CIRC prover: an overview

Dorel Lucanu

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

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

2 Circular Coinduction

3 Special Contexts

4 Equational Interpolants

5 Conclusion
Plan

1. Introduction
2. Circular Coinduction
3. Special Contexts
4. Equational Interpolants
5. Conclusion
Circular Coinduction and CIRC

- joint work Al. I. Cuza Univ. of Iasi (UAIC, RO) and Univ. of Illinois at Urbana-Champaign (UIUC, US)
- Theoretical Achievements:
  - **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, $\Sigma$ a $S$-signature, $E$ a set of (conditional) equations

- a $\Sigma$-context $C$ is a $\Sigma$-term with one occurrence of a distinguished variable $*:s$ of sort $s$

- contexts as equation transformers: if $e$ is $(\forall X)\ t = t'\text{ if } cond$, then $C[e]$ denotes $(\forall X \cup Y)\ C[t] = C[t']\text{ if } cond$

- behavioral specification $\mathcal{B} = (S, (\Sigma, \Delta), E)$, where $\Delta$ is a set of $\Sigma$-contexts
  - hidden sorts: $H = \{h | \delta[{*:h}] \in \Delta\}$, and
  - visible sorts: $V = S \setminus H$

- experiment $= a \ \Delta$-context of visible sort
Behavioral Equivalence

- contextual entailment system: an entailment relation $\vdash$ satisfying reflexivity, monotonicity, transitivity, and $\Delta$-congruence ($E \vdash e$ implies $E \vdash \delta[e]$ for each $\delta \in \Delta$)

- we write $B \vdash e$ for $E \vdash e$, where $B = (S, (\Sigma, \Delta), E)$

- behavioral entailment: $B \Vdash e$ iff $B \vdash C[e]$ for each $\Delta$-experiment $C$ appropriate for the equation $e$

- behavioral equivalence: $\equiv = \{ e \mid B \Vdash e \}$

Example of streams:
- experiments:
  - $\text{hd}(\ast:\text{Stream})$, $\text{hd}(\text{tl}(\ast:\text{Stream}))$, $\text{hd}(\text{tl}(\text{tl}(\ast:\text{Stream})))$, ...

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

- showing beh. equiv. is $\Pi^0_2$-hard (S. Buss, G. Roșu, 2000, 2006)
Behavioral Specifications: Maude like syntax

(theory STREAM is

sort Bit .
op 0 1 : -> Bit .

ops ~_ : Bit -> Bit .
eq ~ 0 = 1 .
eq ~ 1 = 0 .

sort Stream .
var S, S’ : Stream .

derivative hd(*:Stream) .
derivative tl(*:Stream) .

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

eq hd(not(S)) = ~ hd(S) .
eq hd(f(S)) = hd(S) .
eq hd(tl(S)) = ~ hd(S) .
eq hd(tl(S)) = f(tl(S)) .

op not : Stream -> Stream .

op f : Stream -> Stream .

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

endtheory)
Plan

1. Introduction
2. Circular Coinduction
3. Special Contexts
4. Equational Interpolants
5. Conclusion
Circular Coinduction: Intuition

\[ f(B:S) = B : \neg B : S \]
\[ \text{zip}(B:S',S') = B':\text{zip}(S',S) \]
\[ \text{not}(B:S) = \neg B:\text{not}(S) \]

\[ f(S) = \text{zip}(S, \text{not}(S)) \]

\[ \text{hd}(f(S)) = \text{hd}(\text{zip}(S, \text{not}(S))) \]
\[ \text{tl}(f(S)) = \text{tl}(\text{zip}(S, \text{not}(S))) \]
\[ \text{hd}(\text{tl}(f(S))) = \text{hd}(\text{tl}(\text{zip}(S, \text{not}(S)))) \]
\[ \text{tl}^2(f(S)) = \text{tl}^2(\text{zip}(S, \text{not}(S))) \]
\[ f(\text{tl}(S)) = \text{zip}(\text{tl}(S), \text{not}(\text{tl}(S))) \]
Circular Coinduction: Intuition

\[
f(B:S) = B : \sim B : S \\
\text{zip}(B:S,S') = B':\text{zip}(S',S) \\
\text{not}(B:S) = \sim B:\text{not}(S)
\]

\[
f(S) = \text{zip}(S, \text{not}(S))
\]

\[
\begin{align*}
\text{hd}(f(S)) &= \text{hd}(\text{zip}(S, \text{not}(S))) \\
\text{tl}(f(S)) &= \text{tl}(\text{zip}(S, \text{not}(S)))
\end{align*}
\]

\[
\begin{align*}
\text{hd}(\text{tl}(f(S))) &= \text{hd}(\text{tl}(\text{zip}(S, \text{not}(S)))) \\
\text{tl}^2(f(S)) &= \text{tl}^2(\text{zip}(S, \text{not}(S))) \\
f(\text{tl}(S)) &= \text{zip}(\text{tl}(S), \text{not}(\text{tl}(S))}
\end{align*}
\]
Circular Coinduction: Intuition

\[ f(B : S) = B : \sim B : S \]
\[ \text{zip}(B : S, S') = B' : \text{zip}(S', S) \]
\[ \text{not}(B : S) = \sim B : \text{not}(S) \]

\[ f(S) = \text{zip}(S, \text{not}(S)) \]

\[ \text{hd}(f(S)) = \text{hd}(\text{zip}(S, \text{not}(S))) \]
\[ \text{tl}(f(S)) = \text{tl}(\text{zip}(S, \text{not}(S))) \]

\[ \text{hd}(\text{tl}(f(S))) = \text{hd}(\text{tl}(\text{zip}(S, \text{not}(S)))) \]
\[ \text{tl}^2(f(S)) = \text{tl}^2(\text{zip}(S, \text{not}(S))) \]
\[ f(\text{tl}(S)) = \text{zip}(\text{tl}(S), \text{not}(\text{tl}(S))) \]
Circular Coinduction: Intuition

\[ f(B:S) = B : \sim B : S \]
\[ \text{zip}(B:S,S') = B' \cdot \text{zip}(S',S) \]
\[ \text{not}(B:S) = \sim B \cdot \text{not}(S) \]

\[
\begin{align*}
\text{hd}(f(S)) &= \text{hd}(\text{zip}(S, \text{not}(S))) \\
\text{tl}(f(S)) &= \text{tl}(\text{zip}(S, \text{not}(S))) \\
\text{hd}(\text{tl}(f(S))) &= \text{hd}(\text{tl}(\text{zip}(S, \text{not}(S)))) \\
\text{tl}^2(f(S)) &= \text{tl}^2(\text{zip}(S, \text{not}(S))) \\
f(\text{tl}(S)) &= \text{zip}(\text{tl}(S), \text{not}(\text{tl}(S)))
\end{align*}
\]
Circular Coinduction Proof System

(Roșu & Lucanu, CALCO 2009)

\( \mathcal{B} \) a behavioral specification \((S, \Sigma, E)\)

\( \Delta \) a set of derivatives

\( \mathcal{F} \) a set of frozen hypotheses \( e ::= t = t' \) if \( \text{cond} \)

\( \mathcal{G} \) a set of goals, which are frozen equations

\( \vdash \) an entailment relation \( \vdash \) between \( \mathcal{B} \) and equations

\[ \begin{align*}
    & \mathcal{B} \cup \mathcal{F} \vdash^\varnothing \emptyset, \\
    & \mathcal{B} \cup \mathcal{F} \vdash^\varnothing \mathcal{G}, \quad \mathcal{B} \cup \mathcal{F} \vdash e \\
    & \mathcal{B} \cup \mathcal{F} \cup \{e\} \vdash^\varnothing \mathcal{G} \cup \{\Delta[e]\}, \\
    & \mathcal{B} \cup \mathcal{F} \vdash^\varnothing \mathcal{G} \cup \{e\}
\end{align*} \]

[Done]

[Reduce]

[Derive]

if \( e \) derivable

D. Lucanu (UAIC)
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[δ[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:
  \[
  \ldots t_i \ldots = \ldots t'_i \ldots \\
  f(\ldots t_i \ldots) = f(\ldots t'_i \ldots)
  \]
- the other rules of equational deduction are sound in combination with the accumulated hypotheses in \( F \), 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 output for

STREAM ⊨ 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. \[\vdash *\ \text{hd}(\text{tl}(f(S:\text{Stream}))) \] = \[ *\ \text{hd}(\text{zip}(\text{not}(S:\text{Stream}),\text{tl}(S:\text{Stream}))) \]
2. \[\vdash *\ \text{tl}(\text{tl}(f(S:\text{Stream}))) \] = \[ *\ \text{tl}(\text{zip}(\text{not}(S:\text{Stream}),\text{tl}(S:\text{Stream}))) \]

[Derive]
\[\vdash *\ \text{tl}(f(S:\text{Stream})) \] = \[ *\ \text{zip}(\text{not}(S:\text{Stream}),\text{tl}(S:\text{Stream})) \]

[Normalize]
\[\vdash *\ \text{tl}(f(S:\text{Stream})) \] = \[ *\ \text{tl}(\text{zip}(S:\text{Stream},\text{not}(S:\text{Stream}))) \]

[Reduce]
\[\vdash *\ \text{hd}(f(S:\text{Stream})) \] = \[ *\ \text{hd}(\text{zip}(S:\text{Stream},\text{not}(S:\text{Stream}))) \]

[Derive]
\[\vdash *\ f(S:\text{Stream}) \] = \[ *\ \text{zip}(S:\text{Stream},\text{not}(S:\text{Stream}))) \]
Plan

1. Introduction
2. Circular Coinduction
3. Special Contexts
4. Equational Interpolants
5. Conclusion
Special Contexts: Intuition

1. \( \text{hd}(g) = 1 \)
2. \( \text{tl}(g) = \text{zip}(g, g) \)
3. \( \text{hd}(h) = 1 \)
4. \( \text{tl}(h) = \text{zip}(h, h) \)
5. \( \text{hd}(\text{zip}(S_1, S_2)) = \text{hd}(S_1) \)
6. \( \text{tl}(\text{zip}(S_1, S_2)) = \text{zip}(S_2, \text{tl}(S_1)) \)
7. \( g = h \)
8. \( \text{zip}(g, g) = \text{zip}(h, h) \)
Special Contexts: Intuition

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

7. \begin{align*}
g & = h \\
\text{zip}(g, g) & = \text{zip}(h, h)
\end{align*}

8. \begin{align*}
\text{hd}(*) & = \text{hd} (*) \\
\text{tl}(*) & = \text{tl} (*)
\end{align*}

\begin{align*}
1 & = 1 \\
\text{zip}(g, g) & = \text{zip}(h, h) \\
\text{hd}(\text{zip}(g, g)) & = \text{hd}(\text{zip}(h, h)) \\
\text{tl}(\text{zip}(g, g)) & = \text{tl}(\text{zip}(h, h)) \\
1 & = 1 \\
\text{zip}(g, \text{zip}(g, g)) & = \text{zip}(h, \text{zip}(h, h))
\end{align*}
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. \( \text{hd}(g) = 1 \)
2. \( \text{tl}(g) = \text{zip}(g, g) \)
3. \( \text{hd}(h) = 1 \)
4. \( \text{tl}(h) = \text{zip}(h, h) \)
5. \( \text{hd}(\text{zip}(S_1, S_2)) = \text{hd}(S_1) \)
6. \( \text{tl}(\text{zip}(S_1, S_2)) = \text{zip}(S_2, \text{tl}(S_1)) \)

7. \( g = h \)
8. \( \text{zip}(g, g) = \text{zip}(g, h) \)
9. \( \text{zip}(g, h) = \text{zip}(h, h) \)

special hypotheses

\[ \text{hd}(\ast) \]
\[ \text{tl}(\ast) \]

1. \( \text{hd}(g) = \text{hd}(h) \)
2. \( \text{tl}(g) = \text{tl}(h) \)
3. \( 1 = 1 \)
4. \( \text{zip}(g, g) = \text{zip}(h, h) \)

special contexts

zip(\ast, S)
zip(S, \ast)
1. \( \text{hd}(a) = \text{hd}(b) \)
2. \( \text{tl}(a) = \text{odd}(a) \)
3. \( \text{tl}(\text{tl}(b)) = \text{odd}(b) \)
4. \( \text{hd}(\text{odd}(S)) = \text{hd}(S) \)
5. \( \text{tl}(\text{odd}(S)) = \text{odd}(\text{tl}(\text{tl}(S))) \)

6. \( \text{odd}(b) = a \)
7. \( \text{odd}(\text{odd}(b)) = \text{odd}(a) \)

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

\[
\ldots \boxed{t_i} \ldots = \ldots \boxed{t_i'} \ldots
\]

\[
\frac{\boxed{f(\ldots t_i \ldots)}}{f(\ldots t_i' \ldots)}
\]

with a set of special hypotheses: \(f(\ldots \ast \ldots)\) 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 \(f(\ldots \ast \ldots)\) is safe (special), then add to \(\mathcal{F}\) simultaneously \(t_i = t_i'\)

and \(f(\ldots t_i \ldots) = f(\ldots t_i' \ldots)\)
Extended Circular Coinduction Proof System

(Lucanu & Roșu, ICFEM 2009)

\[
\begin{array}{c}
\vdash B \cup F \vdash \emptyset \\
\hline
B \cup F \vdash G, \quad B \cup F \vdash e \\
\hline
B \cup F \vdash G \cup \{e\}
\end{array}
\]

[Done]

\[
\begin{array}{c}
B \cup F \cup \{e\} \vdash \Gamma[e] \vdash G \cup \Delta[e] \\
\hline
B \cup F \vdash G \cup \{e\}
\end{array}
\]

[Reduce]

[Derive\textsuperscript{scx}]

where $\Gamma$ is a given set of special contexts

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

How can we find such a $\Gamma$?

$\Rightarrow$ CIRC tool provides an algorithm computing a $\Gamma$
Plan

1. Introduction
2. Circular Coinduction
3. Special Contexts
4. Equational Interpolants
5. Conclusion
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

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

– the proof requires case analysis

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

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

\(S \text{ is sorted} \iff \text{toBits}(S) \equiv \text{ones}\)

where \(\text{ones} = 1 : \text{ones}\)
Equational interpolants

– case analysis as an inference rule

\[
\begin{align*}
\text{hd}(\text{toBits}(\text{merge}(S, S')))) &= 1 \text{ if } \text{isSorted}(S) \land \text{hd}(S) \leq \text{hd}(S') \\
\text{hd}(\text{toBits}(\text{merge}(S, S')))) &= 1 \text{ if } \text{isSorted}(S) \land \text{hd}(S) > \text{hd}(S') \\
\text{hd}(\text{toBits}(\text{merge}(S, S')))) &= 1 \text{ if } \text{isSorted}(S)
\end{align*}
\]

– 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:

\[
\begin{align*}
E \vdash e & \quad \frac{E \vdash e \quad (E, I) \vdash itp}{(E, I) \vdash e} \\
\text{if } \langle e, itp \rangle \in I
\end{align*}
\]
CC extended with equational interpolants

– the proof system is enhanced with just one rule

\[
\frac{B \cup F \models \emptyset \quad G \cup \text{itp}}{B \cup F \models \emptyset \quad G \cup \text{e}}
\]

if \( \langle e, \text{itp} \rangle \in I \) \[\text{[itp]}\]

– equational interpolants can be used in two ways:

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

2. extending the initial entailment relation:

\[
t(x) = t'(x) \text{ if } \text{even}(x) = \text{true}, \quad t(x) = t'(x) \text{ if } \text{even}(x) = \text{false}
\]

\[
t(x) = t'(x)
\]

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

a more elaborated example:

Case analysis as equational interpolants

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

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

- enumerated sorts:
  ```
  enum Bit is 0 1 .
  defines the ann. case sent. \((B::Bit, B = 0 \lor tB = 1)\)
  ```

- guarded equations:
  ```
  geq \text{hd}(\text{merge}(S_1, S_2)) =
  \text{hd}(S_1) \text{ if } \text{hd}(S_1) < \text{hd}(S_2) = \text{true } []
  \text{hd}(S_2) \text{ if } \text{hd}(S_1) \leq \text{hd}(S_2) = \text{false } [] .
  ```
  defines the ann. case sent.
  \[
  (\text{hd}(\text{merge}(S_1, S_2)), \text{hd}(S_1) < \text{hd}(S_2) = \text{true} \lor \text{hd}(S_1) \leq \text{hd}(S_2) = \text{true})
  \]
An example


- specification:
  \[ Z_3(a : S_1, S_2, S_3) = a : Z_3(S_2, S_3, S_1) \]
  \[ T_3(0)(a_0 : a_1 : a_2 : S) = a_0 : T_3(0)(tl^3(S)) \]
  \[ T_3(1)(a_0 : a_1 : a_2 : S) = a_1 : T_3(1)(tl^3(S)) \]
  \[ T_3(2)(a_0 : a_1 : a_2 : S) = a_2 : T_3(2)(tl^3(S)) \]
  \[ Rev_3(N)(S) = Z_3(T_3(N)(S), T_3(N - 1)(S), T_3(N - 2)(S)) \]

- property (goal):
  \[ Rev_3(N)(Rev_3(N)(S)) = S \]

- the proof uses the case sentence
  \[ \text{cases pattern } = N \text{ if } N \mod 3 = 0 \lor N \mod 3 = 1 \lor N \mod 3 = 2. \]

- 12 case analyses and 14 new lemmas automatically discovered
Plan

1. Introduction
2. Circular Coinduction
3. Special Contexts
4. Equational Interpolants
5. Conclusion
Conclusion

Achievements:

- Circular coinduction together with the special contexts and equational interpolants is a simple and 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 backtracking procedure to automatically try different proving tactics.
Thanks!