CIRC: A Behavioral Verification Tool based on Circular Coinduction

Dorel Lucanu[1], Eugen-Ioan Goriac[1], Georgiana Caltais[1], and Grigore Roşu[2]

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

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

CALCO Tools, 2009

Contents

- CIRC Introduction
- Understanding circular coinduction
- Behavioral specifications
- 3 demos
- Conclusions

CIRC Introduction

- What is CIRC ?
  - an automated theorem prover
  - based on the circularity principle both for coinduction and induction
  - a metalanguage application implemented as an extension of Maude
- Purpose
  - verification of programs
  - equivalence checking between programs
  - proving the bisimulation between processes
  - verification of protocols
- http://fsl.cs.uiuc.edu/index.php/Special:CircOnline

Example

- consider
  - two datatypes: *Bit* = {0, 1} and *Stream* ($S = a_1a_2a_3 \dots$)
  - two behavioral operations:
    - *hd* : *Stream* -> *Bit* ($hd(S) = a_1$)
    - *tl* : *Stream* -> *Stream* ($tl(S) = a_2a_3 \dots$)
- define the behavioral equivalence ≡ over *Streams* :
  - experiments: *hd*(*), *hd*(*tl*(*)), *hd*(*tl*(*tl*(*))) ...
  - $S_1 \equiv S_2$ iff $C[S_1] = C[S_2]$, forall experiments C

Example

- consider
  - three other operations:
    - *zip* : *Stream* *Stream* -> *Stream*
    - *odd, even* : *Stream* -> *Stream*
    - *zip*($a_1\ a_2$ ... , $a'_1\ a'_2$ ...) = $a_1\ a'_1\ a_2\ a'_2$ ...
    - *odd*($a_1\ a_2\ a_3\ a_4$ ...) = $a_1\ a_3$ ...
    - *even*($a_1\ a_2\ a_3\ a_4$ ...) = $a_2\ a_4$ ...
- let us prove that
  - *zip*(*odd*(S), *even*(S)) ≡ S

Example – understanding circular coinduction

Example – understanding circular coinduction

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))

Example – understanding circular coinduction

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. $tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))$

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

Example – understanding circular coinduction

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. $tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))$

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

$$\Big\downarrow 3$$

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

Example – understanding circular coinduction

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

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

$$\downarrow 3$$

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

Example – understanding circular coinduction

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

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

↓ 3

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

hd(*) ↙ ↘ tl(*)

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

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

Example – understanding circular coinduction

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

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

↓ 3

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

hd(*) ↙ ↘ tl(*)

$$hd(zip(odd(S), odd(tl(S)))) = hd(S) \qquad tl(zip(odd(S), odd(tl(S)))) = tl(S)$$

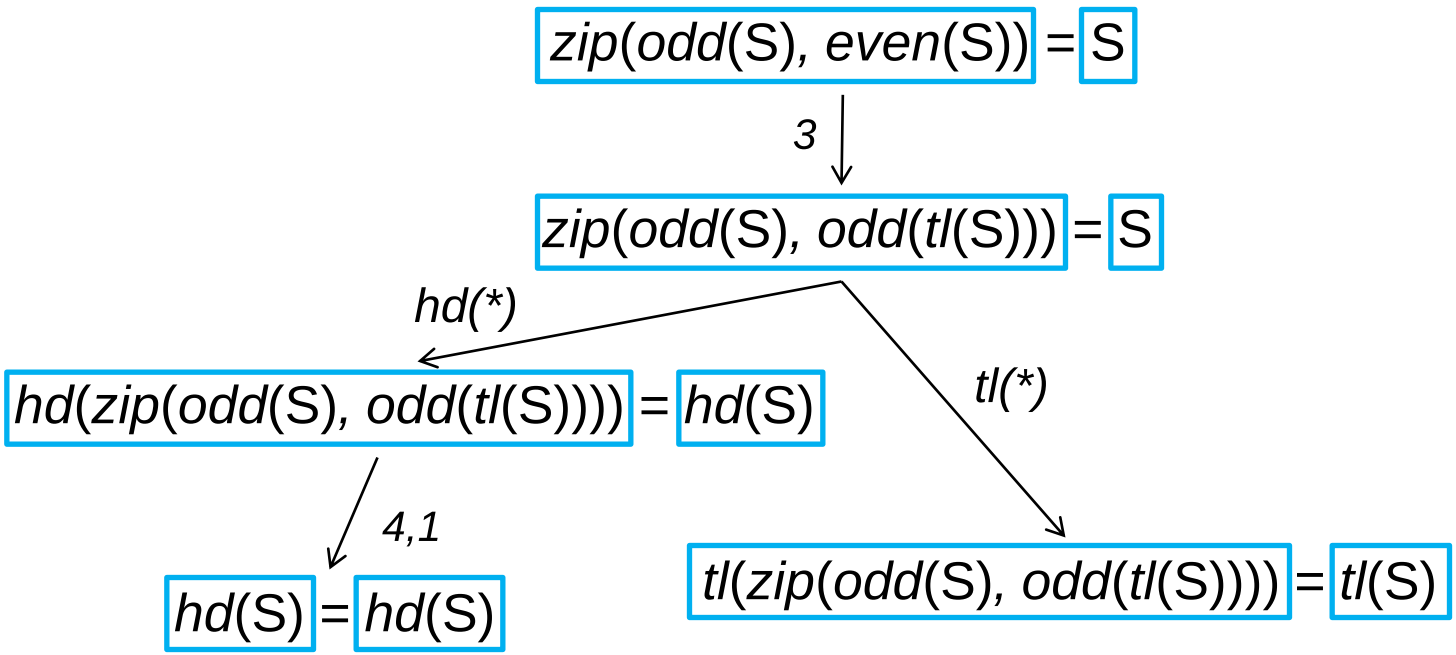
↓ 4,1

$$hd(S) = hd(S)$$

Example – understanding circular coinduction

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. $tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))$

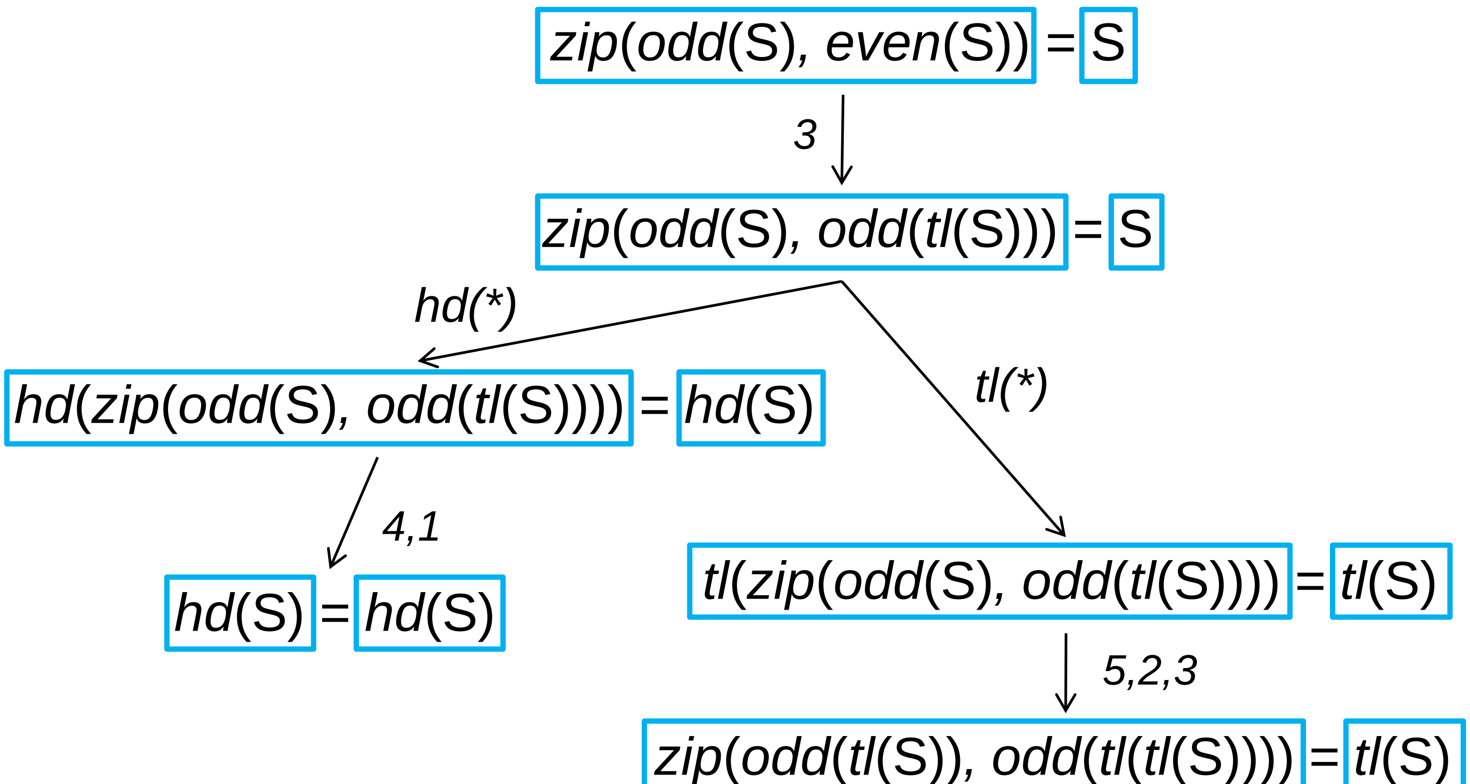
6. $zip(odd(S), odd(tl(S))) = S$



Example – understanding circular coinduction

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. $tl(zip(S_1,S_2)) = zip(S_2,tl(S_1))$

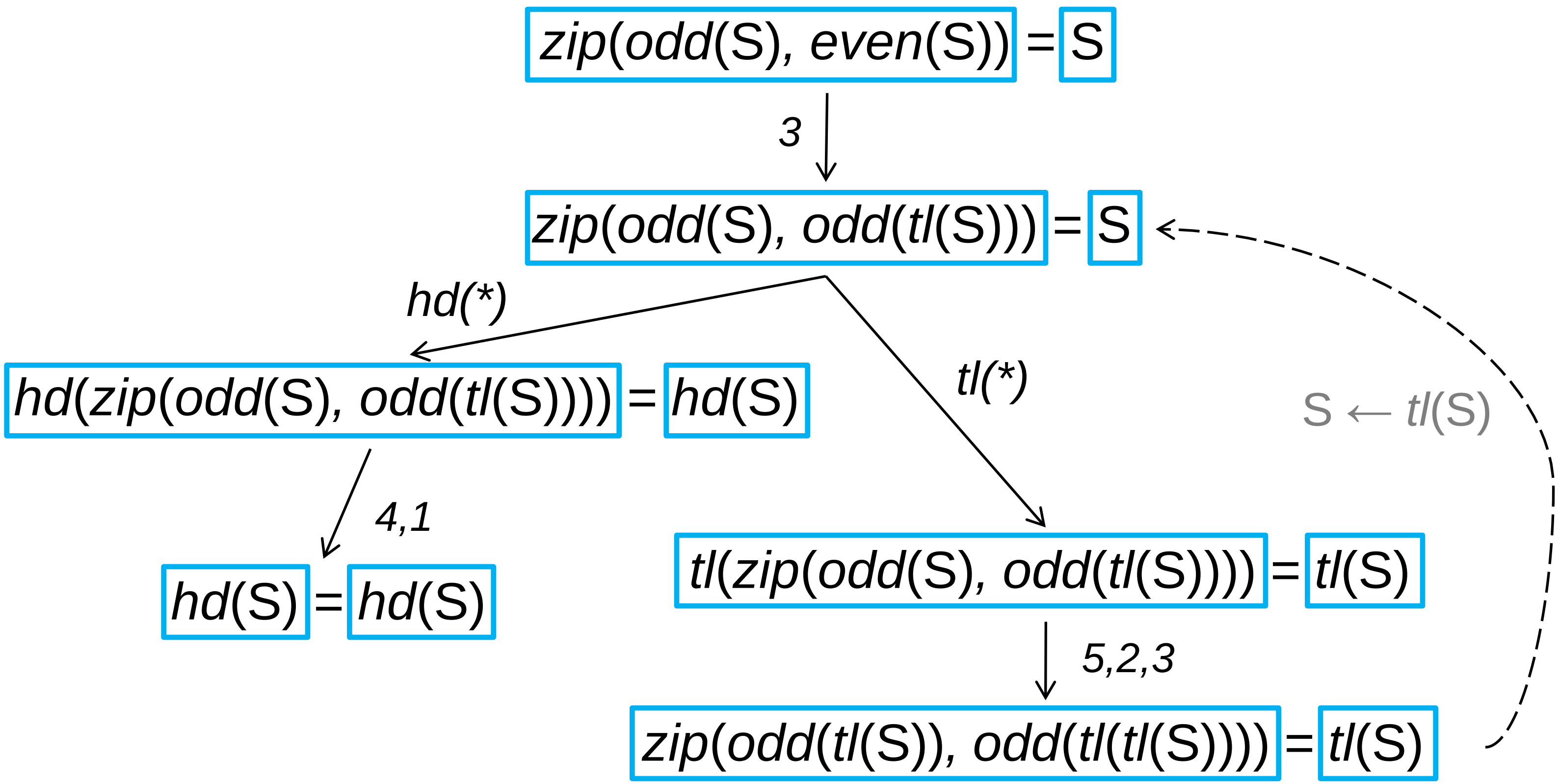
6. $zip(odd(S), odd(tl(S))) = S$



Example – understanding circular coinduction

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



CIRC reduction rules

[Done]: $(\mathcal{B}, \mathcal{F}, \emptyset) \Rightarrow \cdot$

[Reduce]: $(\mathcal{B}, \mathcal{F}, \mathcal{G} \cup \{\boxed{e}\}) \Rightarrow (\mathcal{B}, \mathcal{F}, \mathcal{G})$

if $\mathcal{B} \cup \mathcal{F} \vdash_{\rightarrow\leftarrow} e$

[Derive]: $(\mathcal{B}, \mathcal{F}, \mathcal{G} \cup \{\boxed{e}\}) \Rightarrow (\mathcal{B}, \mathcal{F} \cup \{\boxed{e}\}, \mathcal{G} \cup \{\boxed{\Delta(e)}\})$

if $\mathcal{B} \cup \mathcal{F} \not\vdash_{\rightarrow\leftarrow} e$

[Normalize]: $(\mathcal{B}, \mathcal{F}, \mathcal{G} \cup \{\boxed{e}\}) \Rightarrow (\mathcal{B}, \mathcal{F}, \mathcal{G} \cup \{\boxed{\mathrm{nf}(e)}\})$

[Fail]: $(\mathcal{B}, \mathcal{F}, \mathcal{G} \cup \{\boxed{e}\}) \Rightarrow \mathit{fail}$

if $\mathcal{B} \cup \mathcal{F} \not\vdash_{\rightarrow\leftarrow} e$ and e is visible

The equational specification

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

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

  op zip : Stream Stream ->
           Stream .
  op odd : Stream -> Stream .
  op even : Stream -> Stream .

  var S S1 S2 : Stream .

  eq hd(odd(S)) = hd(S) .
  eq tl(odd(S)) = even(tl(S)) .
  eq even(S) = odd(tl(S)) .

  eq hd(zip(S1, S2)) = hd(S1) .
  eq tl(zip(S1, S2)) =
     zip(S2, tl(S1)) .

endtheory)
```

The behavioral specification

```
(ctheory STREAM is
  including STREAM-EQ .
  derivative hd(*:Stream) .
  derivative tl(*:Stream) .
endctheory)

Maude> (add goal zip(odd(S:Stream), even(S:Stream)) =
                 S:Stream .)
Maude> (coinduction .)

Proof succeeded.
  Number of derived goals: 2
  Number of proving steps performed: 12
  Maximum number of proving steps is set to: 256
Proved properties:
  zip(odd(S:Stream),odd(tl(S:Stream))) = S:Stream
```

DEMO

01.zip-odd-even.maude

The main commands

```
(add goal lhs = rhs .)
(coinduction .)
(show proof .)

(set show details on/off .)
```

DEMO

02.morse.maude

Other important commands

```
(add goal (op declaration [attr] .) .) (*)
(coinduction-grlz .)

(set auto contexts on/off .)
(set max no steps number .)
```

(*) Details in [G. Grigoraş, D. Lucanu, G. Caltais, and E. Goriac, *Automated proving of the behavioral attributes,* BCI2009]

DEMO

03.combine-ind-coind.maude

Advanced commands

```
(reduce .)
(derive .)
(save proof state .)
(undo .)
(apply strategy .) (*)
```

(*) Details in [G. Caltais, E.-I. Goriac, D. Lucanu, and G. Grigoraş, *A Rewrite Stack Machine for ROC!*, SYNASC2008, IEEE Proc.]

Conclusions

- Other CIRC features
  - simplification rules and proof correctness verification
  - conditional goals
  - declaration of enumerable types
- Future aims
  - extension with automated case analysis
  - implementation of a strategy for automatically combining induction and coinduction

Thank you !