CIRC prover: an overview

Dorel Lucanu

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

2nd Romanian-Japanesse on Algebraic Specifications Workshop,
Sinaia, 1/3/2011

Plan

Circular Coinduction and CIRC

- joint work Al. I. Cuza Univ. of Iasi (UAIC, RO) and Univ. of Illinois at Urbana-Champaign (UIUC, US)
- Theoretical achievments:
  - Circular Coinduction (CC) proof system
  - extensions with special contexts and equational interpolants (generalization, case analysis, inductive definition of the basic entailment relation)
- Implementation
  - CIRC implements circular coinduction 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
  - the proof tactics are given using a specific rewriting strategy language
  - study cases: streams, infinite binary trees, processes, regular expressions, automata described by functorial functors, . . .

Behavioral Specifications

- algebraic specification $\mathcal{E} = (S, \Sigma, E)$, where S is a set of sorts, Σ a S-signature, E a set of (conditional) equations
- a Σ-context C is a Σ-term with one occurrence of a distinguished variable $*{:}s$ of sort s
- contexts as equation transformers: if e is $(\forall X)\, t = t'$ `if` $cond$, then $C[e]$ denotes $(\forall X \cup Y)\, C[t] = C[t']$ `if` $cond$
- behavioral specification $\mathcal{B} = (S, (\Sigma, \Delta), E)$, where Δ is a set of Σ-contexts
  - hidden sorts: $H = \{h \mid \delta[*{:}h] \in \Delta\}$, and
  - visible sorts: $V = S \setminus H$
- experiment = a Δ-context of visible sort

Behavioral Equivalence

- contextual entailment system: an entailemnt relation $\vdash$ satisfying reflexivity, monotonicity, transitivity, and Δ-congruence ($E \vdash e$ implies $E \vdash \delta[e]$ for each $\delta \in \Delta$)
- we write $\mathcal{B} \vdash e$ for $E \vdash e$, where $\mathcal{B} = (S, (\Sigma, \Delta), E)$
- behavioral entailment: $\mathcal{B} \Vdash e$ iff $\mathcal{B} \vdash C[e]$ for each Δ-experiment C appropriate for the equation e
- behavioral equivalence: $\equiv = \{e \mid \mathcal{B} \Vdash e\}$

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 $b_1 : b_2 : b_3 : \ldots$
- showing beh. equiv. is Π^0_2-hard (S. Buss, G. Roșu, 2000, 2006)

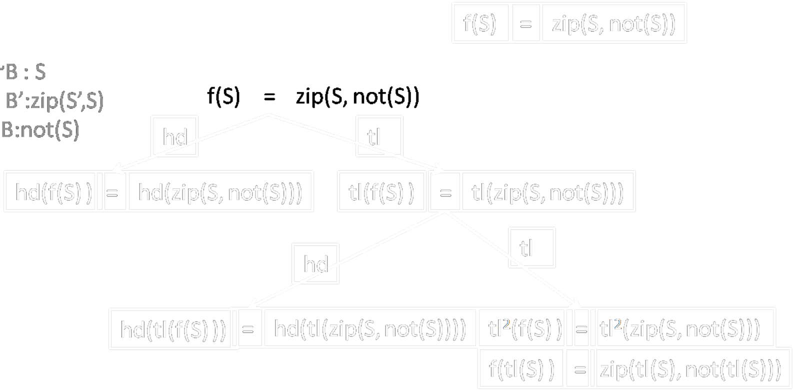
Behavioral Specifications: Maude like syntax

```
(theory STREAM is
  sort Bit .
  ops 0 1 : -> Bit .

  sort Stream .
  op hd : Stream -> Bit .
  op tl : Stream -> Stream .

  op not : Stream -> Stream .
  eq hd(not(S)) = ~ hd(S) .
  eq tl(not(S)) = not(tl(S)) .

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

  op ~_ : Bit -> Bit .
  eq ~ 0 = 1 .
  eq ~ 1 = 0 .
  var S, S' : Stream .
  derivative hd(*:Stream) .
  derivative tl(*:Stream) .

  op f : Stream -> Stream .
  eq hd(f(S)) = hd(S) .
  eq hd(tl(S)) = ~ hd(S) .
  eq tl(tl(S)) = f(tl(S)) .
endtheory)
```

Plan

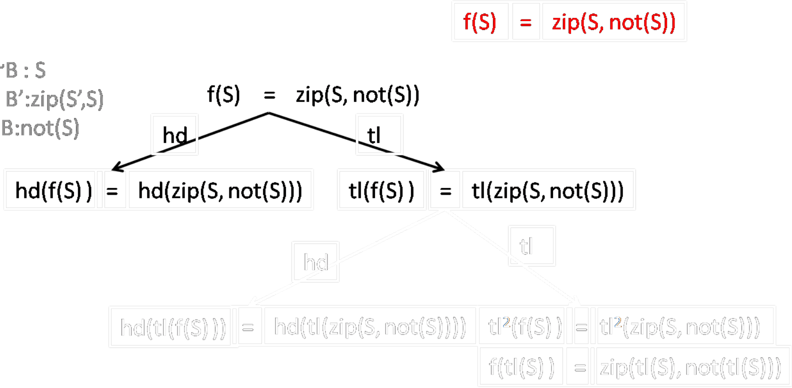
Circular Coinduction: Intuition

f(B:S) = B : ~B : S
zip(B:S,S') = B':zip(S',S)
not(B:S) = ~B:not(S)



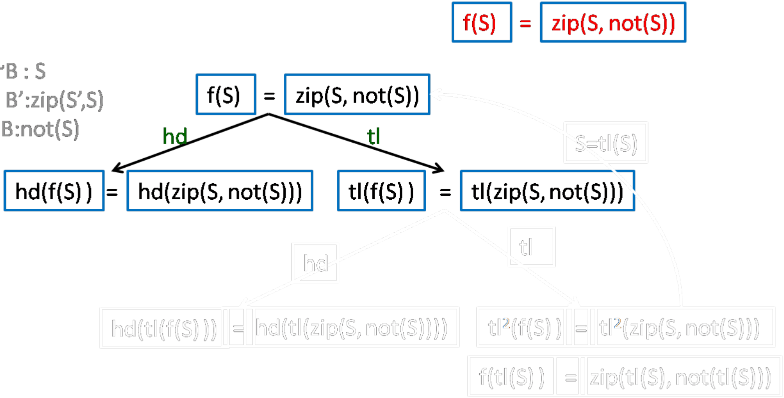
Circular Coinduction: Intuition

f(B:S) = B : ~B : S
zip(B:S,S') = B':zip(S',S)
not(B:S) = ~B:not(S)

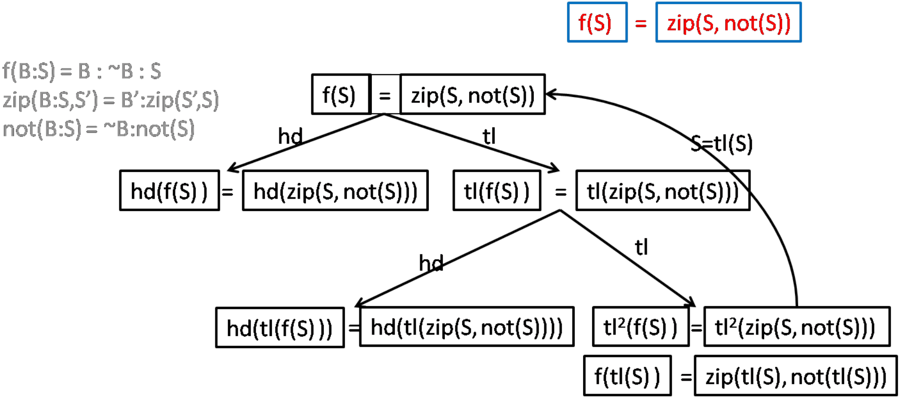


Circular Coinduction: Intuition

f(B:S) = B : ~B : S
zip(B:S,S') = B':zip(S',S)
not(B:S) = ~B:not(S)



Circular Coinduction: Intuition



Circular Coinduction Proof System

(Roșu & Lucanu, CALCO 2009)

$\mathcal{B}$ a behavioral 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

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

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

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

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

$$\frac{\ldots \boxed{t_i} \ldots = \ldots \boxed{t'_i} \ldots}{\boxed{f(\ldots t_i \ldots)} = \boxed{f(\ldots t'_i \ldots)}}$$

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

CC in CIRC 1/2

– CIRC commands

```
(add goal f(S:Stream) = zip(S:Stream,not(S:Stream)) .)
(coinduction .)
```

– Here is the ouput for

STREAM $\Vdash$ f(S:Stream) = zip(S:Stream,not(S:Stream))

the commands used: `(add goal ... .)` and `(coinduction .)`

```
Goal added: f(S:Stream) = zip(S:Stream,not(S:Stream))

Proof succeeded.
  Number of derived goals: 4
  Number of proving steps performed: 22
  Maximum number of proving steps is set to: 256

Proved properties:

  tl(f(S:Stream)) = zip(not(S:Stream),tl(S:Stream))
  f(S:Stream) = zip(S:Stream,not(S:Stream))
```

CC in CIRC 2/2 ("show proof" command)

```
. . .

1. |||-  [* hd(tl(f(S:Stream))) *] = [* hd(zip(not(S:Stream),tl(S:Stream)))
2. |||-  [* tl(tl(f(S:Stream))) *] = [* tl(zip(not(S:Stream),tl(S:Stream)))
-------------------------------------------------------------[Derive]
|||-  [* tl(f(S:Stream)) *] = [* zip(not(S:Stream),tl(S:Stream)) *]


|-  [* tl(f(S:Stream)) *] = [* zip(not(S:Stream),tl(S:Stream)) *]
------------------------------------------------------------[Normalize]
|-  [* tl(f(S:Stream)) *] = [* tl(zip(S:Stream,not(S:Stream))) *]


|-  [* hd(f(S:Stream)) *] = [* hd(zip(S:Stream,not(S:Stream))) *]
-----------------------------------------------------------[Reduce]
|||-  [* hd(f(S:Stream)) *] = [* hd(zip(S:Stream,not(S:Stream))) *]


1. |||-  [* hd(f(S:Stream)) *] = [* hd(zip(S:Stream,not(S:Stream))) *]
2. |||-  [* tl(f(S:Stream)) *] = [* tl(zip(S:Stream,not(S:Stream))) *]
-----------------------------------------------------------[Derive]
|||-  [* f(S:Stream) *] = [* zip(S:Stream,not(S:Stream)) *]
```

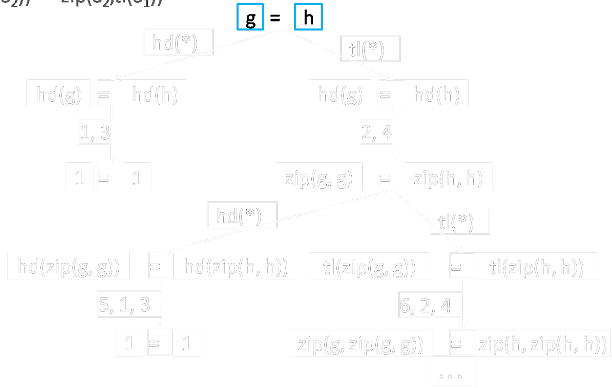
Plan

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)

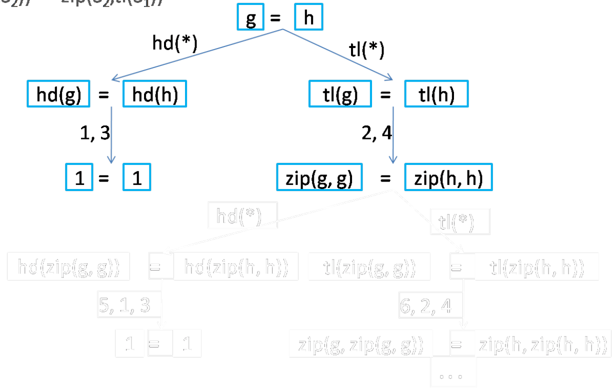


Special Contexts: Intuition

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))

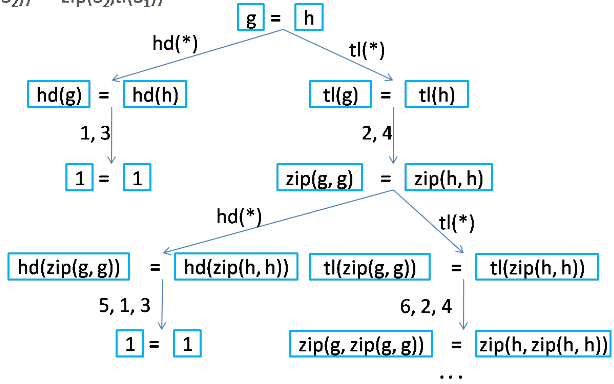
7. g = h

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)

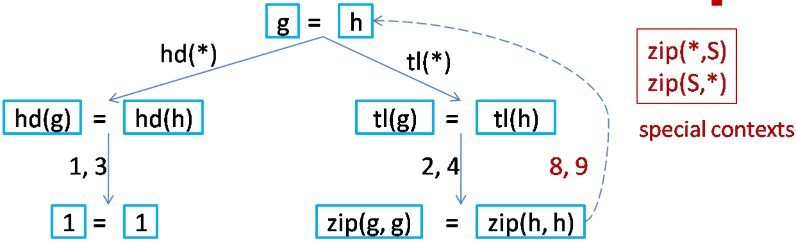


Special Contexts: Intuition

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))

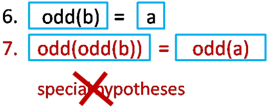
7. g = h
8. zip(g, g) = zip(g, h)
9. zip(g, h) = zip(h, h)

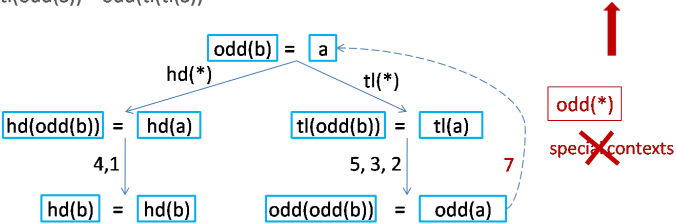
special hypotheses



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)))





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

Special Hypotheses

- 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: $f(\ldots * \ldots)$ special implies that

$$\boxed{f(\ldots t_i \ldots)} = \boxed{f(\ldots t'_i \ldots)}$$

  is sound and it can be added 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 $\boxed{f(\ldots t_i \ldots)} = \boxed{f(\ldots t'_i \ldots)}$

Extended Circular Coinduction Proof System

(Lucanu & Roșu, ICFEM 2009)

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

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

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

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

Plan

Equational Interpolants: Intuition

– we consider streams whose experiments return natural numbers
– we want to prove that merging two sorted streams we get a sorted stream

$$merge(B : S, B' : S') = \begin{cases} B : merge(S, B' : S') & \text{if } B \leq B' \\ B' : merge(B : S, S') & \text{if } B > B' \end{cases}$$

– the proof requires case analysis

Note. Since we want to use CC, "S is sorted" predicate must be encoded as a behavioral property:

$$toBits(B : B' : S) = \begin{cases} 1 : toBits(B' : S) & \text{if } B \leq B' \\ 0 : toBits(B' : S) & \text{otherwise} \end{cases}$$

S is sorted $\iff$ $toBits(S) \equiv ones$
where $ones = 1 : ones$

Equational interpolants

– case analysis as an inference rule

$$\frac{\begin{array}{c} hd(toBits(merge(S,S'))) = 1 \ if \ \ isSorted(S) \wedge hd(S) \leq hd(S') \\ hd(toBits(merge(S,S'))) = 1 \ if \ \ isSorted(S) \wedge hd(S) > hd(S') \end{array}}{hd(toBits(merge(S,S'))) = 1 \ if \ \ isSorted(S)}$$

– the above is an instance of what we call equational interpolants

– an equational interpolant is a pair $\langle e, itp\rangle$, where e is an equation and itp is a finite set of equations

– $(E, \vdash)$ is extended to specifications with interpolants:

$$\frac{E \vdash e}{(E,\mathcal{I}) \vdash e} \qquad \frac{(E,\mathcal{I}) \vdash itp}{(E,\mathcal{I}) \vdash e} \text{ if } \langle e, itp\rangle \in \mathcal{I}$$

CC extended with equational interpolants

– the proof system is enhanced with just one rule

$$\frac{\mathcal{B} \cup \mathcal{F} \Vdash^{\circlearrowleft} \mathcal{G} \cup \boxed{itp}}{\mathcal{B} \cup \mathcal{F} \Vdash^{\circlearrowleft} \mathcal{G} \cup \boxed{e}} \quad \text{if} \quad \langle e, itp \rangle \in \mathcal{I} \qquad [\text{itp}]$$

– equational interpolants can be used in two ways:

1. preserving the initial entailment relation ($E \vdash itp$ implies $E \vdash e$)
   example: generalization rule when a goal is replaced with a more general one
2. extending the initial entailment relation:

$$\frac{t(x) = t'(x) \ if \ even(x) = true, \quad t(x) = t'(x) \ if \ even(x) = false}{t(x) = t'(x)}$$

(equivalent to say that the spec is enriched with an inductive property)

a more elaborated example:
M. Bonsangue et al. A decision procedure for bisimilarity of generalized regular expressions. SBMF 2010.

Case analysis as equational interpolants

- annotated case sentences: (pattern, cases)
- if there is an instance θ of the pattern in $t = t'$, then we have the equational interpolant
  $(e, \{t = t' \ \mathit{if} \ \ c \wedge \theta(\mathit{case}_1), \ldots, t = t' \ \mathit{if} \ \ c \wedge \theta(\mathit{case}_n)\})$

Main idea: use special syntactical constructs from which equational interoplants to be used are automatically generated

- enumerated sorts:
  ```
  enum Bit is 0 1 .
  ```
  defines the ann. case sent. $(B{:}Bit, B = 0 \vee tB = 1)$
- guarded equations:
  ```
  geq hd(merge(S1, S2)) =
    hd(S1) if hd(S1) <  hd(S2) = true  []
    hd(S2) if hd(S1) <= hd(S2) = false [] .
  ```

defines the ann. case sent.
$(hd(merge(S_1, S_2)), hd(S_1) < hd(S_2) = true \vee hd(S_1) < hd(S_2) = true)$

An example

M. Niqui and J.J.M.M. Rutten. Sampling, splitting and merging in coinductive stream calculus. In MPC 2010.

- specification:

$$Z3(a : S_1, S_2, S_3) = a : Z3(S_2, S_3, S_1)$$
$$T3(0)(a_0 : a_1 : a_2 : S) = a_0 : T3(0)(tl^3(S))$$
$$T3(1)(a_0 : a_1 : a_2 : S) = a_1 : T3(1)(tl^3(S))$$
$$T3(2)(a_0 : a_1 : a_2 : S) = a_2 : T3(2)(tl^3(S))$$
$$Rev3(N)(S) = Z3(T3(N)(S), T3(N-1)(S), T3(N-2)(S))$$

- property (goal):

$$Rev3(N)(Rev3(N)(S)) = S$$

- the proof uses the case sentence

`cases pattern` $= N$ `if` $N \bmod 3 = 0 \vee N \bmod 3 = 1 \vee N \bmod 3 = 2$.

- 12 case analyses and 14 new lemmas automatically discovered

Plan

Conclusion

Achievements:

- circular coinduction together with the special contexts and equational interpolants is a simple an powerful proof method by coinduction
- we defined patterns for case analysis (annotated case sentences) which can be handled as equational interpolants
- CIRC implementation of all above in an uniform way
- case studies include: streams, infinite trees, processes, (coalgebra) regular expressions

Future and in progress work:

- a new proving technique recently implemented in CIRC is circular induction
- extend this new technique with case analysis (it should be a matter of routine)
- extend CIRC with bactracking procedure to automatically try different proving tactics

Thanks!