Circular Coinduction with Special Contexts

Dorel Lucanu[1] Grigore Roșu[2]

[1]Faculty of Computer Science
Alexandru Ioan Cuza University, Iași, Romania
dlucanu@info.uaic.ro

[2]Department of Computer Science
University of Illinois at Urbana-Champaign, USA
grosu@illinois.edu

11/12/2009, ICFEM 2009, Rio de Janeiro

1. Introduction
   - Behavioral Equivalence, intuitively
   - Behavioral Specifications, Intuitively
2. Circular Coinduction
   - Circular Coinduction, intuitively
   - Proof System
3. Special Contexts
   - Intuition
   - Special Hypothesis Defined by Special Contexts
   - Extended Proof System
   - Implementation in CIRC
4. Conclusion

Plan

1. Introduction
   - Behavioral Equivalence, intuitively
   - Behavioral Specifications, Intuitively
2. Circular Coinduction
   - Circular Coinduction, intuitively
   - Proof System
3. Special Contexts
   - Intuition
   - Special Hypothesis Defined by Special Contexts
   - Extended Proof System
   - Implementation in CIRC
4. Conclusion

Behavioral Equivalence: Intuition 1/2

Behavioral equivalence is the non-distinguishability under experiments

Example of streams:

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

Behavioral Equivalence: Intuition 1/2

Behavioral equivalence is the non-distinguishability under experiments

Example of streams:

- experiments:
  $hd(*{:}Stream), hd(tl(*{:}Stream)), hd(tl(tl(*{:}Stream))), \ldots$
- if $hd(S) = b_1$,

Behavioral Equivalence: Intuition 1/2

Behavioral equivalence is the non-distinguishability under experiments

Example of streams:

- experiments:
  $hd(*{:}Stream), hd(tl(*{:}Stream)), hd(tl(tl(*{:}Stream))), \ldots$
- if $hd(S) = b_1$, $hd(tl(S)) = b_2$,

Behavioral Equivalence: Intuition 1/2

Behavioral equivalence is the non-distinguishability under experiments

Example of streams:

- experiments:
  $hd(*{:}Stream), hd(tl(*{:}Stream)), hd(tl(tl(*{:}Stream))), \ldots$
- if $hd(S) = b_1$, $hd(tl(S)) = b_2$, $hd(tl(tl(S))) = b_3$, $\ldots$

Behavioral Equivalence: Intuition 1/2

Behavioral equivalence is the non-distinguishability under experiments

Example of streams:

- experiments:
  $hd(*{:}Stream), hd(tl(*{:}Stream)), hd(tl(tl(*{:}Stream))), \ldots$
- if $hd(S) = b_1$, $hd(tl(S)) = b_2$, $hd(tl(tl(S))) = b_3$, $\ldots$
  then the stream S is $b_1 : b_2 : b_3 : \ldots$

Behavioral Equivalence: Intuition 1/2

Behavioral equivalence is the non-distinguishability under experiments

Example of streams:

- experiments:
  $hd(*{:}Stream), hd(tl(*{:}Stream)), hd(tl(tl(*{:}Stream))), \ldots$
- if $hd(S) = b_1$, $hd(tl(S)) = b_2$, $hd(tl(tl(S))) = b_3$, $\ldots$
  then the stream S is $b_1 : b_2 : b_3 : \ldots$
- two streams S and S' are behavioral equivalent ($S \equiv S'$) iff
  $C[S] = C[S']$ for each experiment C
- showing beh. equiv. is Π^0_2-hard (S. Buss, G. Roşu, 2000, 2006)

Behavioral Equivalence: Intuition 2/2

Example of processes:

experiments: $\checkmark?(*\{a_1\}\ldots\{a_n\}) :\equiv$ is $a_1 \ldots a_n$ a successful evolution?

Behavioral Equivalence: Intuition 2/2

Example of processes:

experiments: $\checkmark?(*\{a_1\}\ldots\{a_n\}) :\equiv$ is $a_1 \ldots a_n$ a successful evolution?

$$\checkmark?(p\{a\}) = \mathit{true} \qquad\qquad p \xrightarrow{a} \checkmark$$

Behavioral Equivalence: Intuition 2/2

Example of processes:

experiments: $\checkmark?(*\{a_1\}\ldots\{a_n\}) :\equiv$ is $a_1 \ldots a_n$ a successful evolution?

$$\checkmark?(p\{a\}) = true \qquad p \xrightarrow{a} \checkmark$$

$$\checkmark?(p\{a\}\{b\}) = true \qquad p \xrightarrow{a} \bullet \xrightarrow{b} \checkmark$$

Behavioral Equivalence: Intuition 2/2

Example of processes:

experiments: $\checkmark?(*\{a_1\}\ldots\{a_n\}) :\equiv$ is $a_1 \ldots a_n$ a successful evolution?

$$\checkmark?(p\{a\}) = true \qquad p \xrightarrow{a} \checkmark$$

$$\checkmark?(p\{a\}\{b\}) = true \qquad p \xrightarrow{a} \bullet \xrightarrow{b} \checkmark$$

$$\checkmark?(p\{a\}\{a\}\{a\}) = true \qquad p \xrightarrow{a} \bullet \xrightarrow{a} \bullet \xrightarrow{a} \checkmark$$

$\ldots$

Behavioral Equivalence: Intuition 2/2

Example of processes:

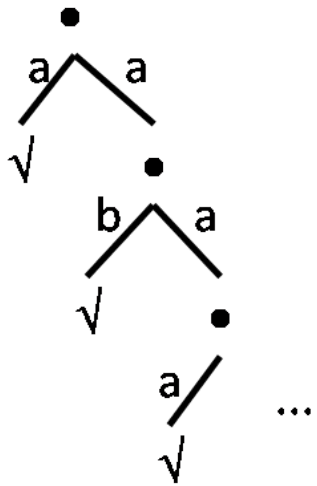
experiments: $\checkmark?(*\{a_1\}\ldots\{a_n\}) :\equiv$ is $a_1 \ldots a_n$ a successful evolution?

$$\checkmark?(p\{a\}) = true \qquad p \xrightarrow{a} \checkmark$$

$$\checkmark?(p\{a\}\{b\}) = true \qquad p \xrightarrow{a} \bullet \xrightarrow{b} \checkmark$$

$$\checkmark?(p\{a\}\{a\}\{a\}) = true \qquad p \xrightarrow{a} \bullet \xrightarrow{a} \bullet \xrightarrow{a} \checkmark$$

$\ldots$



for this case, behavioral equivalence coincides with complete trace equivalence

Behavioral Specifications: Intuition 1/2

Streams (STREAM):

- derivatives: $hd(*{:}Stream)$ and $tl(*{:}Stream)$
- behavioral specifications are derivative-based specifications

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

Behavioral Specifications: Intuition 2/2

Processes (BPA):

- derivatives: $\checkmark?(*{:}Proc), *{:}Proc\{a\}, *{:}Proc\{b\}, \ldots$
- beh specs are derivative-based specs

Corecursive spec	Behavioral spec
$X = a + a\,;a\,;X$	$a\{a\} = \checkmark$ $(p+q)\{a\} = p\{a\} + q\{a\}$ $(p\,;q)\{a\} = p\{a\}\,;q \quad \texttt{if} \quad p \neq \checkmark$ $\checkmark?(\checkmark) = true$ $\checkmark?(p+q) = \checkmark?(p) \vee \checkmark?(q)$ $\ldots$

Plan

1. Introduction
   - Behavioral Equivalence, intuitively
   - Behavioral Specifications, Intuitively
2. Circular Coinduction
   - Circular Coinduction, intuitively
   - Proof System
3. Special Contexts
   - Intuition
   - Special Hypothesis Defined by Special Contexts
   - Extended Proof System
   - Implementation in CIRC
4. Conclusion

Circular Coinduction: Intuition

1. $hd(odd(S)) = hd(S)$
2. $tl(odd(S)) = even(tl(S))$
3. $even(S) = odd(tl(S))$
4. $hd(zip(S_1, S_2)) = hd(S_1)$
5. $tl(zip(S_1, S_2)) = zip(S_2, tl(S_1))$

Circular Coinduction: Intuition

1. $hd(odd(S)) = hd(S)$
2. $tl(odd(S)) = even(tl(S))$
3. $even(S) = odd(tl(S))$
4. $hd(zip(S_1, S_2)) = hd(S_1)$
5. $tl(zip(S_1, S_2)) = zip(S_2, tl(S_1))$

$$zip(odd(S), even(S)) = S$$

Circular Coinduction: Intuition

1. $hd(odd(S)) = hd(S)$
2. $tl(odd(S)) = even(tl(S))$
3. $even(S) = odd(tl(S))$
4. $hd(zip(S_1,S_2)) = hd(S_1)$
5. $tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))$

$$zip(odd(S), even(S)) = S$$

$$\downarrow 3$$

$$zip(odd(S), odd(tl(S))) = S$$

Circular Coinduction: Intuition

1. $hd(odd(S)) = hd(S)$
2. $tl(odd(S)) = even(tl(S))$
3. $even(S) = odd(tl(S))$
4. $hd(zip(S_1,S_2)) = hd(S_1)$
5. $tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))$

6. $zip(odd(S), odd(tl(S))) = S$

$$zip(odd(S), even(S)) = S$$

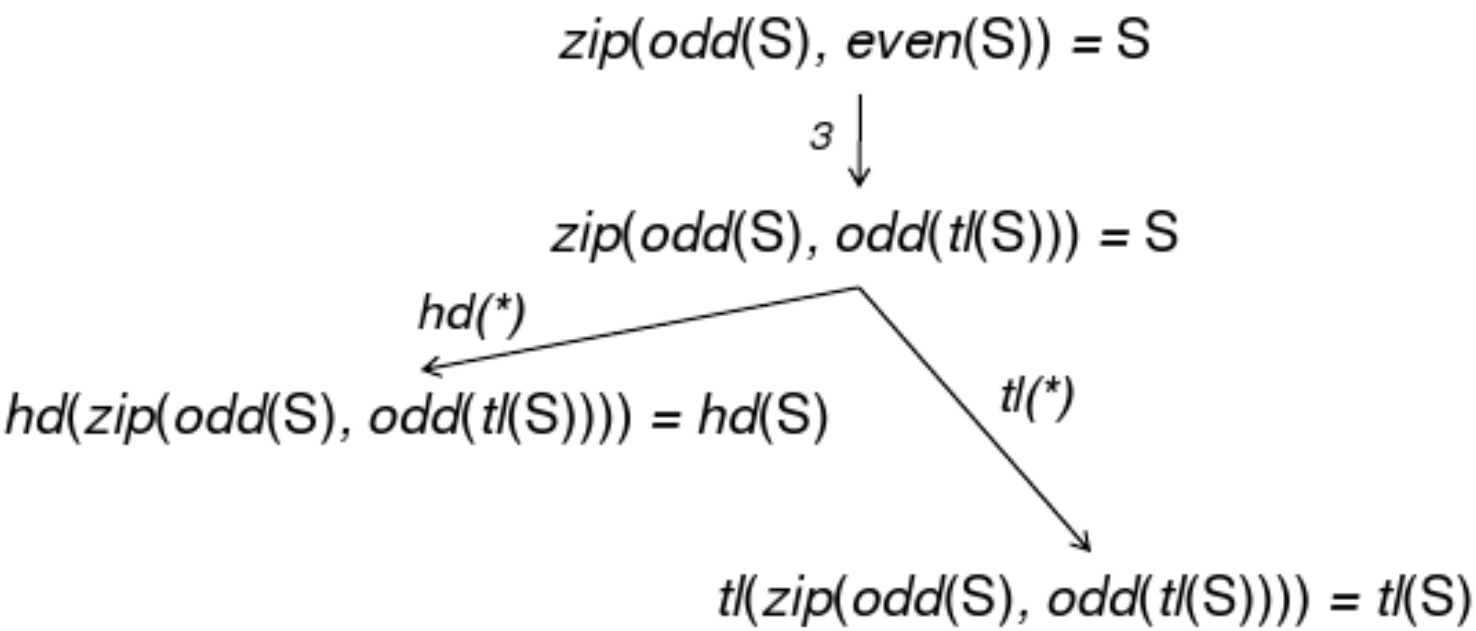
$$\downarrow 3$$

$$zip(odd(S), odd(tl(S))) = S$$

Circular Coinduction: Intuition

1. $hd(odd(S)) = hd(S)$
2. $tl(odd(S)) = even(tl(S))$
3. $even(S) = odd(tl(S))$
4. $hd(zip(S_1,S_2)) = hd(S_1)$
5. $tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))$

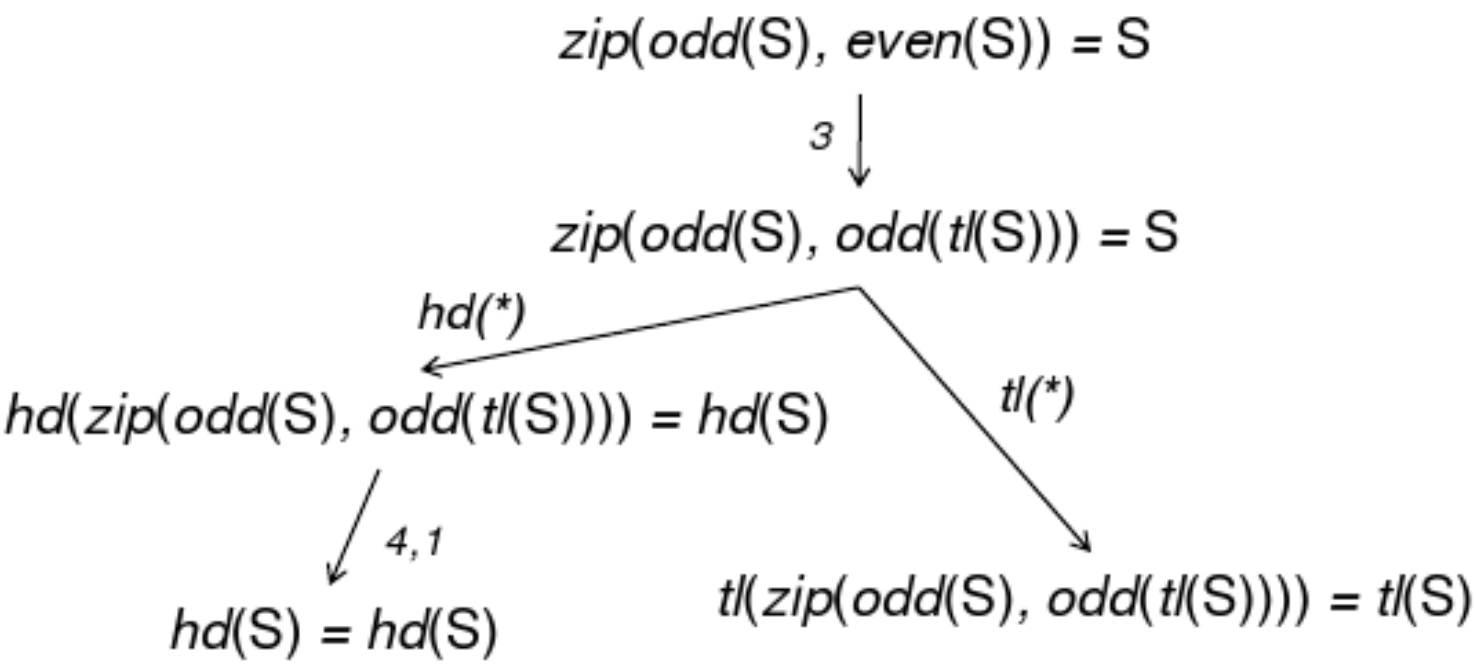
6. $zip(odd(S), odd(tl(S))) = S$



Circular Coinduction: Intuition

1. $hd(odd(S)) = hd(S)$
2. $tl(odd(S)) = even(tl(S))$
3. $even(S) = odd(tl(S))$
4. $hd(zip(S_1,S_2)) = hd(S_1)$
5. $tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))$

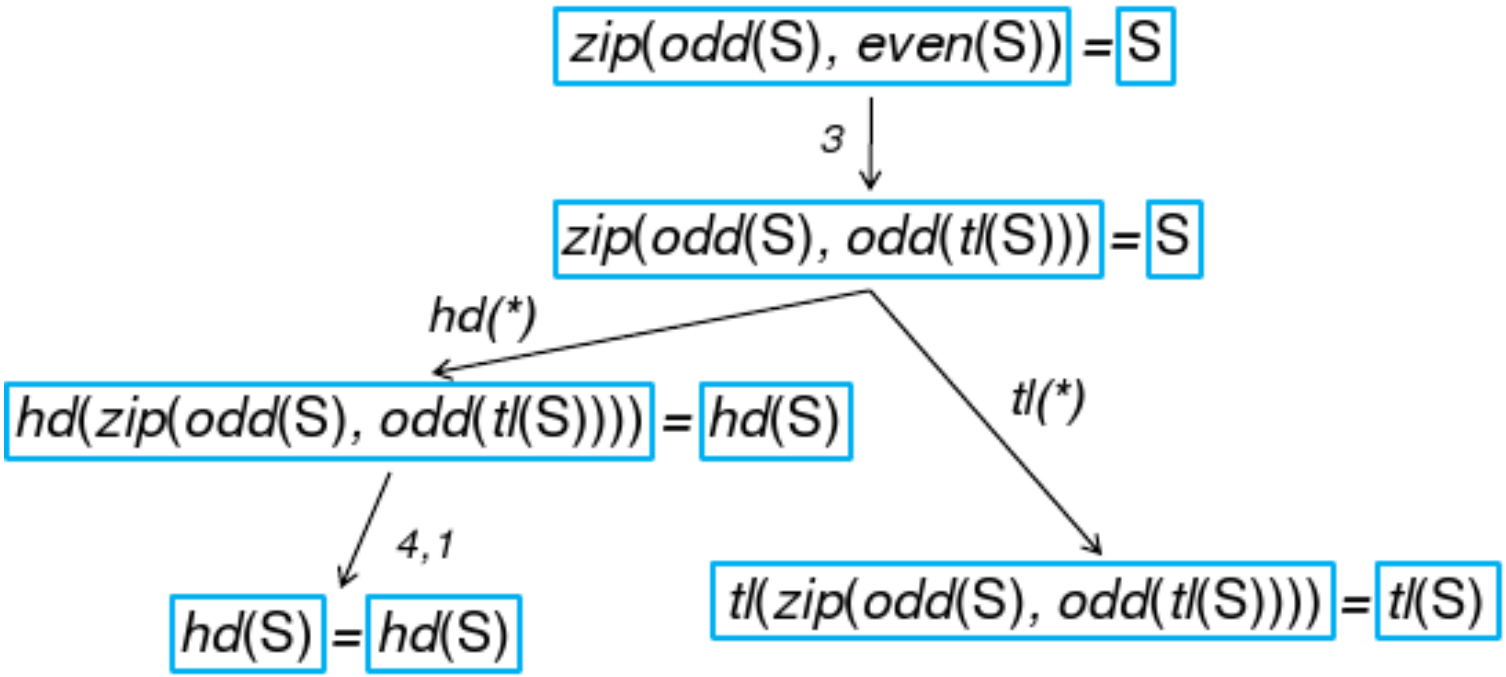
6. $zip(odd(S), odd(tl(S))) = S$



Circular Coinduction: Intuition

1. $hd(odd(S)) = hd(S)$
2. $tl(odd(S)) = even(tl(S))$
3. $even(S) = odd(tl(S))$
4. $hd(zip(S_1,S_2)) = hd(S_1)$
5. $tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))$

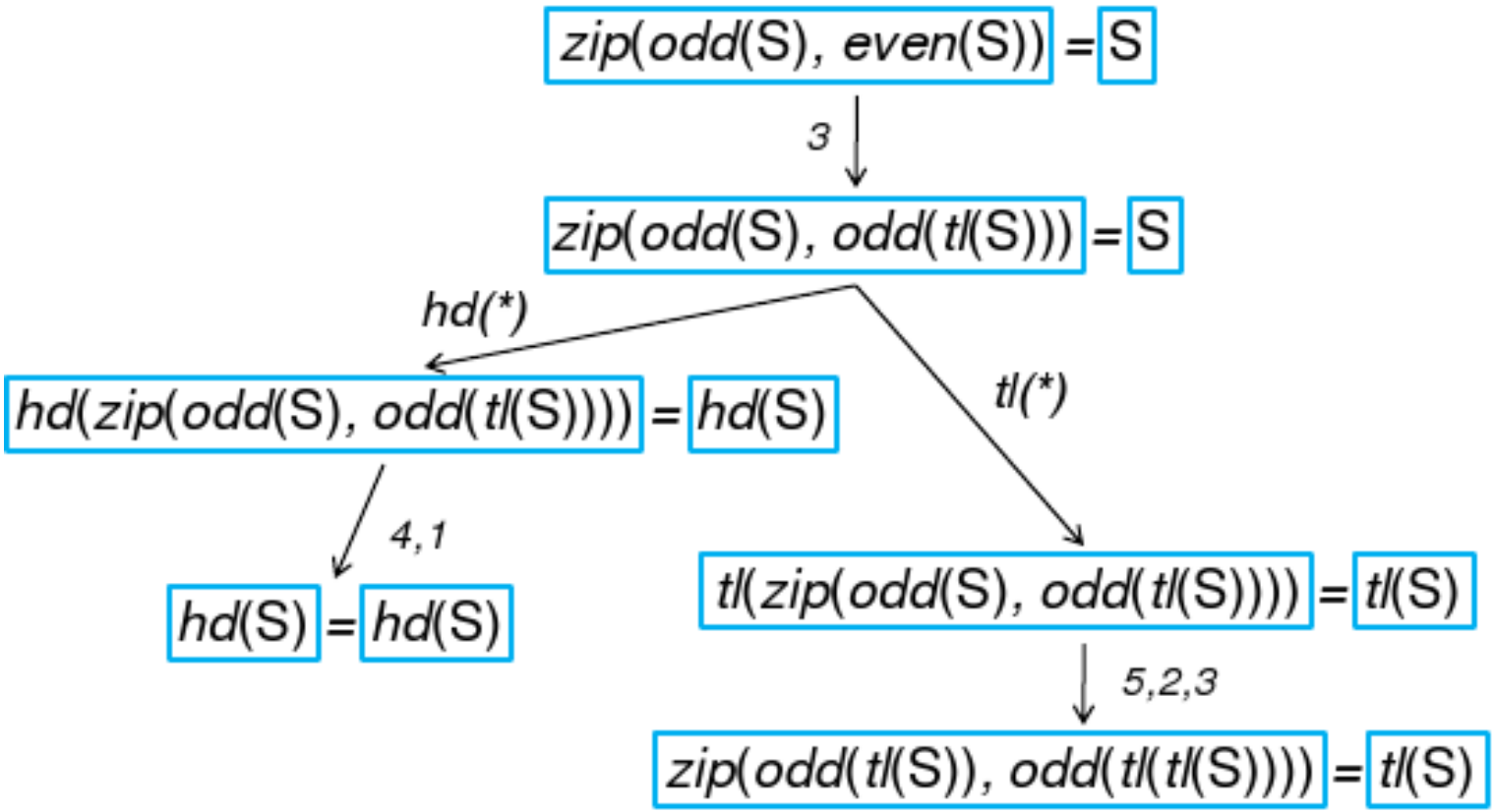
6. $zip(odd(S), odd(tl(S))) = S$



Circular Coinduction: Intuition

1. $hd(odd(S)) = hd(S)$
2. $tl(odd(S)) = even(tl(S))$
3. $even(S) = odd(tl(S))$
4. $hd(zip(S_1,S_2)) = hd(S_1)$
5. $tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))$

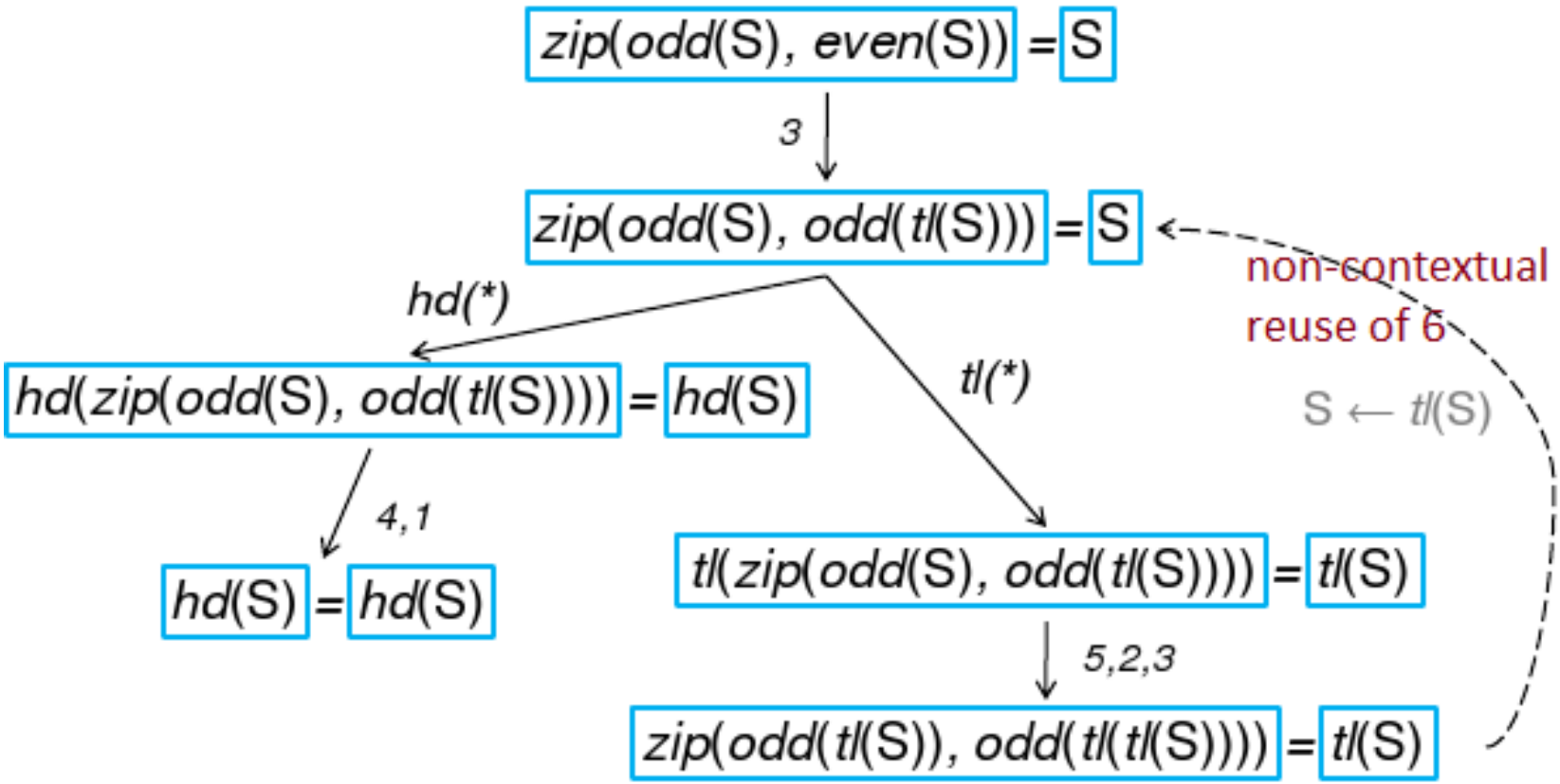
6. $zip(odd(S), odd(tl(S))) = S$



Circular Coinduction: Intuition

1. $hd(odd(S)) = hd(S)$
2. $tl(odd(S)) = even(tl(S))$
3. $even(S) = odd(tl(S))$
4. $hd(zip(S_1,S_2)) = hd(S_1)$
5. $tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))$

6. $zip(odd(S), odd(tl(S))) = S$



Circular Coinduction Proof System

$\mathcal{B}$ a many-sorted algebraic specification (S, Σ, E)
Δ a set of derivatives
$\mathcal{F}$ a set of frozen hypotheses $\boxed{e} ::= \boxed{t} = \boxed{t'}$ if *cond*
$\mathcal{G}$ a set of goals, which are frozen equations
$\vdash$ an entailment relation $\vdash$ between $\mathcal{B}$ and equations

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

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

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

Circular Coinduction Proof System Explained

- the rule [Derive]] is strongly related to induction on contexts ([Hennicker, Bidoit, Kurz]): in order to prove e, assume $C[e]$ for an arbitrary but fixed context C and prove $C[\delta[e]]$ for any derivative δ
- the freezing relieves the user of our proof system from performing explicit induction on contexts;
  - the user of our proof system needs not be aware of any contexts at all (except for the derivatives), nor of induction on contexts
- the frozen 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

$$\frac{\ldots t_i \ldots = \ldots t_i' \ldots}{f(\ldots t_i \ldots) = f(\ldots t_i' \ldots)}$$

- the other rules of equational deduction are sound in combination with the accumulated hypotheses in $\mathcal{F}$, including substitution and transitivity

Example

$$\frac{\mathtt{STREAM} \cup \left\{ \boxed{\mathtt{zip(odd(S), even(S))}} = \boxed{\mathtt{S}} \right\} \quad \Vdash^{\circlearrowleft} \emptyset}{\mathtt{STREAM} \cup \left\{ \boxed{\mathtt{zip(odd(S), even(S))}} = \boxed{\mathtt{S}} \right\} \quad \vdash \boxed{\mathtt{hd(zip(odd(S), even(S)))}} = \boxed{\mathtt{hd(S)}}} \text{ [Done]}$$

$$\frac{\mathtt{STREAM} \cup \left\{ \boxed{\mathtt{zip(odd(S), even(S))}} = \boxed{\mathtt{S}} \right\} \quad \Vdash^{\circlearrowleft} \left\{ \boxed{\mathtt{hd(zip(odd(S), even(S)))}} = \boxed{\mathtt{hd(S)}} \right\}}{\mathtt{STREAM} \cup \left\{ \boxed{\mathtt{zip(odd(S), even(S))}} = \boxed{\mathtt{S}} \right\} \quad \vdash \boxed{\mathtt{tl(zip(odd(S), even(S)))}} = \boxed{\mathtt{tl(S)}}} \text{ [Reduce]}$$

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

$$\frac{\mathtt{STREAM} \quad \Vdash^{\circlearrowleft} \left\{ \boxed{\mathtt{zip(odd(S), even(S))}} = \boxed{\mathtt{S}} \right\}}{} \text{ [Derive]}$$

Plan

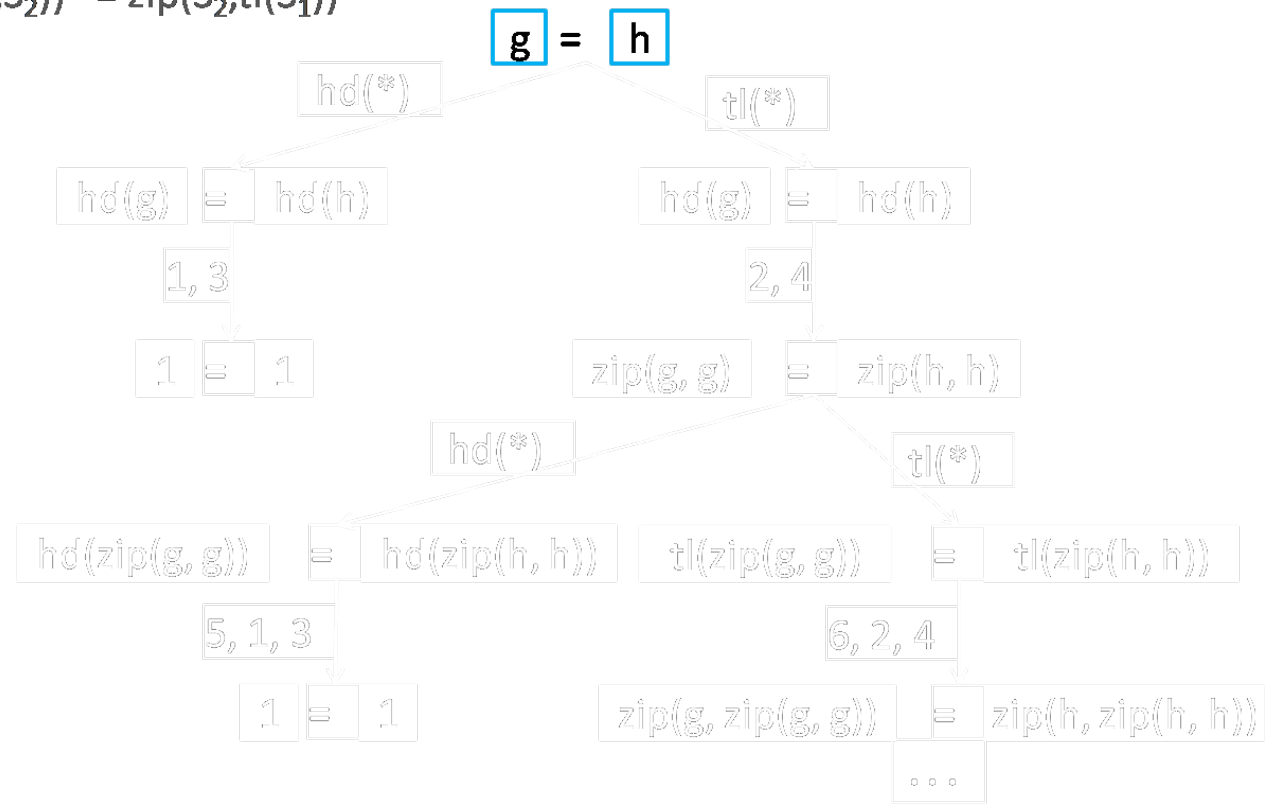
1. Introduction
   - Behavioral Equivalence, intuitively
   - Behavioral Specifications, Intuitively
2. Circular Coinduction
   - Circular Coinduction, intuitively
   - Proof System
3. Special Contexts
   - Intuition
   - Special Hypothesis Defined by Special Contexts
   - Extended Proof System
   - Implementation in CIRC
4. Conclusion

Special Contexts: Intuition

1. hd(g) = 1
2. tl(g) = zip(g, g)
3. hd(h) = 1
4. tl(h) = zip(h, h)
5. hd(zip(S_1,S_2)) = hd(S_1)
6. tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))

7. g = h

8. zip(g, g) = zip(h, h)

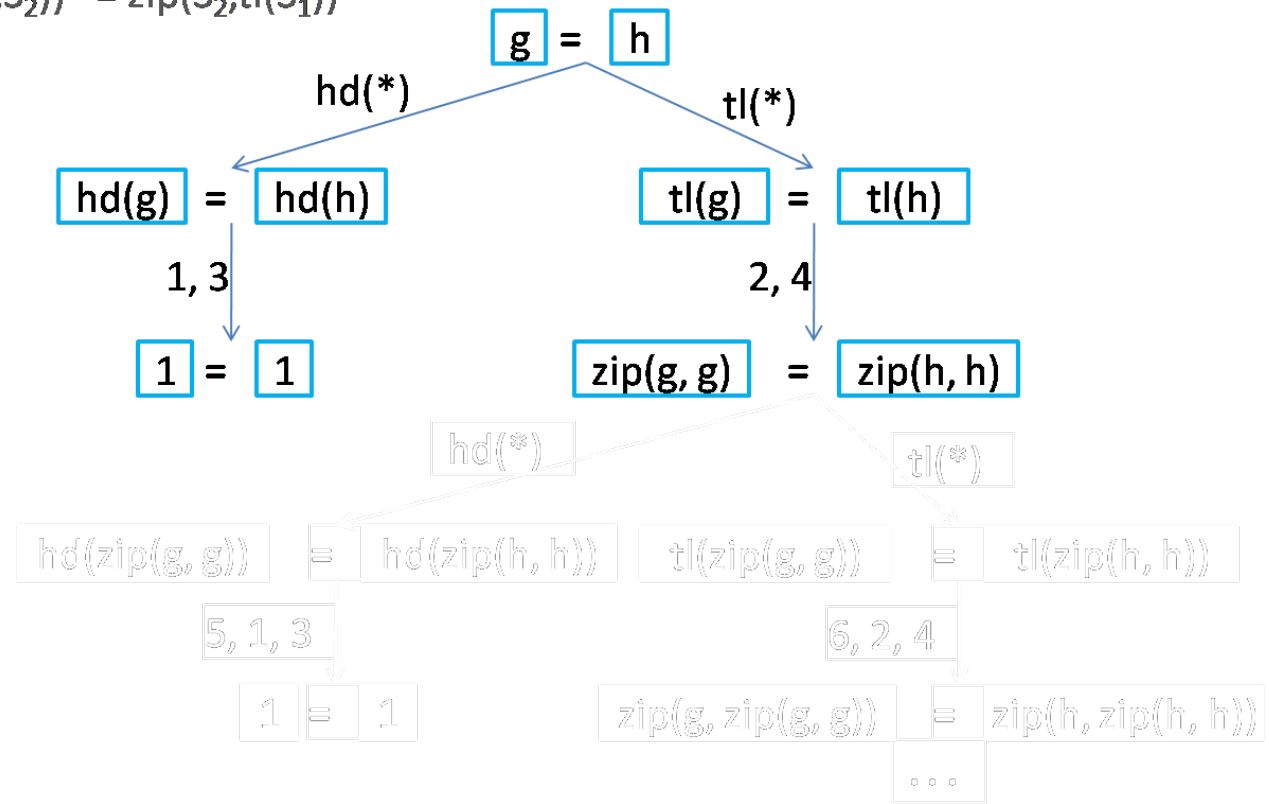


Special Contexts: Intuition

1. hd(g) = 1
2. tl(g) = zip(g, g)
3. hd(h) = 1
4. tl(h) = zip(h, h)
5. hd(zip(S_1,S_2)) = hd(S_1)
6. tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))

7. g = h

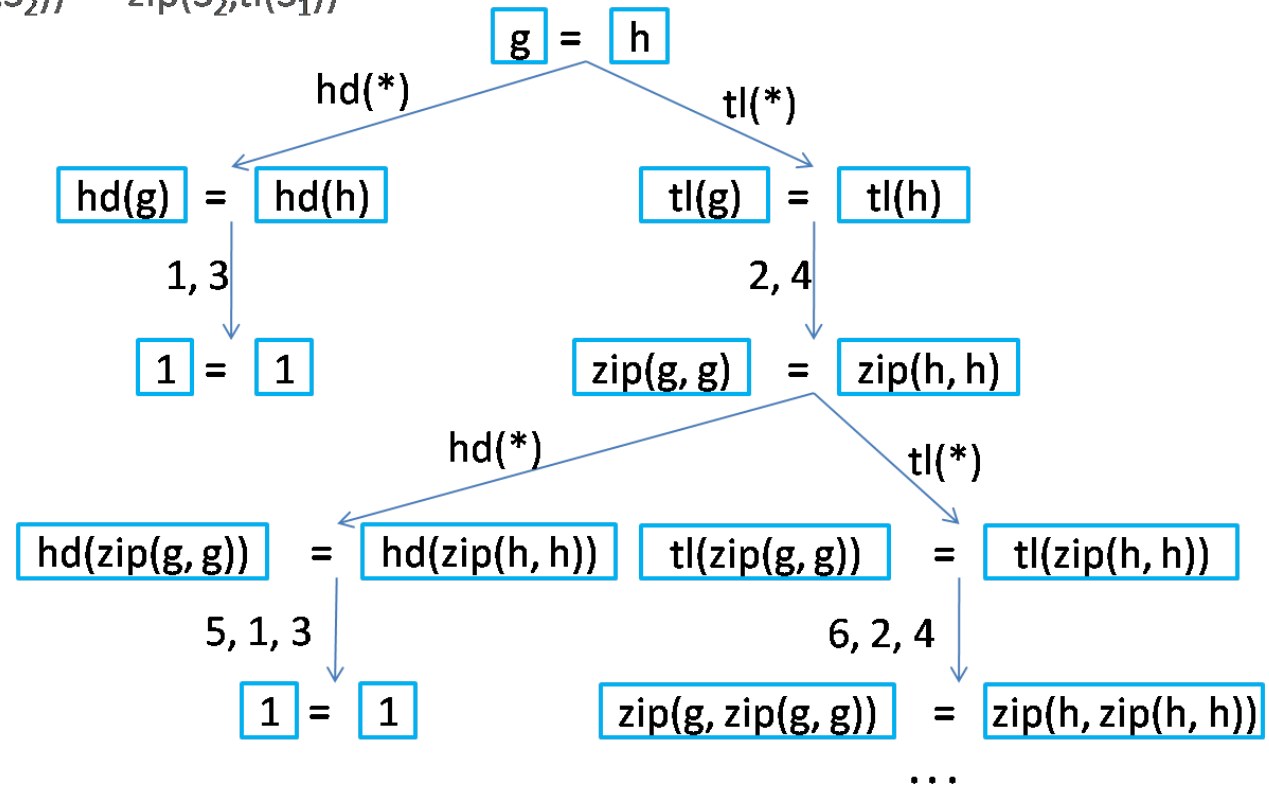
8. zip(g, g) = zip(h, h)



Special Contexts: Intuition

1. hd(g) = 1
2. tl(g) = zip(g, g)
3. hd(h) = 1
4. tl(h) = zip(h, h)
5. hd(zip(S_1,S_2)) = hd(S_1)
6. tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))

7. g = h
8. zip(g, g) = zip(h, h)



Special Contexts: Intuition

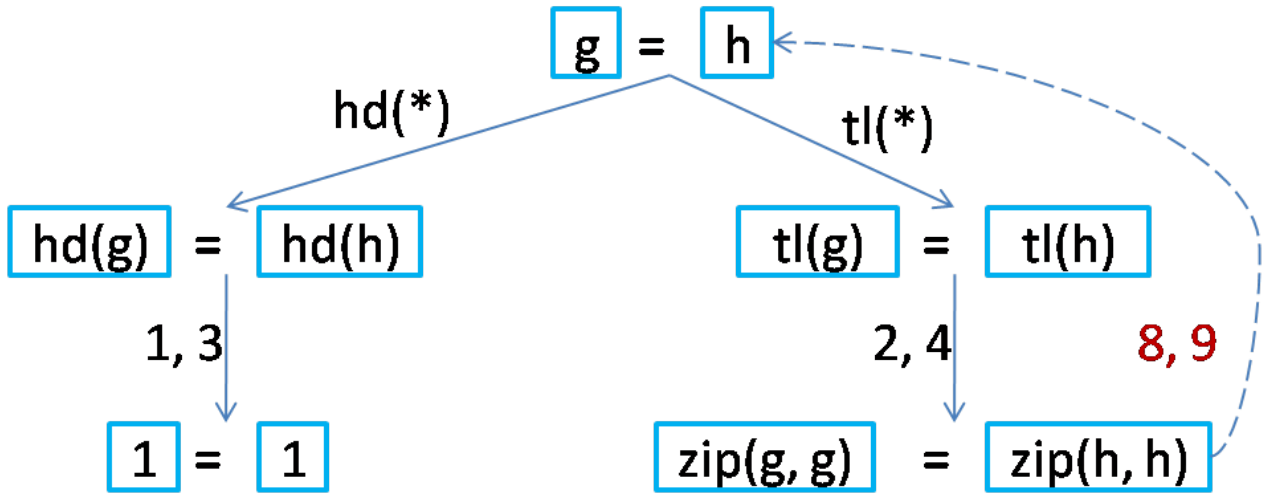
1. hd(g) = 1
2. tl(g) = zip(g, g)
3. hd(h) = 1
4. tl(h) = zip(h, h)
5. hd(zip(S_1,S_2)) = hd(S_1)
6. tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))

7. g = h
8. zip(g, g) = zip(g, h)
9. zip(g, h) = zip(h, h)

special hypotheses

zip(*,S)
zip(S,*)

special contexts



Special Contexts: Counter-example

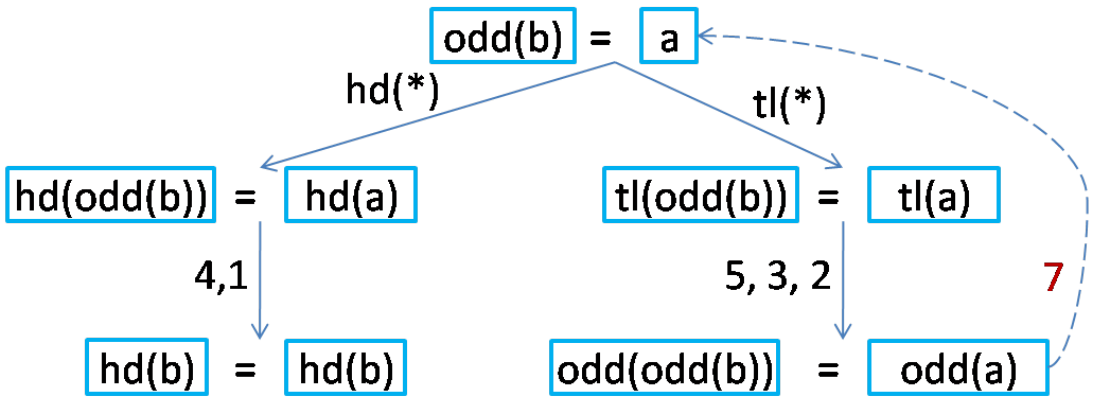
1. hd(a) = hd(b)
2. tl(a) = odd(a)
3. tl(tl(b)) = odd(b)
4. hd(odd(S)) = hd(S)
5. tl(odd(S)) = odd(tl(tl(S))

6. odd(b) = a
7. odd(odd(b)) = odd(a)

special hypotheses

odd(*)

special contexts



Counter-example: $a = 0 : 0 : 1 : 2^{\infty}$ and $b = 0 : 1 : 0^{\infty}$

Special Hypotheses

- however, the contextual reasoning with the frozen hypotheses is needed ... but it is not always sound
- our solution: replace the congruence rule

$$\frac{\ldots \boxed{t_i} \ldots = \ldots \boxed{t_i'} \ldots}{\boxed{f(\ldots t_i \ldots)} = \boxed{f(\ldots t_i' \ldots)}}$$

with a set of special hypotheses: whenever the use of

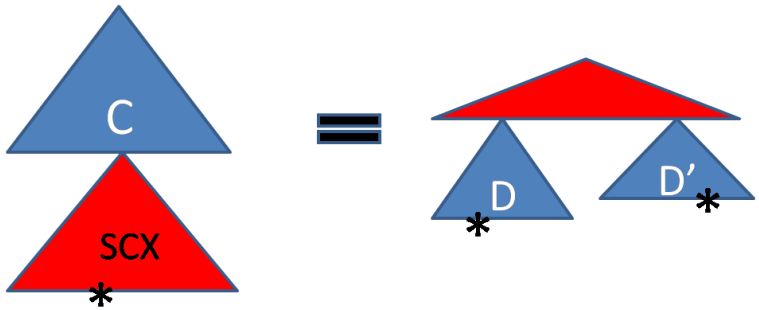
$$\boxed{f(\ldots t_i \ldots)} = \boxed{f(\ldots t_i' \ldots)}$$

is sound, add it to the set $\mathcal{F}$ of frozen hypotheses

- the special hypotheses can be obtained for free: if we know that $f(\ldots * \ldots)$ is safe (special), then add to $\mathcal{F}$ simultaneously $\boxed{t_i} = \boxed{t_i'}$ and $\boxed{f(\ldots t_i \ldots)} = \boxed{f(\ldots t_i' \ldots)}$

Special Contexts

A context $SCX[*{:}h]$ is special iff for any experiment C for SCX there is some term t such that $\mathcal{B} \vdash C[SCX[*{:}h]] = t$ and each occurrence of $*{:}h$ in t appears only in a subterm of depth $\leq$ depth of C



Theorem

If F is a hidden equation set and SCX a special context, $SCX[F]$ is a set of special hypotheses.

Extended Circular Coinduction Proof System

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

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

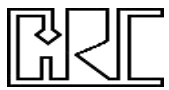
$$\frac{\mathcal{B} \cup \mathcal{F} \cup \{\boxed{e}\} \cup \boxed{\Gamma[e]} \Vdash^{\circlearrowleft} \mathcal{G} \cup \boxed{\Delta[e]}}{\mathcal{B} \cup \mathcal{F} \Vdash^{\circlearrowleft} \mathcal{G} \cup \{\boxed{e}\}} \text{[Derive}^{\text{scx}}\text{]}$$

where Γ is a given set of special contexts

$\Rightarrow$ The special frozen hypotheses are added on-the-fly!

How can we find such a Γ?

$\Rightarrow$ CIRC tool provides an algorithm computing a Γ



CIRC

- joint work Al. I. Cuza Univ. of Iasi (UAIC, RO) and Univ. of Illinois at Urbana-Champaign (UIUC, US)
- CIRC implements circular coinduction and circular induction completely automated
- CIRC is developed in Maude at metalevel using the reflection of rewriting logic
- CIRC can be seen as an extension of Maude with behavioral ingredients (and not only)
- the proof tactics are described using a specific rewriting strategy language
- the coinduction and induction can be combined in an interactive way
- study cases: streams, infinite binary trees, processes, regular expressions, automata described by functorial functors, ...

Special Contexts in CIRC

- CIRC includes facilities for automatically using of the special hypotheses defined by the special contexts
- CIRC includes an algorithm computing a set of special contexts for a given theory
  - the algorithm tries to find a maximal set of minimal special contexts $f(\ldots * \ldots)$, which is closed under composition (the composition of two special contexts is a special context)
- there are no guaranties that the algorithm find all the special contexts (see Conclusion for why not)
- the algorithm showed to be efficient in many practical cases

DEMO

Plan

1. Introduction
   - Behavioral Equivalence, intuitively
   - Behavioral Specifications, Intuitively
2. Circular Coinduction
   - Circular Coinduction, intuitively
   - Proof System
3. Special Contexts
   - Intuition
   - Special Hypothesis Defined by Special Contexts
   - Extended Proof System
   - Implementation in CIRC
4. Conclusion

Conclusion

Achievements:

- circular coinduction is a simple an powerful proof method by coinduction
- special hypotheses defined by the special contexts significantly improves the circular coinduction proof systems
- the automated computation of the special contexts is more than necesarily in practice
- CIRC implementation offers means to automatically compute and use the special contexts

Future and in progress work:

- Special Contexts Problem is Π^0_2-complete (unpublished)
- Extension of the algorithm with case-analysis (unpublished)
- Correctness of the algorithm (unpublished)
- to use CIRC in the verification of OO programs (in progress)

Thanks!