Circular Coinduction
–A Proof Theoretical Foundation–

Grigore Roșu\textsuperscript{1}  Dorel Lucanu\textsuperscript{2}

\textsuperscript{1}Department of Computer Science
University of Illinois at Urbana-Champaign, USA
grosu@illinois.edu

\textsuperscript{2}Faculty of Computer Science
Alexandru Ioan Cuza University, Iași, Romania
dlucanu@info.uaic.ro

08/09/2009, CALCO 2009, Udine
1. Introduction
   - CC History
   - Behavioral Equivalence, intuitively
   - Behavioral Specifications, intuitively
   - Circular Coinduction, intuitively

2. Circular Coinduction Proof System
   - Formal Framework
   - Coinductive Circularity Principle
   - The Proof System

3. Conclusion
Plan

1. Introduction
   - CC History
   - Behavioral Equivalence, intuitively
   - Behavioral Specifications, intuitively
   - Circular Coinduction, intuitively

2. Circular Coinduction Proof System
   - Formal Framework
   - Coinductive Circularity Principle
   - The Proof System

3. Conclusion
Circular Coinduction: History

1998 first implementation of CC in BOBJ system [J. Goguen & K. Lin & G. Roșu, ASE 2000]

2000 CC formalized as a inference rule enriching hidden logic [G. Roșu & J. Goguen, written in 1999]

2002 CC described as a more complex algorithm [J. Goguen & K. Lin & G. Roșu, WADT 2002]
(a first version for special contexts, case analysis)


2006 CC implemented in Maude (first version of CIRC) [D. Lucanu & A. Popescu & G. Roșu]

2007 first major refactoring of CIRC [CALCO Tools, 2007]
(Maude meta-language application, regular strategies as proof tactics, simplification rules)

2009 CC formalized as a proof system [CALCO 2009, this paper] – second major refactoring of CIRC [CALCO Tools, 2009]
Behavioral Equivalence: Intuition 1/2

Behavioral equivalence is the non-distinguishability under experiments

Example of streams:

- a stream (of bits) \( S \) is an infinite sequence \( b_1 : b_2 : b_3 : \ldots \)
  - the head of \( S \): \( \text{hd}(S) = b_1 \)
  - the tail of \( S \): \( \text{tl}(S) = b_2 : b_3 : \ldots \)

- experiments:
  \( \text{hd}(*:\text{Stream}), \text{hd}(\text{tl}(*:\text{Stream})), \text{hd}(\text{tl}(\text{tl}(*:\text{Stream}))), \ldots \)

- the basic elements upon on the experiments are built (here \( \text{hd}(*) \) and \( \text{tl}(*) \)) are called derivatives

- application of an experiment over a stream: \( C[S] = C[S/*] \)

- two streams \( S \) and \( S' \) are behavioral equivalent \( (S \equiv S') \) iff \( C[S] = C[S'] \) for each exp. \( C \)

- for this particular case, beh. equiv. is the same with the equality of streams

- showing beh. equiv. is \( \Pi^0_2 \)-hard (S. Buss, G. Roșu, 2000, 2006)
Behavioral Equivalence: Intuition 2/2

(not in this paper)

Example of infinite binary trees (over bits):

- A infinite binary tree over $D$ is a function $T: \{L, R\}^* \rightarrow D$
- The root of $T$: $T(\varepsilon)$
- The left subtree $T_\ell$: $T_\ell(w) = T(Lw)$ for all $w$
- The right subtree $T_r$: $T_r(w) = T(Rw)$ for all $w$
- Knowing the root $d$, $T_\ell$ and $T_r$, then $T$ can be written as $d / T_\ell, T_r \setminus$
- The derivatives: $root(\ast: \text{Tree})$, $left(\ast: \text{Tree})$, and $right(\ast: \text{Tree})$
- The experiments: $root(\ast: \text{Tree})$, $root(left(\ast: \text{Tree}))$, $root(right(\ast: \text{Tree}))$ and so on
- Two trees $T$ and $T'$ are beh. equiv. ($T \equiv T'$) iff $C[T] = C[T']$ for each exp. $C$
Behavioral Specifications: Intuition 1/2

Streams:
- derivatives: $hd(* : Stream)$ and $tl(* : Stream)$
- beh specs are derivative-based specs

<table>
<thead>
<tr>
<th>Corecursive spec</th>
<th>Behavioral spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>zeroes $= 0 : zeroes$</td>
<td>$hd(zeroes) = 0$</td>
</tr>
<tr>
<td></td>
<td>$tl(zeroes) = zeroes$</td>
</tr>
<tr>
<td>ones $= 1 : ones$</td>
<td>$hd(ones) = 1$</td>
</tr>
<tr>
<td></td>
<td>$tl(ones) = ones$</td>
</tr>
<tr>
<td>blink $= 0 : 1 : blink$</td>
<td>$hd(blink) = 0$</td>
</tr>
<tr>
<td></td>
<td>$tl(blink) = 1 : blink$</td>
</tr>
<tr>
<td>zip($B : S, S'$) $= B : zip(S', S)$</td>
<td>$hd(zip(S, S')) = hd(S)$</td>
</tr>
<tr>
<td></td>
<td>$tl(S, S') = zip(S', S)$</td>
</tr>
</tbody>
</table>

- for streams, this can be done with STR tool (see H. Zantema’s tool paper)
Behavioral Specifications: Intuition 2/2

Infinite binary trees (TREE):

- derivatives: \( \text{root}(\ast: \text{Tree}) \), \( \text{left}(\ast: \text{Tree}) \), and \( \text{right}(\ast: \text{Tree}) \)
- beh specs are derivative-based specs

<table>
<thead>
<tr>
<th>Corecursive spec</th>
<th>Behavioral spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ones} = 1/\text{ones}, \text{ones} ) | ( \text{root}(\text{ones}) = 1 )</td>
<td></td>
</tr>
<tr>
<td>( \text{ones} )</td>
<td>( \text{left}(\text{ones}) = \text{ones} )</td>
</tr>
<tr>
<td>( \text{ones} )</td>
<td>( \text{right}(\text{ones}) = \text{ones} )</td>
</tr>
<tr>
<td>( b/\text{T}<em>\ell, \text{T}<em>r ) + ( b'/\text{T}'</em>\ell, \text{T}'<em>r ) = ( b \lor b'/\text{T}</em>\ell + \text{T}'</em>\ell, \text{T}_r + \text{T}'_r ) | ( \text{root}(T + T') = \text{root}(T) \lor \text{root}(T) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \text{left}(T + T') = \text{left}(T) + \text{left}(T') )</td>
</tr>
<tr>
<td></td>
<td>( \text{right}(T + T') = \text{right}(T) + \text{right}(T') )</td>
</tr>
<tr>
<td>( \text{thue} = 0/\text{thue}, \text{thue} + \text{one} ) | ( \text{root}(\text{thue}) = 0 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \text{left}(\text{thue}) = \text{thue} )</td>
</tr>
<tr>
<td></td>
<td>( \text{right}(\text{thue}) = \text{thue} + \text{one} )</td>
</tr>
</tbody>
</table>
Circular Coinduction: Intuition 1/2

– the goal is to prove that \( \text{zip}(\text{zeroes}, \text{ones}) \equiv \text{blink} \) holds in STREAM

\[
\begin{align*}
\text{zip}(\text{zeroes}, \text{ones}) &= \text{blink} \\
0 &= 0 & \sqrt{\text{zip}(\text{ones}, \text{zeroes}) &= 1:\text{blink} \\
1 &= 1 & \sqrt{\text{zip}(\text{zeroes}, \text{ones}) &= \text{blink}}
\end{align*}
\]
Circular Coinduction: Intuition 2/2

– the goal is to prove that $\text{ones} + T \equiv \text{ones}$ holds in TREE

\[
\text{ones} + T = \text{ones}
\]

\[
\text{ones} + \text{left}(T) = \text{ones} \quad \text{ones} + \text{right}(T) = \text{ones}
\]

– a more challenging property: $\text{thue} + \text{one} = \text{not}(\text{thue})$
Plan

1. Introduction
   - CC History
   - Behavioral Equivalence, intuitively
   - Behavioral Specifications, intuitively
   - Circular Coinduction, intuitively

2. Circular Coinduction Proof System
   - Formal Framework
   - Coinductive Circularity Principle
   - The Proof System

3. Conclusion
Formal Framework 1/2

A behavioral specification consists of:

- a many-sorted algebraic spec $\mathcal{B} = (S, \Sigma, E)$
  
  $S$ = set of sorts, $\Sigma$ = set of opns, $E$ = set of eqns

- a set of derivatives $\Delta = \{\delta[\ast:h]\}$
  
  $\delta[\ast:h]$ is a context
  
  the sort $h$ of the special variable $\ast$ occurring in a derivative $\delta$ is called hidden; the other sorts are called visible

- each derivative can be seen as an equation transformer:
  
  if $e$ is $t = t'$ iff $\text{cond}$, then $\delta[e]$ is $\delta[t] = \delta[t']$ iff $\text{cond}$

  $\Delta[e] = \{\delta[e] | \delta \in \Delta\}$

- an entailment relation $\vdash$, which is reflexive, transitive, monotonic, and $\Delta$-congruent ($E \vdash e$ implies $E \vdash \Delta[e]$)
Experiment: each visible $\delta[\ast:h] \in \Delta$ is an experiment, and if $C[\ast:h']$ is an experiment and $\delta[\ast:h] \in \Delta$, then so is $C[\delta[\ast:h]]$

Behavioral satisfaction: $B \models e$ iff: $B \vdash e$, if $e$ is visible, and $B \vdash C[e]$ for each experiment $C$, if $e$ is hidden

Behavioral equivalence of $B$: $\equiv_B \overset{\text{def}}{=} \{ e \mid B \models e \}$

A set of equations $\mathcal{G}$ is behaviorally closed iff $B \vdash \text{visible}(\mathcal{G})$ and $\Delta(\mathcal{G} - B^\bullet) \subseteq \mathcal{G}$, where $B^\bullet = \{ e \mid B \vdash e \}$

Theorem

(coinduction) The behavioral equivalence $\equiv$ is the largest behaviorally closed set of equations.
The Freezing Operator

- is the most important ingredient of CC
- it inhibits the use of the coinductive hypothesis underneath proper contexts;
- if $e$ is $t = t'$ if $\text{cond}$, then its frozen form is $t = t'$ if $\text{cond}$

(A1) $E \cup F \vdash e$ iff $E \vdash e$, for each visible eqn $e$;
(A2) $E \cup F \vdash G$ implies $E \cup \delta[F] \vdash \delta[G]$ for each $\delta \in \Delta$, equivalent to saying that for any $\Delta$-context $C$, $E \cup F \vdash G$ implies $E \cup C[F] \vdash C[G]$.

Theorem

(coinductive circularity principle) If $\mathcal{B}$ is a behavioral specification and $F$ is a set of hidden equations with $\mathcal{B} \cup F \vdash \Delta[F]$ then $\mathcal{B} \not\vdash F$. 

G. Roșu, D. Lucanu (UIUC, UAIC)
Circular Coinduction Proof System

\[ B \cup F \downarrow \left\downarrow \emptyset \]

\[ B \cup F \downarrow \left\downarrow G, \quad B \cup F \vdash \{e\} \]

\[ B \cup F \downarrow \left\downarrow G \cup \Delta[e] \]

if \( e \) hidden

[Done]

[Reduce]

[Derive]
Soundness

Theorem

(soundness of circular coinduction) \( \text{If } B \text{ is a behavioral specification and } G \text{ is a set of equations such that } B \vDash \mathcal{C} G \text{ is derivable using the Circular Coinduction Proof System, then } B \vDash G. \)

The proof is \textit{monolithic} and, intuitively, the correctness can be explained in different ways:

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 by applications of derivatives;

2. the obtained circular graph structure can be used as a backbone to "consume" any possible experiment applied on the two original terms;

3. the equalities that appear as nodes in the obtained graph can be regarded as lemmas inferred in order to prove the original task;

4. when it stabilizes, it "discovers" a relation which is compatible with the derivatives and is the identity on data, so the stabilized set of equations is included in the behavioral equivalence;

5. it incrementally completes a given equality into a bisimulation relation on terms.
Soundness

Theorem

(soundness of circular coinduction) If $B$ is a behavioral specification and $G$ is a set of equations such that $B \circ\rhd G$ is derivable using the Circular Coinduction Proof System, then $B \parallel G$.

The proof is monolithic and, intuitively, the correctness can be explained in different ways:

(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 by applications of derivatives;

(2) the obtained circular graph structure can be used as a backbone to “consume” any possible experiment applied on the two original terms;

(3) the equalities that appear as nodes in the obtained graph can be regarded as lemmas inferred in order to prove the original task;

(4) when it stabilizes, it “discovers” a relation which is compatible with the derivatives and is the identity on data, so the stabilized set of equations is included in the behavioral equivalence;

(5) it incrementally completes a given equality into a bisimulation relation on terms.
Soundness

Theorem

(soundness of circular coinduction) If $B$ is a behavioral specification and $G$ is a set of equations such that $B \triangleright\triangleleft G$ is derivable using the Circular Coinduction Proof System, then $B \triangleright\triangleleft G$.

The proof is monolithic and, intuitively, the correctness can be explained in different ways:

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 by applications of derivatives;

2. the obtained circular graph structure can be used as a backbone to “consume” any possible experiment applied on the two original terms;

3. the equalities that appear as nodes in the obtained graph can be regarded as lemmas inferred in order to prove the original task;

4. when it stabilizes, it “discovers” a relation which is compatible with the derivatives and is the identity on data, so the stabilized set of equations is included in the behavioral equivalence;

5. it incrementally completes a given equality into a bisimulation relation on terms.
Soundness

Theorem

(soundness of circular coinduction) If $B$ is a behavioral specification and $G$ is a set of equations such that $B \vdash \Box G$ is derivable using the Circular Coinduction Proof System, then $B \vdash G$.

The proof is monolithic and, intuitively, the correctness can be explained in different ways:

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 by applications of derivatives;

2. the obtained circular graph structure can be used as a backbone to “consume” any possible experiment applied on the two original terms;

3. the equalities that appear as nodes in the obtained graph can be regarded as lemmas inferred in order to prove the original task;

4. when it stabilizes, it “discovers” a relation which is compatible with the derivatives and is the identity on data, so the stabilized set of equations is included in the behavioral equivalence;

5. it incrementally completes a given equality into a bisimulation relation on terms.

G. Roșu, D. Lucanu (UIUC, UAIC)
Soundness

**Theorem**

*(soundness of circular coinduction)* If $B$ is a behavioral specification and $G$ is a set of equations such that $B \vdash \bigcirc G$ is derivable using the Circular Coinduction Proof System, then $B \vdash G$.

The proof is *monolithic* and, intuitively, the correctness can be explained in different ways:

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 by applications of derivatives;

2. the obtained circular graph structure can be used as a backbone to “consume” any possible experiment applied on the two original terms;

3. the equalities that appear as nodes in the obtained graph can be regarded as lemmas inferred in order to prove the original task;

4. when it stabilizes, it “discovers” a relation which is compatible with the derivatives and is the identity on data, so the stabilized set of equations is included in the behavioral equivalence;

5. it incrementally completes a given equality into a bisimulation relation on terms.
Soundness

Theorem

(soundness of circular coinduction) If $B$ is a behavioral specification and $G$ is a set of equations such that $B \vdash \circ \boxdot G$ is derivable using the Circular Coinduction Proof System, then $B \vdash G$.

The proof is monolithic and, intuitively, the correctness can be explained in different ways:

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 by applications of derivatives;

2. the obtained circular graph structure can be used as a backbone to “consume” any possible experiment applied on the two original terms;

3. the equalities that appear as nodes in the obtained graph can be regarded as lemmas inferred in order to prove the original task;

4. when it stabilizes, it “discovers” a relation which is compatible with the derivatives and is the identity on data, so the stabilized set of equations is included in the behavioral equivalence;

5. it incrementally completes a given equality into a bisimulation relation on terms.
### Example

<table>
<thead>
<tr>
<th>STREAM $\cup { \text{odd}(S), \text{even}(S) } = S }$</th>
<th>$\models \not\vdash \emptyset$</th>
<th>[Done]</th>
</tr>
</thead>
<tbody>
<tr>
<td>STREAM $\cup { \text{odd}(S), \text{even}(S) } = S }$</td>
<td>$\vdash \text{hd}(\text{zip}(\text{odd}(S), \text{even}(S))) = \text{hd}(S)$</td>
<td>[Reduce]</td>
</tr>
<tr>
<td>STREAM $\cup { \text{odd}(S), \text{even}(S) } = S }$</td>
<td>$\vdash \text{tl}(\text{zip}(\text{odd}(S), \text{even}(S))) = \text{tl}(S)$</td>
<td>[Reduce]</td>
</tr>
<tr>
<td>STREAM $\cup { \text{odd}(S), \text{even}(S) } = S }$</td>
<td>$\vdash \begin{cases} \text{hd}(\text{zip}(\text{odd}(S), \text{even}(S))) = \text{hd}(S), \ \text{tl}(\text{zip}(\text{odd}(S), \text{even}(S))) = \text{tl}(S) \end{cases}$</td>
<td>[Derive]</td>
</tr>
<tr>
<td>STREAM $\cup { \text{odd}(S), \text{even}(S) } = S }$</td>
<td>$\vdash \begin{cases} \text{zip}(\text{odd}(S), \text{even}(S)) = S \end{cases}$</td>
<td>[End]</td>
</tr>
</tbody>
</table>
1 Introduction
   - CC History
   - Behavioral Equivalence, intuitively
   - Behavioral Specifications, intuitively
   - Circular Coinduction, intuitively

2 Circular Coinduction Proof System
   - Formal Framework
   - Coinductive Circularity Principle
   - The Proof System

3 Conclusion
Related Approaches

**Context induction** [R. Hennicker, 1990]
– exploits the inductive definition of the experiments [used also here in CCP]
– requires human guidance, generalization of the induction assertions

**Observational Logic** [M. Bidoit, R. Hennicker, and Al. Kurz, 2002]
– model based (organized as an institution)
– there is a strong similarity between our beh equiv and their infinitary proof system

**Coalgebra**[e.g., J. Adamek 2005, B. Jacobs and J. Rutten 1997] – used to study the states and their operations and their properties
– final coalgebras use to give (behavioral) semantics for processes
– when coalgebra specs are expressed as beh. specs, CC Proof System builds a bisimulation

**Observational proofs by rewriting** [A. Bouhoula and M. Rusinowitch, 2002]
– based on *critical contexts*, which allow to prove or disprove conjectures

**A coinductive calculus of streams** [Jan Rutten, 2005]
– almost all properties proved with CIRC
– extended to *infinite binary trees* [joint work with Al. Silva]
Future Work

Theoretical aspects:
– in some cases the freezing operator is too restrictive ⇒ extend the proof system with new capabilities (special contexts, generalizations, simplifications etc)
– productivity of the behavioral specs vs. well-definedness
– (full) behavioral specification of the non-deterministic processes (behavioral TRS?)
– complexity of the related problems

CIRC Tool:
– automated case analysis
– more case studies (e.g., behavioral semantics of the functors)
– the use of CC as a framework (its use in other applications)
– its use in program verification and analysis
Thanks!