Circular Coinduction
–A Proof Theoretical Foundation–

Grigore Roșu[1] 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, Iași, Romania
dlucanu@info.uaic.ro

08/09/2009, CALCO 2009, Udine

Plan

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)

2005 CC implemented in CoCASL [D. Hausmann& T. Mossakowski & L. Schröder, FASE 2005]

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: $hd(S) = b_1$
  the tail of S: $tl(S) = b_2 : b_3 : \ldots$
- experiments:
  $hd(*{:}Stream), hd(tl(*{:}Stream)), hd(tl(tl(*{:}Stream))), \ldots$
- the basic elements upon on the expriments are built (here $hd(*)$ and $tl(*)$) 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 Π^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_ℓ: $T_\ell(w) = T(Lw)$ for all w
  the right subtree T_r: $T_r(w) = T(Rw)$ for all w
- knowing the root d, T_ℓ and T_r, then T can be written as $d/T_\ell, T_r\backslash$.
- the derivatives: $root(*{:}Tree)$, $left(*{:}Tree)$, and $right(*{:}Tree)$
- the experiments: $root(*{:}Tree)$, $root(left(*{:}Tree))$, $root(right(*{:}Tree))$ 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

  STREAM:

Corecursive spec	Behavioral spec
$zeroes = 0 : zeroes$	$hd(zeroes) = 0$ $tl(zeroes) = zeroes$
$ones = 1 : ones$	$hd(ones) = 1$ $tl(ones) = ones$
$blink = 0 : 1 : blink$	$hd(blink) = 0$ $tl(blink) = 1 : blink$
$zip(B : S, S') = B : zip(S', S)$	$hd(zip(S, S')) = hd(S)$ $tl(S, S') = zip(S', S)$

- for streams, this can be done with STR tool (see H. Zantema's tool paper)

Behavioral Specifications: Intuition 2/2

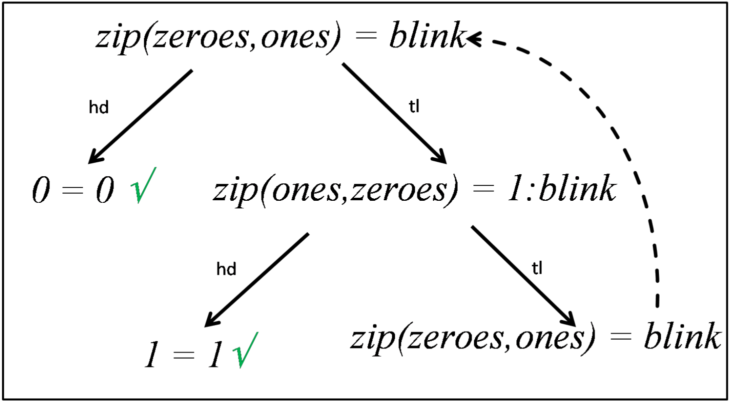
Infinite binary trees (TREE):

- derivatives: $root(*{:}Tree)$, $left(*{:}Tree)$, and $right(*{:}Tree)$
- beh specs are derivative-based specs

Corecursive spec	Behavioral spec
$ones = 1/ones, ones\backslash$	$root(ones) = 1$ $left(ones) = ones$ $right(ones) = ones$
$b/T_\ell, T_r\backslash + b'/T'_\ell, T'_r\backslash =$ $b{\vee}b'/T_\ell{+}T'_\ell, T_r{+}T'_r\backslash$	$root(T + T') = root(T) \vee root(T)$ $left(T + T') = left(T) + left(T')$ $right(T + T') = right(T) + right(T')$
$thue = 0/thue, thue + one\backslash$	$root(thue) = 0$ $left(thue) = thue$ $right(thue) = thue + one$

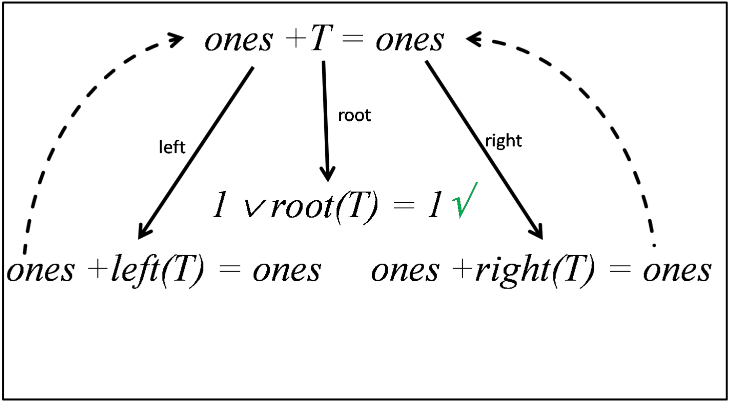
Circular Coinduction: Intuition 1/2

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



Circular Coinduction: Intuition 2/2

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



– a more challenging property: $thue + one = not(thue)$

Plan

Formal Framework 1/2

A behavioral specification consists of:

- a many-sorted algebraic spec $\mathcal{B} = (S, \Sigma, E)$
  (S = set of sorts, Σ = set of opns, E = set of eqns)
- a set of derivatives $\Delta = \{\delta[*{:}h]\}$
  $\delta[*{:}h]$ is a context
  the sort h of the special variable $*$ occuring in a derivative δ is called hidden; the other sorts are called visible
- each derivative can be seen as an equation transformer:
  if e is $t = t'$ `if` *cond*, then $\delta[e]$ is $\delta[t] = \delta[t']$ `if` *cond*
  $\Delta[e] = \{\delta[e] \mid \delta \in \Delta\}$
- an entailment relation $\vdash$, which is reflexive, transitive, monotonic, and Δ-congruent ($E \vdash e$ implies $E \vdash \Delta[e]$)

Formal Framework 2x/2

Experiment:
each visible $\delta[*{:}h] \in \Delta$ is an experiment, and
if $C[*{:}h']$ is an experiment and $\delta[*{:}h] \in \Delta$, then so is $C[\delta[*{:}h]]$

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

Behavioral equivalence of B: $\equiv_{\mathcal{B}} \stackrel{def}{=} \{e \mid \mathcal{B} \Vdash e\}$

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

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` *cond*, then its frozen form is $\boxed{t} = \boxed{t'}$ `if` *cond* ($\boxed{\text{-}} : s \to \mathit{Frozen}$)

– $\vdash$ is extended for frozen equations s.t.
(A1) $E \cup \mathcal{F} \vdash \boxed{e}$ iff $E \vdash e$, for each visible eqn 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 $E \cup C[\mathcal{F}] \vdash C[\mathcal{G}]$

Theorem

(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 $\mathcal{B} \Vdash F$.*

Circular Coinduction Proof System

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

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

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

Soundness

Theorem

(soundness of circular coinduction) *If $\mathcal{B}$ is a behavioral specification and G is a set of equations such that $\mathcal{B} \Vdash^{\circlearrowleft} \boxed{G}$ is derivable using the Circular Coinduction Proof System, then $\mathcal{B} \Vdash G$.*

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 $\mathcal{B}$ is a behavioral specification and G is a set of equations such that $\mathcal{B} \Vdash^{\circlearrowleft} \boxed{G}$ is derivable using the Circular Coinduction Proof System, then $\mathcal{B} \Vdash G$.*

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 $\mathcal{B}$ is a behavioral specification and G is a set of equations such that $\mathcal{B} \Vdash^{\circlearrowleft} \boxed{G}$ is derivable using the Circular Coinduction Proof System, then $\mathcal{B} \Vdash G$.*

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 $\mathcal{B}$ is a behavioral specification and G is a set of equations such that $\mathcal{B} \Vvdash^{\circlearrowleft} \boxed{G}$ is derivable using the Circular Coinduction Proof System, then $\mathcal{B} \Vvdash G$.*

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 $\mathcal{B}$ is a behavioral specification and G is a set of equations such that $\mathcal{B} \Vdash^{\circlearrowleft} \boxed{G}$ is derivable using the Circular Coinduction Proof System, then $\mathcal{B} \Vdash G$.*

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 $\mathcal{B}$ is a behavioral specification and G is a set of equations such that $\mathcal{B} \Vdash^{\circlearrowleft} \boxed{G}$ is derivable using the Circular Coinduction Proof System, then $\mathcal{B} \Vdash G$.*

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

$$\mathtt{STREAM} \cup \left\{ \boxed{\mathtt{zip(odd(S), even(S))}} = \boxed{\mathtt{S}} \right\} \Vdash^{\circlearrowleft} \emptyset$$

$$\mathtt{STREAM} \cup \left\{ \boxed{\mathtt{zip(odd(S), even(S))}} = \boxed{\mathtt{S}} \right\} \vdash \boxed{\mathtt{hd(zip(odd(S), even(S)))}} = \boxed{\mathtt{hd(S)}}$$

[Done]

$$\mathtt{STREAM} \cup \left\{ \boxed{\mathtt{zip(odd(S), even(S))}} = \boxed{\mathtt{S}} \right\} \Vdash^{\circlearrowleft} \left\{ \boxed{\mathtt{hd(zip(odd(S), even(S)))}} = \boxed{\mathtt{hd(S)}} \right\}$$

$$\mathtt{STREAM} \cup \left\{ \boxed{\mathtt{zip(odd(S), even(S))}} = \boxed{\mathtt{S}} \right\} \vdash \boxed{\mathtt{tl(zip(odd(S), even(S)))}} = \boxed{\mathtt{tl(S)}}$$

[Reduce]

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

[Reduce]

$$\mathtt{STREAM} \Vdash^{\circlearrowleft} \left\{ \boxed{\mathtt{zip(odd(S), even(S))}} = \boxed{\mathtt{S}} \right\}$$

[Derive]

Plan

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 $\Vdash$ 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 apsects:

– in some cases the freezing operator is too restrictive $\Rightarrow$ 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!