$ and $E_2 \vdash e$ then $E_1 \vdash e$;* ***(Δ-congruence)*** *If $E \vdash e$ then $E \vdash \Delta[e]$. Above, E, E_1, E_2 range over sets of equations and e over equations; we tacitly extended $\vdash$ to sets of equations as expected: $E_1 \vdash E_2$ iff $E_1 \vdash e$ for any $e \in E_2$. We let $E^\bullet$ denote the set of equations $\{e \mid E \vdash e\}$.*

One can use the above to prove many properties of $\vdash$ on sets of equations. Here are some of them used later in the paper (their proofs are simple exercises): $E \vdash \emptyset$, $E \vdash E$, if $E_1 \vdash E_2$ and $E_2 \vdash E_3$ then $E_1 \vdash E_3$, if $E_1 \vdash E_2$ then $E \cup E_1 \vdash E \cup E_2$, if $E_1 \vdash E_2$ then $E_1 \vdash \Delta[E_2]$, if $E_1 \vdash E_2$ then $E_1 \vdash E_1 \cup E_2$, if $E_1 \supseteq E_2$ then $E_1 \vdash E_2$, if $E \vdash E_1$ and $E \vdash E_2$ then $E \vdash E_1 \cup E_2$.

We take the liberty to slightly abuse the syntax of entailment and allow one to write a specification instead of a set of equations, with the obvious meaning: if $\mathcal{B} = (S, \Sigma, E)$ is a specification and e is a Σ-equation, then $\mathcal{B} \vdash e$ iff $E \vdash e$. Also, if $\mathcal{B} = (S, \Sigma, E)$ then we may write $\mathcal{B}^\bullet$ instead of $E^\bullet$.

3 Behavioral Specification and Satisfaction

Our approach in this paper is purely proof theoretical, that is, there are no models involved. We prefer this approach because we think that it is the most general one for our purpose. As pointed out in [13], there is a plethora of model theoretical variants of behavioral logics, some with models with fixed data others with loose data, some which are co-algebraic (i.e., isomorphic to a category of coalgebras) others which are not (e.g., some do not even admit final models), some which come with a behavioral equality others whose behavioral equality can be computed, etc. However, the proof theoretical behavioral entailment relation defined in this section is sound for all these different variants of behavioral logics. On the other hand, as shown in [13, 14], neither of the behavioral logics is complete. Therefore, one looses nothing by working with a proof theoretical instead of a model theoretical notion of behavioral entailment. On the contrary, this allows us to define behavioral equivalence and coinduction in a more general way, independent upon the particular (adhoc) choice for models.

A *behavioral specification* is a pair $(\mathcal{B}, \Delta)$, where $\mathcal{B} = (S, \Sigma, E)$ is a many sorted algebraic specification and Δ is a set of Σ-contexts, called *derivatives.* We let Δ_s denote all the derivatives of sort s in Δ. If $\delta[*{:}h] \in \Delta$ is a derivative, then the sort h is called a *hidden sort*; we let $H \subseteq S$ denote the set of all hidden sorts of $\mathcal{B}$. Remaining sorts are called *data, or visible, sorts* and we let $V = S - H$ denote their set. A *data operator* is an operator in Σ taking and returning only visible sorts; a *data term* is a term built with only data operators and variables of data sorts; a *data equation* is an equation built with only data terms.

Sorts are therefore split into hidden and visible, so that one can derive terms of hidden sort until they possibly become visible. Formally, a *Δ-experiment* is a Δ-context of visible sort, that is: (1) each $\delta[*{:}h] \in \Delta_v$ with $v \in V$ is an experiment, and (2) if $C[*{:}h']$ is an experiment and $\delta[*{:}h] \in \Delta_{h'}$, then so is $C[\delta[*{:}h]]$. An equation $(\forall X)\, t = t'$ `if` c is called a *hidden equation* iff the common sort of t and t' is hidden, and it is called a *data, or visible, equation* iff the common sort of t and t' is visible. In this paper we consider only equations whose conditions are conjunctions of visible equalities. If Δ is understood, then we may write experiment for Δ-experiment and context for Δ-context; also, we may write only $\mathcal{B}$ instead of $(\mathcal{B}, \Delta)$ and, via previous conventions, identify $\mathcal{B}$ with its set of equations. If $\mathcal{G}$ is a set of Σ-equations, we let *visible*$(\mathcal{G})$ and *hidden*$(\mathcal{G})$ denote the sets of $\mathcal{G}$'s visible and hidden equations, respectively.

From here on in the paper we fix a signature Σ, a set of Σ-contexts Δ, and a generic Δ-contextual entailment system, $\vdash$. For the sake of concreteness, one may think of $\vdash$ as the ordinary equational entailment relation; however, the proofs in the sequel rely only on the axioms in Definition 1. Unless otherwise specified, we also assume a fixed behavioral specification $(\mathcal{B}, \Delta)$.

Example 1. Let `STREAM` be the behavioral specification of streams discussed in Section 1, with the conventional equational reasoning as basic entailment relation; in other words, $\mathtt{STREAM} \vdash \mathtt{e}$ iff $\mathtt{e}$ is derivable using equational reasoning from the equations in `STREAM`. The set Δ of derivatives is $\{\mathtt{hd}(*{:}\mathtt{Stream}), \mathtt{tl}(*{:}\mathtt{Stream})\}$ and the experiments one can perform on streams are therefore contexts of the form $\mathtt{hd}(\mathtt{tl}^{\mathtt{i}}(*{:}\mathtt{Stream}))$, where $\mathtt{i} \geq \mathtt{0}$.

Definition 2. *$\mathcal{B}$ **behaviorally satisfies** equation e, written $\mathcal{B} \Vvdash e$, iff: $\mathcal{B} \vdash e$ if e is visible, and $\mathcal{B} \vdash C[e]$ for each appropriate experiment C if e is hidden.*

Example 2. In the case of streams, we have that $\mathtt{STREAM} \Vvdash \mathtt{str} = \mathtt{str}'$ iff $\mathtt{STREAM} \vdash \mathtt{hd}(\mathtt{tl}^{\mathtt{i}}(\mathtt{str})) = \mathtt{hd}(\mathtt{tl}^{\mathtt{i}}(\mathtt{str}'))$ for all $\mathtt{i} \geq \mathtt{0}$. For example, one can directly show that $\mathtt{STREAM} \Vvdash \mathtt{zip}(\mathtt{odd}(\mathtt{S}), \mathtt{even}(\mathtt{S})) = \mathtt{S}$ by showing by induction on $\mathtt{i}$ that $\mathtt{hd}(\mathtt{tl}^{\mathtt{i}}(\mathtt{zip}(\mathtt{odd}(\mathtt{S}), \mathtt{even}(\mathtt{S})))) = \mathtt{hd}(\mathtt{tl}^{\mathtt{i}}(\mathtt{S}))$. Also, one can show that $\mathtt{STREAM} \Vvdash \mathtt{f}(\mathtt{not}(\mathtt{S})) = \mathtt{not}(\mathtt{f}(\mathtt{S}))$, by showing that $\mathtt{STREAM} \vdash \mathtt{hd}(\mathtt{tl}^{\mathtt{i}}(\mathtt{f}(\mathtt{not}(\mathtt{S}))))$ $= \mathtt{g}(\mathtt{i}, \mathtt{hd}(\mathtt{S}))$ and $\mathtt{STREAM} \vdash \mathtt{hd}(\mathtt{tl}^{\mathtt{i}}(\mathtt{not}(\mathtt{f}(\mathtt{S})))) = \mathtt{g}(\mathtt{i}, \mathtt{hd}(\mathtt{S}))$ for all $\mathtt{i} \geq \mathtt{0}$, where $\mathtt{g}(\mathtt{i}, \mathtt{B}) = \mathtt{not}(\mathtt{B})$ if $\mathtt{i}$ is even, and $\mathtt{g}(\mathtt{i}, \mathtt{B}) = \mathtt{B}$ if $\mathtt{i}$ is odd. Such direct proofs based on direct exhaustive analysis of experiments are essentially proofs by context induction [9], known to be problematic in practice due to the large number of lemmas needed (see [4] for an early reference to this aspect) and difficult to mechanize. We are going to give more elegant behavioral proofs of these properties, first by coinduction and then by circular coinduction.

Proposition 1. *$\Vvdash$ extends $\vdash$, that is, if $\mathcal{B} \vdash e$ then $\mathcal{B} \Vvdash e$.*

Proof. If e is visible then it holds by Definition 2. If e is hidden, then by the Δ-congruence property of $\vdash$ in Definition 1, it follows that $\mathcal{B} \vdash C[e]$ for any appropriate Δ-context C, in particular for any Δ-experiment. Hence, $\mathcal{B} \Vvdash e$. $\square$

4 Behavioral Equivalence and Coinduction

Definition 3. *The **behavioral equivalence** of B is the set $\equiv_{\mathcal{B}} \overset{def}{=} \{e \mid \mathcal{B} \Vvdash e\}$.*

Thus, the behavioral equivalence of $\mathcal{B}$ is the set of equations behaviorally satisfied by $\mathcal{B}$. When $\mathcal{B}$ is clear, we may write $\equiv$ instead of $\equiv_{\mathcal{B}}$. Note that $\equiv$ may contain also conditional equations with visible conditions. However, the visible conditions can be regarded as data side conditions, so we took the liberty to make the slight abuse of terminology above and called $\equiv_{\mathcal{B}}$ an equivalence.

A major result of any behavioral logic, making proofs by coinduction sound, is that behavioral equivalence is the largest relation that is consistent with the data and is compatible with the behavioral operations, or the observers. In what follows we capture this same result in our proof theoretical setting. To do that, we first need to formally define what is meant here by "consistent with data and compatible with behavioral operators". Recall that $\mathcal{B}^\bullet$ uses the original entailment relation, that is, $\mathcal{B}^\bullet = \{e \mid \mathcal{B} \vdash e\}$.

Definition 4. *A set of equations $\mathcal{G}$ is **behaviorally closed** iff $\mathcal{B} \vdash visible(\mathcal{G})$ and $\Delta(\mathcal{G} - \mathcal{B}^\bullet) \subseteq \mathcal{G}$.*

In other words, a set of equations $\mathcal{G}$ is behaviorally closed iff for any $e \in \mathcal{G}$, we have that $\mathcal{B} \vdash e$ or otherwise e is hidden and $\Delta[e] \subseteq \mathcal{G}$. Therefore, $\mathcal{G}$ can contain as many equations entailed by $\mathcal{B}$ as needed, both visible and hidden, even all of them. The first condition says that if $\mathcal{G}$ contains any visible equation at all, then that equation must be entailed by $\mathcal{B}$. The second condition says that if a hidden equation $e \in \mathcal{G}$ is not entailed by $\mathcal{B}$ then it must be the case that its derivatives are also in $\mathcal{G}$. A degenerate case is when $\mathcal{G} = \mathcal{B}$, in which case the second condition is superfluous. In other words, $\mathcal{G}$ is closed under, or compatible with, the derivatives and the only way to "escape" this derivative closing process is by a direct proof using $\mathcal{B}$ and the original entailment system.

Theorem 1. ***(Coinduction)*** *For any behavioral specification, the behavioral equivalence $\equiv$ is the largest behaviorally closed set of equations.*

One can prove Theorem 1 directly by induction on contexts; however, we are going to prove a more general result later in the paper, the coinductive circularity principle (Theorem 2), so we refrain from giving a direct proof here.

Theorem 1 justifies the correctness of behavioral proofs by coinduction. Indeed, in order to prove that $\mathcal{B}$ behaviorally satisfies an equation e, all we need to do is to find some behaviorally closed set of equations $\mathcal{G}$ such that $e \in \mathcal{G}$.

Experience with many coinductive proofs suggests that the following simple coinductive proving technique works in most practical cases of interest:

(Coinductive proving technique) *Aim is to prove $\mathcal{B} \Vdash e$.*

1. *If one can show $\mathcal{B} \vdash e$ then stop with **success**, otherwise continue;*
2. *If e is visible then stop with **failure**;*
3. *Find some (typically finite) set of hidden equations G with $e \in G$;*
4. *Let $\mathcal{G}$ be the closure of $G \cup \mathcal{B}^\bullet$ under substitution, symmetry, and transitivity;*
5. *Show $\Delta[G] \subseteq \mathcal{G}$;*

The most difficult part is to find the right G at step *3*. If G is too small, in particular if one picks $G = \{e\}$, then $\mathcal{G}$ may be too small and thus one may not be able to show step *5* because it may be the case for some $f \in G$ and appropriate $\delta \in \Delta$ that $\delta[f] \notin \mathcal{G}$ and one cannot show $\mathcal{B} \vdash \delta[f]$. If G is too large, then the number of proof tasks at step *5* may be prohibitive. Finding a good G at step *3* is the most human-intensive part (the rest is, to a large extent, mechanizable). Steps *4* and *5* above are combined in practice, by showing that one can prove $\Delta[G]$ using unrestricted equational reasoning with the equations in $\mathcal{B}$ and equational reasoning without the congruence rule with the equations in G. The correctness of the technique above follows from the following proposition:

Proposition 2. *Let G be a set of hidden equations such that $\Delta[G] \subseteq \mathcal{G}$, where $\mathcal{G}$ is the closure of $G \cup \mathcal{B}^\bullet$ under substitution, symmetry, and transitivity. Then $\Delta[\mathcal{G}] \subseteq \mathcal{G}$ and $\mathcal{G}$ is behaviorally closed.*

Proof. If $\mathcal{E}$ is a set of equations, let $\overline{\mathcal{E}}$ be the closure of $\mathcal{E}$ under substitution, symmetry, and transitivity, and note that $\Delta[\overline{\mathcal{E}}] = \overline{\Delta[\mathcal{E}]}$. Taking $\mathcal{E} = G \cup \mathcal{B}^\bullet$, we get $\Delta[\mathcal{G}] = \Delta[\overline{G \cup \mathcal{B}^\bullet}] = \overline{\Delta[G \cup \mathcal{B}^\bullet]} = \overline{\Delta[G] \cup \Delta[\mathcal{B}^\bullet]} \subseteq \overline{\mathcal{G} \cup \mathcal{B}^\bullet} = \overline{\mathcal{G}} = \mathcal{G}$. Since G contains only hidden equations and since substitution, symmetry and transitivity are sort preserving, we conclude that $\mathit{visible}(\mathcal{G}) = \mathit{visible}(\mathcal{B}^\bullet)$. It is now clear that $\mathcal{G}$ is behaviorally closed (both conditions in Definition 4 hold). $\square$

Proposition 2 therefore implies that the $\mathcal{G}$ in the coinductive proving technique above is behaviorally closed. Since $e \in G \subseteq \mathcal{G}$, Theorem 1 implies that $\mathcal{B} \Vvdash e$, so our coinductive proving technique is indeed sound.

Example 3. Let us prove by coinduction the two stream properties in Example 2, $\mathtt{STREAM} \Vvdash \mathtt{zip}(\mathtt{odd}(\mathtt{S}), \mathtt{even}(\mathtt{S})) = \mathtt{S}$ and $\mathtt{STREAM} \Vvdash \mathtt{f}(\mathtt{not}(\mathtt{S})) = \mathtt{not}(\mathtt{f}(\mathtt{S}))$.

For the first property, let G be the set $\{\mathtt{zip}(\mathtt{odd}(\mathtt{S}), \mathtt{even}(\mathtt{S})) = \mathtt{S}\}$ containing only the equation to prove, and let $\mathcal{G}$ be the closure of $G \cup \mathtt{STREAM}^\bullet$ under substitution, symmetry and transitivity. Since

$$\begin{aligned}
&\mathtt{hd}(\mathtt{zip}(\mathtt{odd}(\mathtt{S}), \mathtt{even}(\mathtt{S}))) = \mathtt{hd}(\mathtt{S}) \\
&\mathtt{tl}(\mathtt{zip}(\mathtt{odd}(\mathtt{S}), \mathtt{even}(\mathtt{S}))) = \mathtt{zip}(\mathtt{odd}(\mathtt{tl}(\mathtt{S})), \mathtt{even}(\mathtt{tl}(\mathtt{S})))
\end{aligned}$$

are in $\mathtt{STREAM}^\bullet$ (and hence in $\mathcal{G}$), the following equations are in $\mathcal{G}$ as well:

$$\begin{aligned}
&\mathtt{zip}(\mathtt{odd}(\mathtt{tl}(\mathtt{S})), \mathtt{even}(\mathtt{tl}(\mathtt{S}))) = \mathtt{tl}(\mathtt{S}) && \text{(substitution)} \\
&\mathtt{tl}(\mathtt{zip}(\mathtt{odd}(\mathtt{S}), \mathtt{even}(\mathtt{S}))) = \mathtt{tl}(\mathtt{S}) && \text{(transitivity)}
\end{aligned}$$

Therefore $\Delta(G) \subseteq \mathcal{G}$, so we are done.

For the second property, $\mathtt{f}(\mathtt{not}(\mathtt{S})) = \mathtt{not}(\mathtt{f}(\mathtt{S}))$, let G be the two equation set $\{\mathtt{f}(\mathtt{not}(\mathtt{S})) = \mathtt{not}(\mathtt{f}(\mathtt{S})),\ \mathtt{tl}(\mathtt{f}(\mathtt{not}(\mathtt{S}))) = \mathtt{tl}(\mathtt{not}(\mathtt{f}(\mathtt{S})))\}$, which includes the equation to prove, and let $\mathcal{G}$ the closure under substitution, symmetry and transitivity of $G \cup \mathtt{STREAM}^\bullet$. Since the following equations

$$\begin{array}{ll}
\mathtt{hd}(\mathtt{f}(\mathtt{not}(\mathtt{S}))) = \mathtt{not}(\mathtt{hd}(\mathtt{S})) & \mathtt{hd}(\mathtt{not}(\mathtt{f}(\mathtt{S}))) = \mathtt{not}(\mathtt{hd}(\mathtt{S})) \\
 & \mathtt{tl}(\mathtt{not}(\mathtt{f}(\mathtt{S}))) = \mathtt{not}(\mathtt{tl}(\mathtt{f}(\mathtt{S}))) \\
\mathtt{hd}(\mathtt{tl}(\mathtt{f}(\mathtt{not}(\mathtt{S})))) = \mathtt{hd}(\mathtt{S}) & \mathtt{hd}(\mathtt{tl}(\mathtt{not}(\mathtt{f}(\mathtt{S})))) = \mathtt{hd}(\mathtt{S}) \\
\mathtt{tl}(\mathtt{tl}(\mathtt{f}(\mathtt{not}(\mathtt{S})))) = \mathtt{f}(\mathtt{not}(\mathtt{tl}(\mathtt{S}))) & \mathtt{tl}(\mathtt{tl}((\mathtt{not}(\mathtt{f}(\mathtt{S})))) = \mathtt{not}(\mathtt{f}(\mathtt{tl}(\mathtt{S})))
\end{array}$$

are all in $\mathtt{STREAM}^\bullet$ (and hence in $\mathcal{G}$), the following equations are in $\mathcal{G}$ as well:

$$\begin{aligned}
&\mathtt{hd}(\mathtt{f}(\mathtt{not}(\mathtt{S}))) = \mathtt{hd}(\mathtt{not}(\mathtt{f}(\mathtt{S}))) && \text{(symm. \& trans.)} \\
&\mathtt{hd}(\mathtt{tl}(\mathtt{f}(\mathtt{not}(\mathtt{S})))) = \mathtt{hd}(\mathtt{tl}(\mathtt{not}(\mathtt{f}(\mathtt{S})))) && \text{(symm. \& trans.)} \\
&\mathtt{tl}(\mathtt{tl}(\mathtt{f}(\mathtt{not}(\mathtt{S})))) = \mathtt{tl}(\mathtt{tl}(\mathtt{not}(\mathtt{f}(\mathtt{S})))) && \text{(subst. \& symm. \& trans.)}
\end{aligned}$$

Therefore $\Delta(G) \subseteq \mathcal{G}$, so we are done with the second proof as well.

The set G was straightforward to choose in the first proof in the example above, but it was not that easy in the second one. If we take G in the second proof to consist only of the initial goal like in the first proof, then we fail to prove that $\texttt{tl(f(not(S)))} = \texttt{tl(not(f(S)))}$ is in $\mathcal{G}$.

Unfortunately, there is no magic recipe to choose good sets G, which is what makes coinduction hard to automate. In fact, a naive enumeration of all sets G followed by an attempt to prove $\Delta(G) \subseteq \mathcal{G}$ leads to a Π^0_2 algorithm, which matches the worst case complexity of the behavioral satisfaction problem even for streams [14]. In practice, however, we can do better than enumerating all G. For example, after failing to prove $\texttt{tl(f(not(S)))} = \texttt{tl(not(f(S)))}$ in the second example above when one naively chooses $G = \{\texttt{f(not(S))} = \texttt{not(f(S))}\}$, then one can add the failed task to G and resume the task of proving that $\Delta(G) \subseteq \mathcal{G}$, this time successfully. We will see in the next section that this way of searching for a suitable G is the basis on which the circular coinductive reasoning is developed.

5 Circular Coinduction

A key notion in our formalization and even implementation of circular coinduction is that of a "frozen" equation. The motivation underlying frozen equations is that they structurally inhibit their use underneath proper contexts; because of that, they will allow us to capture the informal notion of "circular behavior" elegantly, rigorously, and generally (modulo a restricted form of equational reasoning). Formally, let $(\mathcal{B}, \Delta)$ be a behavioral specification and let us extend its signature Σ with a new sort *Frozen* and a new operation $\boxed{\text{-}} : s \to \mathit{Frozen}$ for each sort s. If t is a term, then we call $\boxed{t}$ the *frozen (form of) t*. Note that freezing only acts on the original sorts in Σ, so double freezing, e.g., $\boxed{\boxed{t}}$, is not allowed. If e is an equation $(\forall X)\, t = t'$ $\texttt{if}$ c, then we let $\boxed{e}$ be the *frozen equation* $(\forall X)\, \boxed{t} = \boxed{t'}$ $\texttt{if}$ c; note that the condition c stays unfrozen, but recall that we only assume visible conditions. By analogy, we call the equations over the original signature Σ *unfrozen equations*. If e is an (unfrozen) visible equation then $\boxed{e}$ is called a *frozen visible equation*; similarly when e is hidden. If C is a context for e, then we take the freedom to write $C[\boxed{e}]$ as a shortcut for $\boxed{C[e]}$. It is important to note here that if $E \cup \mathcal{F} \vdash \mathcal{G}$ for some unfrozen equation set E and frozen equation sets $\mathcal{F}$ and $\mathcal{G}$, then it is not necessarily the case that $E \cup \mathcal{F} \vdash C[\mathcal{G}]$ for a context C. Freezing therefore inhibits the free application of the congruence deduction rule of equational reasoning.

Recall that, for generality, we work with an arbitrary entailment system in this paper, which may or may not necessarily be the entailment relation of equational deduction. We next add two more axioms:

Definition 5. *A **Δ-contextual entailment system with freezing** is a Δ-contextual entailment system such that (below, E ranges over unfrozen equations, e over visible unfrozen equations, and $\mathcal{F}$ and $\mathcal{G}$ over frozen hidden equations):*

(A1) $E \cup \mathcal{F} \vdash \boxed{e}$ *iff* $E \vdash e$;

(A2) $E \cup \mathcal{F} \vdash \mathcal{G}$ *implies* $E \cup \delta[\mathcal{F}] \vdash \delta[\mathcal{G}]$ *for each* $\delta \in \Delta$*, equivalent to saying that for any Δ-context C,* $E \cup \mathcal{F} \vdash \mathcal{G}$ *implies* $E \cup C[\mathcal{F}] \vdash C[\mathcal{G}]$.

The first axiom says that frozen hidden equations do not interfere with the entailment of frozen or unfrozen visible equations. The second equation says that if some frozen hidden equations $\mathcal{F}$ can be used to derive some other frozen hidden equations $\mathcal{G}$, then one can also use the frozen equations $\delta[\mathcal{F}]$ to derive $\delta[\mathcal{G}]$. Freezing acts as a protective shell that inhibits applications of the congruence rule on frozen equations but instead allows the application of derivatives both onto the hypothesis and the conclusion of an entailment pair at the same time.

Therefore, our working entailment system $\vdash$ is now defined over both unfrozen and frozen equations. Note, however, that this extension is conservative, in that one cannot infer any new entailments of unfrozen equations that were not possible before; because of that, we take the liberty to use the same symbol, $\vdash$, for the extended entailment system. It is easy to check these additional axioms for concrete entailment relations. Consider, for example, the equational deduction entailment: (A1) follows by first noticing that $E \cup \mathcal{F} \vdash \boxed{e}$ iff $E \vdash \boxed{e}$ (because there is no way to make use of the equations in $\mathcal{F}$ in any derivation of $\boxed{e}$) and then that $E \vdash \boxed{e}$ iff $E \vdash e$ (because E contains no frozen equations); and (A2) holds because any proof π of a frozen hidden equation $\boxed{e_{\mathcal{G}}}$ in $\mathcal{G}$ from $E \cup \mathcal{F}$ can use the frozen equations in $\mathcal{F}$ only "at the top", that is, not underneath operators via an equational congruence deduction step, so one can simply replace any frozen term $\boxed{t}$ in π by $\boxed{\delta[t]}$ and thus get a proof for $\boxed{e_{\mathcal{G}}}$.

Theorem 2. ***(coinductive circularity principle)*** *If $\mathcal{B}$ is a behavioral specification and F is a set of hidden equations with $\mathcal{B} \cup \boxed{F} \vdash \boxed{\Delta[F]}$ then $\mathcal{B} \Vdash F$.*

Proof. We first prove by well-founded induction on the depth of C that the hypothesis of the theorem implies $\mathcal{B} \cup \boxed{F} \vdash \boxed{C[F]}$ for any Δ-context C. If C is a degenerated context $*{:}h$ then $C[F]$ is a subset of F (its equations of sort h), so the result follows by the fact that $\vdash$ is an entailment system. The case $C \in \Delta$ follows from the theorem hypothesis. If $C[*{:}h] = C_1[C_2[*{:}h]]$ for some proper contexts C_1 and C_2 strictly smaller in depth than C, then we have by the induction hypothesis that $\mathcal{B} \cup \boxed{F} \vdash \boxed{C_1[F]}$ and $\mathcal{B} \cup \boxed{F} \vdash \boxed{C_2[F]}$. Since $\vdash$ is an entailment system with freezing, by (A2) it follows that $\mathcal{B} \cup \boxed{C_1[F]} \vdash \boxed{C_1[C_2[F]]}$. Now, since $\mathcal{B} \cup \boxed{F} \vdash \boxed{C_1[F]}$ implies $\mathcal{B} \cup \boxed{F} \vdash \mathcal{B} \cup \boxed{C_1[F]}$, by the transitivity of $\vdash$ (both of these properties of entailment systems), we get $\mathcal{B} \cup \boxed{F} \vdash \boxed{C[F]}$.

Therefore, $\mathcal{B} \cup \boxed{F} \vdash \boxed{C[F]}$ for any Δ-context C, in particular $\mathcal{B} \cup \boxed{F} \vdash \boxed{C[F]}$ for any Δ-experiment C. Then by axiom (A1) of entailment systems with freezing we obtain that $\mathcal{B} \vdash C[e]$ for any $e \in F$ and any Δ-experiment C for which $C[e]$ is defined. Therefore, $\mathcal{B} \Vdash F$. ☐

Theorem 2 serves as the foundation of circular coinduction, because what circular coinduction essentially does is to iteratively attempt to complete a set of hidden equations that it starts with until it becomes a set F that satisfies the hypothesis of Theorem 2. Interestingly, Theorem 1 becomes now a simple corollary of Theorem 2: given a behaviorally closed $\mathcal{G}$ like in the hypothesis of Theorem 1, take F to be the set of all hidden equations in $\mathcal{G}$; then for an $e \in F$

of sort h and a $\delta[*{:}h] \in \Delta$, $\delta[e]$ either is visible and so $\mathcal{B} \vdash \delta[e]$ or is in F, so the hypothesis of Theorem 2 holds vacuously; therefore, $\mathcal{B} \Vvdash \mathcal{G}$. This is not entirely unexpected, because the F in Theorem 2 is stated to be a fixed point w.r.t. Δ-derivability; what may be slightly unexpected is that such fixed points can be safely calculated modulo reasoning into an entailment system with freezing, in particular modulo equational deduction with inhibited congruence.

Fig. 4 defines circular coinduction as a proof system for deriving pairs of the form $\mathcal{H} \Vvdash^{\circlearrowleft} \mathcal{G}$, where $\mathcal{H}$, the *hypotheses*, can contain both frozen and unfrozen equations, and where $\mathcal{G}$, the *goals*, contains only frozen equations. Initially, $\mathcal{H}$ is the original behavioral specification $\mathcal{B}$ and $\mathcal{G}$ is the frozen version $\boxed{G}$ of the original goals G to prove. The circular coinductive proving process proceeds as follows. At each moment when $\mathcal{G} \neq \emptyset$, a frozen equation $\boxed{e}$ is picked from $\mathcal{G}$. If $\mathcal{H} \vdash \boxed{e}$ holds, then $\boxed{e}$ may be discarded via a [Reduce] step. The core rule of circular coinduction is [Derive], which allows one to assume $\boxed{e}$ as hypothesis and generate $\boxed{\Delta[e]}$ as new proof obligations. This rule gives the user of our proof system the impression of circular reasoning, because one appears to assume what one wants to prove and go on. The key observation here is that the equation $\boxed{e}$ is assumed as hypothesis in its frozen form, which means that it cannot be freely used to prove its derivatives; otherwise, those would follow immediately by applying the congruence rule of equational deduction. One is done when the set of goals becomes empty. When that happens, one can conclude that $\mathcal{B} \Vvdash G$.

$$\frac{\cdot}{\mathcal{B} \cup \mathcal{F} \Vvdash^{\circlearrowleft} \emptyset} \qquad \textsf{[Done]}$$

$$\frac{\mathcal{B} \cup \mathcal{F} \Vvdash^{\circlearrowleft} \mathcal{G}, \quad \mathcal{B} \cup \mathcal{F} \vdash \boxed{e}}{\mathcal{B} \cup \mathcal{F} \Vvdash^{\circlearrowleft} \mathcal{G} \cup \{\boxed{e}\}} \qquad \textsf{[Reduce]}$$

$$\frac{\mathcal{B} \cup \mathcal{F} \cup \{\boxed{e}\} \Vvdash^{\circlearrowleft} \mathcal{G} \cup \boxed{\Delta[e]}}{\mathcal{B} \cup \mathcal{F} \Vvdash^{\circlearrowleft} \mathcal{G} \cup \{\boxed{e}\}}, \qquad \begin{array}{r}\textsf{[Derive]}\\ \text{if } e \text{ hidden}\end{array}$$

Fig. 4. Circular coinduction proof system: If $\mathcal{B} \Vvdash^{\circlearrowleft} \boxed{G}$ is derivable then $\mathcal{B} \Vvdash G$

The soundness of the circular coinductive proof system in Fig. 4, which is proved below, is a monolithic result depending upon a derivation of the complete proof for a task of the form $\mathcal{B} \Vvdash^{\circlearrowleft} \boxed{G}$. We do not know any way to decompose the proof of soundness by proving the soundness of each derivation rule, as it is commonly done in soundness proofs. For example, one may attempt to show that $\mathcal{H} \Vvdash^{\circlearrowleft} \mathcal{G}$ derivable implies $\mathcal{H}^\circ \Vvdash \mathcal{G}^\circ$, where $\mathcal{H}^\circ$ unfreezes all the frozen equations in $\mathcal{H}$ (and similarly for $\mathcal{G}^\circ$). Unfortunately, this simplistic approach cannot be used to show the [Derive] rule "sound", as it is *not* sound in this sense: indeed, $\mathcal{H}^\circ \cup \{e\} \Vvdash \mathcal{G}^\circ \cup \Delta[e]$ is equivalent to $\mathcal{H}^\circ \cup \{e\} \Vvdash \mathcal{G}^\circ$ because $\{e\} \Vvdash \Delta[e]$, and there is no way to show from these that $\mathcal{H}^\circ \Vvdash \mathcal{G}^\circ \cup \{e\}$. The soundness arguments of circular coinduction can be found in the complete proof, not in each particular derivation rule, which is what makes the theorem below unexpectedly hard to prove (note it also uses Theorem 2).

Theorem 3. ***(soundness of circular coinduction)*** *If $\mathcal{B}$ is a behavioral specification and G is a set of equations such that $\mathcal{B} \Vvdash^{\circlearrowleft} \boxed{G}$ is derivable using the proof system in Fig. 4, then $\mathcal{B} \Vvdash G$.*

Proof. Any derivation of $\mathcal{H} \Vvdash^{\circlearrowleft} \mathcal{G}$ using the proof system in Fig. 4 yields a sequence of pairs $\mathcal{H}_0 \Vvdash^{\circlearrowleft} \mathcal{G}_0$, $\mathcal{H}_1 \Vvdash^{\circlearrowleft} \mathcal{G}_1$, ... $\mathcal{H}_n \Vvdash^{\circlearrowleft} \mathcal{G}_n$, where $\mathcal{H}_0 = \mathcal{B}$, $\mathcal{G}_0 = \boxed{G}$,

$\mathcal{G}_n = \emptyset$, and for every $0 \leq i < n$, there is some $\boxed{e} \in \mathcal{G}_i$ such that one of the following holds, each corresponding to one of the rules [Reduce] or [Derive]:

[Reduce] $\mathcal{H}_i \vdash \boxed{e}$ and $\mathcal{G}_{i+1} = \mathcal{G}_i - \{\boxed{e}\}$ and $\mathcal{H}_{i+1} = \mathcal{H}_i$; or
[Derive] e is hidden and $\mathcal{G}_{i+1} = (\mathcal{G}_i - \{\boxed{e}\}) \cup \boxed{\Delta[e]}$ and $\mathcal{H}_{i+1} = \mathcal{H}_i \cup \boxed{e}$.

Let $\mathcal{G} = \bigcup_{i=0}^{n} \mathcal{G}_i$, let $\mathcal{G}^\circ = \{e \mid \boxed{e} \in \mathcal{G}\}$, and let $F = \mathit{hidden}(\mathcal{G}^\circ)$. Note that for each $0 \leq i < n$, $\mathcal{H}_i = \mathcal{B} \cup \mathcal{F}_i$ for some set of frozen hidden equations $\mathcal{F}_i$ with $\mathcal{F}_i \cup \Delta[\mathcal{F}_i] \subseteq \mathcal{G}$: indeed, only frozen hidden equations are added to $\mathcal{H}$ and only by the rule [Derive], which also adds at the same time the derivatives of those equations to $\mathcal{G}$. This implies that if i corresponds to a [Reduce] step with $\mathcal{H}_i \vdash \boxed{e}$ for some $\boxed{e} \in \mathcal{G}$, then either $\mathcal{B} \vdash e$ by (A1) when e is visible, or $\mathcal{B} \cup \Delta[\mathcal{F}_i] \vdash \boxed{\Delta[e]}$ by (A2) when $e \in F$.

If $e \in \mathcal{G}^\circ$ visible, then there must be some $0 \leq i < n$ such that $\mathcal{H}_i \vdash \boxed{e}$, so $\mathcal{B} \vdash e$. Since $\Delta[\mathcal{F}_i] \subseteq \mathcal{G}$, equations $\boxed{f} \in \Delta[\mathcal{F}_i]$ either are visible and so $\mathcal{B} \vdash f$, or are hidden and $\boxed{f} \in \boxed{F}$; in either case, we conclude that $\mathcal{B} \cup \boxed{F} \vdash \Delta[\mathcal{F}_i]$ for any $0 \leq i < n$, so $\mathcal{B} \cup \boxed{F} \vdash \boxed{\Delta[e]}$ for any $e \in F$ such that $\mathcal{H}_i \vdash \boxed{e}$ in some [Reduce] rule applied at step $0 \leq i < n$. If $e \in F$ such that a $\boxed{\delta[e]}$ for each appropriate $\delta \in \Delta$ is added to $\mathcal{G}$ via a [Derive] rule, then it is either that $\delta[e] \in \mathcal{G}^\circ$ and so $\mathcal{B} \vdash \delta[e]$, or that $\delta[e] \in F$; in either case, for such $e \in F$ we conclude that $\mathcal{B} \cup \boxed{F} \vdash \boxed{\Delta[e]}$. We covered all cases for $e \in F$, so $\mathcal{B} \cup \boxed{F} \vdash \boxed{\Delta[F]}$; then Theorem 2 implies that $\mathcal{B} \Vdash F$. Since F contains all the hidden equations of $\mathcal{G}^\circ$ and since we already proved that $\mathcal{B} \vdash e$, i.e., $\mathcal{B} \Vdash e$, for all $e \in \mathcal{G}^\circ$ visible, we conclude that $\mathcal{B} \Vdash \mathcal{G}^\circ$. Since $G \subseteq \mathcal{G}^\circ$, it follows that $\mathcal{B} \Vdash G$. $\square$

Example 4. Let us now derive by circular coinduction the two stream properties proved manually by coinduction in Example 3, as well as the fixed point property of the `morse` stream defined in Section 1.

The proof tree below summarizes the derivation steps for the first property. We only show the proof steps using the circular coinduction rules, that is, we omit the proof steps associated to the base entailment system $\vdash$:

$$\begin{array}{lll}
\hline
\mathtt{STREAM} \cup \left\{\boxed{\mathtt{zip(odd(S),even(S))}} = \boxed{\mathtt{S}}\right\} & \Vdash^{\circlearrowleft} & \emptyset \\
\mathtt{STREAM} \cup \left\{\boxed{\mathtt{zip(odd(S),even(S))}} = \boxed{\mathtt{S}}\right\} & \vdash & \boxed{\mathtt{hd(zip(odd(S),even(S)))}} = \boxed{\mathtt{hd(S)}} \\
\hline
\mathtt{STREAM} \cup \left\{\boxed{\mathtt{zip(odd(S),even(S))}} = \boxed{\mathtt{S}}\right\} & \Vdash^{\circlearrowleft} & \left\{\boxed{\mathtt{hd(zip(odd(S),even(S)))}} = \boxed{\mathtt{hd(S)}}\right\} \\
\mathtt{STREAM} \cup \left\{\boxed{\mathtt{zip(odd(S),even(S))}} = \boxed{\mathtt{S}}\right\} & \vdash & \boxed{\mathtt{tl(zip(odd(S),even(S)))}} = \boxed{\mathtt{tl(S)}} \\
\hline
\mathtt{STREAM} \cup \left\{\boxed{\mathtt{zip(odd(S),even(S))}} = \boxed{\mathtt{S}}\right\} & \Vdash^{\circlearrowleft} & \left\{\begin{array}{l}\boxed{\mathtt{hd(zip(odd(S),even(S)))}} = \boxed{\mathtt{hd(S)}}, \\ \boxed{\mathtt{tl(zip(odd(S),even(S)))}} = \boxed{\mathtt{tl(S)}}\end{array}\right\} \\
\hline
\mathtt{STREAM} & \Vdash^{\circlearrowleft} & \left\{\boxed{\mathtt{zip(odd(S),even(S))}} = \boxed{\mathtt{S}}\right\}
\end{array}$$

The top horizontal line marks a [Done] step. The second one marks a [Reduce] step which follows by equational reasoning using only the equations of `STREAM`.

The third horizontal line marks also a [Reduce] step but where the frozen hypothesis is used this time: the goal $\mathtt{tl(zip(odd(S), even(S)))} = \mathtt{tl(S)}$ is reduced first to $\mathtt{zip(odd(tl(S)), even(tl(S)))} = \mathtt{tl(S)}$ using the equations of STREAM and then to $\mathtt{tl(S)} = \mathtt{tl(S)}$ using the frozen hypothesis. The bottom horizontal line marks a [Derive] step: the initial goal is added as a frozen hypothesis and its derivatives become the new goals. In terms of the proof of Theorem 3, we have $\mathcal{B} = \mathtt{STREAM}$, $F_i = \{\mathtt{zip(odd(S), even(S))} = \mathtt{S}\}$ for $i = 1, 2, 3$, $\mathcal{G}_1^\circ = \{\mathtt{hd(zip(odd(S), even(S)))} = \mathtt{hd(S)}, \mathtt{tl(zip(odd(S), even(S)))} = \mathtt{tl(S)}\}$, $\mathcal{G}_2^\circ = \{\mathtt{hd(zip(odd(S), even(S)))} = \mathtt{hd(S)}\}$, and $\mathcal{G}_3^\circ = \emptyset$. We observe that the circular coinduction proof system can be easily mechanized: the above proof tree can be automatically built in a bottom-up manner. This feature is fully exploited by the CIRC tool [10].

Similarly, next proof tree shows a derivation of $\mathtt{STREAM} \Vvdash^{\circlearrowleft} \mathtt{f(not(S))} = \mathtt{not(f(S))}$. The following notations are used: $F_1 = \{\mathtt{f(not(S))} = \mathtt{not(f(S))}\}$, $F_2 = F_1 \cup \{\mathtt{tl(f(not(S)))} = \mathtt{tl(not(f(S)))}\}$. Note that this time the index of F does not reflect the step number, as it is the case in the proof of Theorem 3.

$$
\begin{array}{lll}
\hline
\mathtt{STREAM} \cup \boxed{F_2} & \Vvdash^{\circlearrowleft} & \emptyset \\[4pt]
\mathtt{STREAM} \cup \boxed{F_2} & \vdash & \boxed{\mathtt{tl}^2\mathtt{(f(not(S)))}} = \boxed{\mathtt{tl}^2\mathtt{(not(f(S)))}} \\
\hline
\mathtt{STREAM} \cup \boxed{F_2} & \Vvdash^{\circlearrowleft} & \left\{\boxed{\mathtt{tl}^2\mathtt{(f(not(S)))}} = \boxed{\mathtt{tl}^2\mathtt{(not(f(S)))}}\right\} \\[4pt]
\mathtt{STREAM} \cup \boxed{F_2} & \vdash & \boxed{\mathtt{hd(tl(f(not(S))))}} = \boxed{\mathtt{hd(tl(not(f(S))))}} \\
\hline
\mathtt{STREAM} \cup \boxed{F_2} & \Vvdash^{\circlearrowleft} & \boxed{\Delta(\mathtt{tl(f(not(S)))} = \mathtt{tl(not(f(S)))})} \\
\hline
\mathtt{STREAM} \cup \boxed{F_1} & \Vvdash^{\circlearrowleft} & \left\{\boxed{\mathtt{tl(f(not(S)))}} = \boxed{\mathtt{tl(not(f(S)))}}\right\} \\[4pt]
\mathtt{STREAM} \cup \boxed{F_1} & \vdash & \boxed{\mathtt{hd(f(not(S)))}} = \boxed{\mathtt{hd(not(f(S)))}} \\
\hline
\mathtt{STREAM} \cup \boxed{F_1} & \Vvdash^{\circlearrowleft} & \boxed{\Delta(F_1)} \\
\hline
\mathtt{STREAM} & \Vvdash^{\circlearrowleft} & \boxed{F_1}
\end{array}
$$

The top-down sequence of the applied proof rules is: [Done], [Reduce], [Reduce], [Derive], [Reduce], [Derive]. The first [Reduce] uses a frozen hypothesis from $\boxed{F_2}$. The goal $\mathtt{tl}^2\mathtt{(f(not(S)))} = \mathtt{tl}^2\mathtt{(not(f(S)))}$ is reduced to $\mathtt{f(not(tl(S)))} = \mathtt{not(f(tl(S)))}$ by equational reasoning using only the equations of STREAM, and then to $\mathtt{not(f(tl(S)))} = \mathtt{not(f(tl(S)))}$ using the frozen hypothesis from $\boxed{F_1} \subset \boxed{F_2}$. Note that the first [Derive] step in the above sequence exhibits how the circular coinduction has automatically "discovered" the second equation that was necessary to prove the property by plain coinduction in Example 3.

Finally, next proof tree shows a circular coinductive proof that, with the streams defined in Section 1, `morse` is a fixed point of `f`. Note that one cannot prove directly the fixed point property (see [10] for how CIRC fails to prove it); instead, we prove $G = \{\mathtt{f(morse)} = \mathtt{morse},\ (\forall \mathtt{S})\, \mathtt{zip(S,\ not(S))} = \mathtt{f(S)}\}$. The description of the proof tree uses the following notations:

$$\begin{aligned}
F_1 &= \{\mathtt{f(morse)} = \mathtt{morse}, \mathtt{zip(S,\ not\ S)} = \mathtt{f(S)}\}\\
F_2 &= F_1 \cup \{\mathtt{tl(f(morse))} = \mathtt{tl(morse)}\}\\
F_3 &= F_2 \cup \{\mathtt{tl(zip(S,\ not\ S))} = \mathtt{tl(f(S))}\}
\end{aligned}$$

$$\begin{array}{l}
\hline
\mathtt{STREAM} \cup \boxed{F_3} \Vvdash^{\circlearrowleft} \emptyset \\
\mathtt{STREAM} \cup \boxed{F_3} \vdash \boxed{\mathtt{tl(tl(zip(S,\ not\ S)))}} = \boxed{\mathtt{tl(tl(f(S)))}} \\
\hline
\mathtt{STREAM} \cup \boxed{F_3} \Vvdash^{\circlearrowleft} \boxed{\mathtt{tl}^2\mathtt{(zip(S,\ not\ S))}} = \boxed{\mathtt{tl}^2\mathtt{(f(S))}} \\
\mathtt{STREAM} \cup \boxed{F_3} \vdash \boxed{\mathtt{hd(tl(zip(S,\ not\ S)))}} = \boxed{\mathtt{hd(tl(f(S)))}} \\
\hline
\mathtt{STREAM} \cup \boxed{F_3} \Vvdash^{\circlearrowleft} \boxed{\mathit{\Delta}(\mathtt{tl(zip(S,\ not\ S))} = \mathtt{tl(f(S))})} \\
\mathtt{STREAM} \cup \boxed{F_3} \vdash \boxed{\mathtt{tl}^2\mathtt{(f(morse))}} = \boxed{\mathtt{tl}^2\mathtt{(morse)}} \\
\hline
\mathtt{STREAM} \cup \boxed{F_3} \Vvdash^{\circlearrowleft} \boxed{\mathit{\Delta}(\mathtt{tl(zip(S,not\,S))} = \mathtt{tl(f(S))})} \cup \left\{\boxed{\mathtt{tl}^2\mathtt{(f(morse))}} = \boxed{\mathtt{tl}^2\mathtt{(morse)}}\right\} \\
\hline
\mathtt{STREAM} \cup \boxed{F_2} \Vvdash^{\circlearrowleft} \left\{\boxed{\mathtt{tl(zip(S,not\,S))}} = \boxed{\mathtt{tl(f(S))}}\right\} \cup \left\{\boxed{\mathtt{tl}^2\mathtt{(f(morse))}} = \boxed{\mathtt{tl}^2\mathtt{(morse)}}\right\} \\
\mathtt{STREAM} \cup \boxed{F_2} \vdash \boxed{\mathtt{hd(tl(f(morse)))}} = \boxed{\mathtt{hd(tl(morse))}} \\
\hline
\mathtt{STREAM} \cup \boxed{F_2} \Vvdash^{\circlearrowleft} \left\{\boxed{\mathtt{tl(zip(S,\ not\ S))}} = \boxed{\mathtt{tl(f(S))}}\right\} \cup \boxed{\mathit{\Delta}(\mathtt{tl(f(morse))} = \mathtt{tl(morse)})} \\
\hline
\mathtt{STREAM} \cup \boxed{F_1} \Vvdash^{\circlearrowleft} \left\{\boxed{\mathtt{tl(zip(S,\ not\ S))}} = \boxed{\mathtt{tl(f(S))}}, \boxed{\mathtt{tl(f(morse))}} = \boxed{\mathtt{tl(morse)}}\right\} \\
\mathtt{STREAM} \cup \boxed{F_1} \vdash \boxed{\mathtt{hd(zip(S,\ not\ S))}} = \boxed{\mathtt{hd(f(S))}} \\
\mathtt{STREAM} \cup \boxed{F_1} \vdash \boxed{\mathtt{hd(f(morse))}} = \boxed{\mathtt{hd(morse)}} \\
\hline
\mathtt{STREAM} \cup \boxed{F_1} \Vvdash^{\circlearrowleft} \boxed{\mathit{\Delta}(F_1)} \\
\hline
\mathtt{STREAM} \qquad \Vvdash^{\circlearrowleft} \boxed{F_1}
\end{array}$$

For the sake of presentation, the last [Reduce] discards two goals, representing in fact two proof steps.

6 Conclusion

Previous formalizations of circular coinduction were either algorithmic in nature, or limited. For example, [5] introduces circular coinductive rewriting as an operational technique to extend rewriting with coinductive steps. On the other hand, [6] attempts to capture circular coinduction as a proof rule, but, unfortunately, it only works with properties that need at most one derivation step and it is melted away within a particular entailment system for hidden algebra, making it hard to understand what circular coinduction really is.

This paper presented circular coinduction as a sound, generic, self-contained and easy to understand proof system. We believe that this result will enhance understanding of circular coinduction, will allow it to be applicable to various coalgebraic settings, and will lead to improved implementations and extensions.

Acknowledgment. The paper is supported in part by NSF grants CCF-0448501, CNS-0509321 and CNS-0720512, by NASA contract NNL08AA23C, and by CNCSIS grant PN-II-ID-393.

References

1. J.-P. Allouche and J. Shallit. The ubiquitous Prouhet-Thue-Morse sequence. In T. H. C. Ding and H. Niederreiter, editors, *Sequences and Their applications (Proc. SETA'98)*, pages 1–16. Springer-Verlag, 1999.
2. M. Bidoit and R. Hennicker. Constructor-based observational logic. *J. Log. Algebr. Program.*, 67(1-2):3–51, 2006.
3. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, editors. *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.
4. M.-C. Gaudel and I. Privara. Context induction: an exercise. Technical Report 687, LRI, Université de Paris-Sud, 1991.
5. J. Goguen, K. Lin, and G. Roşu. Circular coinductive rewriting. In *ASE '00: Proceedings of the 15th IEEE international conference on Automated software engineering*, pages 123–132, Washington, DC, USA, 2000. IEEE Computer Society.
6. J. Goguen, K. Lin, and G. Roşu. Conditional circular coinductive rewriting with case analysis. In M. Wirsing, D. Pattinson, and R. Hennicker, editors, *WADT*, volume 2755 of *Lecture Notes in Computer Science*, pages 216–232. Springer, 2002.
7. J. Goguen and J. Meseguer. Completeness of Many-Sorted Equational Logic. *Houston Journal of Mathematics*, 11(3):307–334, 1985.
8. D. Hausmann, T. Mossakowski, and L. Schröder. Iterative circular coinduction for CoCasl in Isabelle/HOL. In M. Cerioli, editor, *FASE*, volume 3442 of *Lecture Notes in Computer Science*, pages 341–356. Springer, 2005.
9. R. Hennicker. Context induction: a proof principle for behavioral abstractions. *Formal Aspects of Computing*, 3(4):326–345, 1991.
10. D. Lucanu, E.-I. Goriac, G. Caltais, and G. Roşu. CIRC : A behavioral verification tool based on circular coinduction. In *CALCO'09*, LNCS, 2009. This volume.
11. D. Lucanu and G. Roşu. CIRC : A circular coinductive prover. In T. Mossakowski, U. Montanari, and M. Haveraaen, editors, *CALCO*, volume 4624 of *Lecture Notes in Computer Science*, pages 372–378. Springer, 2007.
12. J. Meseguer. General logics. In H.-D. E. et al., editor, *Logic Colloquium '87*, pages 275–329, North Holland, Amsterdam, 1989.
13. G. Roşu. *Hidden Logic*. PhD thesis, University of California at San Diego, 2000.
14. G. Roşu. Equality of streams is a Π_2^0-complete problem. In *Proceedings of the 11th ACM SIGPLAN International Conference on Functional Programming (ICFP'06)*. ACM, 2006.
15. G. Roşu and J. Goguen. Circular coinduction. 2001. Short paper at the *International Joint Conference on Automated Reasoning (IJCAR'01)*.