A Common Framework for Induction and Coinduction
Work in progress

Dorel Lucanu

1Department of Computer Science
Alexandru Ioan Cuza University of Iaşi

UAIC 2015
1. Induction and Coinduction on Powerset

2. (Co)Inductive Sets Defined by Ground Inference Systems

3. Application to Program Verification

4. Conclusion
Motivating Example: Terminating and Nonterminating programs

```
while (x != 0)
{
    s = s+x;
    x = x-1;
}
```

```
while (true)
{
    input := getInput();
    process(input);
}
```

- how to specify what the above programs do?
- how to prove that the programs meet their specification?
- how to specify that a program (does not) terminate?
- how to prove that a program (does not) terminate?
Motivating Example: Rules

Consider the following BNF grammar:

\[ EP ::= C | C \rightarrow EP \]

that is equivalent to

\[
\begin{align*}
  c \in EP & \quad \text{if } c \in C \\
  c \rightarrow \rho \in EP & \quad \text{if } c \in C
\end{align*}
\]

- the least set satisfying the above equations is the set of finite nonempty lists over \( C \)
- the greatest set satisfying the above equations is the set of infinite and finite nonempty lists over \( C \)
- if \( C \) is the set of configurations, then \( EP \) can be thought as the set of execution paths

How can we formalize it?
Motivating Example: Rules

Consider the following BNF grammar:

\[ EP ::= C \mid C \rightarrow EP \]

that is equivalent to

\[
\begin{align*}
    c \in EP & \quad \text{if } c \in C \\
    \rho \in EP & \quad \text{if } c \in C
\end{align*}
\]

- the least set satisfying the above equations is the set of finite nonempty lists over \( C \)
- the greatest set satisfying the above equations is the set of infinite and finite nonempty lists over \( C \)
- if \( C \) is the set of configurations, then \( EP \) can be thought as the set of execution paths

How can we formalize it?
Plan

1. Induction and Coinduction on Powerset
2. (Co)Inductive Sets Defined by Ground Inference Systems
3. Application to Program Verification
4. Conclusion
Theorem (Knaster-Tarski)

Let $U$ be a set. Any $F : \mathcal{P}(U) \to \mathcal{P}(U)$ monotone w.r.t. $\subseteq$ has a least fixed point $\mu \ y. \ F(y)$ (on short $\mu F$) and a greatest fixed point $\nu \ y. \ F(y)$ (on short $\nu F$).

Moreover,

$$\mu F = \bigcup \{X \mid F(X) \subseteq X\} \text{ and}$$

$$\nu F = \bigcap \{X \mid X \subseteq F(X)\}$$

Definition

$\mu F$ is inductively defined by $F$ and $\nu F$ is coinductively defined by $F$. 
(Co)Induction Proof Principle

Definition

Induction and Coinduction Inference Rules:

- induction proof principle:
  \[ F(X) \subseteq X \quad \mu \text{-rule} \]
  \[ \mu Y. F(Y) \subseteq X \]

- coinduction proof principle:
  \[ X \subseteq F(X) \quad \nu \text{-rule} \]
  \[ X \subseteq \nu Y. F(Y) \]

Definition

- \( X \) is forward closed w.r.t. \( F \) if \( F(X) \subseteq X \)
- \( X \) is backward closed w.r.t. \( F \) if \( X \subseteq F(X) \).
(Co)Continuous Functions and Kleene Theorem

Definition

\[ F : \mathcal{P}(U) \to \mathcal{P}(U) \text{ is continuous if } F(\bigcup_{n \geq 0} X_n) = \bigcup_{n \geq 0} F(X_n) \text{ for any increasing chain } X_0 \subseteq X_1 \subseteq \cdots. \]

\[ F : \mathcal{P}(U) \to \mathcal{P}(U) \text{ is cocontinuous if } F(\bigcap_{n \geq 0} X_n) = \bigcap_{n \geq 0} F(X_n) \text{ for any decreasing chain } X_0 \supseteq X_1 \supseteq \cdots. \]

Theorem (Kleene)

If \( F : \mathcal{P}(U) \to \mathcal{P}(U) \text{ is continuous then } \mu F = \bigcup_{n \geq 0} F^n(\bot). \)

If \( F : \mathcal{P}(U) \to \mathcal{P}(U) \text{ is cocontinuous then } \nu F = \bigcap_{n \geq 0} F^n(\top). \)
Definition

Let \( U \) be a set. A ground (inference) rule on \( U \) is a pair \((S, x)\), where \( S \subseteq U, \; x \in U \).

\( S \) is called the premise of the rule and \( x \) the conclusion of the rule.

If \( S = \{x_1, x_2, \ldots\}\), then a rule \((S, x)\) is written as

\[
\frac{x_1, x_2, \ldots}{x}
\]
Definition
A set $\mathcal{R}$ of ground rules yields a function $\hat{\mathcal{R}} : \mathcal{P}(U) \rightarrow \mathcal{P}(U)$ given by

$$\hat{\mathcal{R}}(X) = \{x \mid (\exists S' \subseteq X)(S', x) \in \mathcal{R}\}.$$  

Proposition

*If $\mathcal{R}$ is a set of ground rules, then $\hat{\mathcal{R}}$ is monotone.*

It follows that each set of ground rules $\mathcal{R}$ inductively defines a set $\mu \hat{\mathcal{R}}$ and coinductively defines a set $\nu \hat{\mathcal{R}}$. 
Example: EP

\[ [A] \xrightarrow{c} \text{ if } c \in C \quad [B] \xrightarrow{\rho} \text{ if } c \in C \]

The set of finite executions: \( C^+ = \mu [A, B] \)

The set of infinite and finite executions: \( C^\infty = \nu [A, B] \)

The set of infinite executions: \( C^\omega = \nu [B] \)

We have \( C^\infty = C^+ \cup C^\omega \).
A Case When $\hat{\mathcal{R}}$ is (Co)Continuous

**Proposition**

Let $\mathcal{R}$ be a set of ground rules.

If for $(S, x) \in \mathcal{R}$, $S$ is finite, then $\hat{\mathcal{R}}$ is continuous.

If for any $x$, the set $\{S \mid (S, x) \in \mathcal{R}\}$ is finite, then $\hat{\mathcal{R}}$ is cocontinuous.
(Co)Induction Proof Principle on Ground Inference Systems

If $\mathcal{R}$ is a set of ground rules, then

- the induction proof principle becomes

\[
\frac{\hat{\mathcal{R}}(X) \subseteq X}{\mu Y. \hat{\mathcal{R}}(Y) \subseteq X}
\]

i.e., for a given $X$, if for all rules $(S, x) \in \mathcal{R}$, $S \subseteq X$ implies $x \in X$, then (the set inductively defined by the rules) $\subseteq X$;

- and the coinduction proof principle becomes

\[
\frac{X \subseteq \hat{\mathcal{R}}(X)}{X \subseteq \nu Y. \hat{\mathcal{R}}(Y)}
\]

i.e., for a given $X$, if for all $x \in X$ there is a rule $(S, x) \in \mathcal{R}$ with $S \subseteq X$, then $X \subseteq$ (the set coinductively defined by the rules).
$X$ is forward closed iff $\hat{R}(X) \subseteq X$.

Intuitively, a set $X$ is forward closed if

for each rule whose premise is included in $X$ there is an element of $X$ that is the conclusion of the rule.

$X$ is backward closed iff $X \subseteq \hat{R}(X)$.

Intuitively, a set $X$ is backward closed if

for each element of $X$, there is a rule whose premise is included in $X$ that is the conclusion of the rule.
Proof Trees

Definition (Proof Trees)

Let $\mathcal{R}$ be a set of ground rules over $U$ and $x \in U$. A (finite or infinite) tree $T$ is a proof tree of $x$ under $\mathcal{R}$ if it satisfies the following properties:

- the root of $T$ is labelled with $x$;
- if $y$ is the label of a node of $T$ and $S$ is the set of labels of the children of this node, then $(S, y) \in \mathcal{R}$.

We often refer the nodes of a proof tree $T$ by their labels. Note that a proof tree can be finite or infinite.
A tree is well-founded if the relation on the nodes that contains a pair of nodes $(n, p)$ if $p$ is the parent of $n$ is well-founded.

**Proposition**

Let $\mathcal{R}$ be a set of a set of ground rules over $U$ such that $\mathcal{R}$ is cocontinuous.

Then $x \in \mu \hat{\mathcal{R}}$ iff there is a well-founded proof tree of $x$ under $\mathcal{R}$.

Then $x \in \nu \hat{\mathcal{R}}$ iff there is a proof tree of $x$ under $\mathcal{R}$. 
Plan

1. Induction and Coinduction on Powerset
2. (Co)Inductive Sets Defined by Ground Inference Systems
3. Application to Program Verification
4. Conclusion
Motivation

- formal operational semantics of programming language are specifications of execution paths
- the definition of execution paths is coinductive
- hence it is natural to use the coinduction for proving program properties
- can we adapt circular coinduction proof technique for proving properties of program executions?
  - what is a derivative?
  - how the circular coinduction rule looks like?
Configuration specifications as (Topmost) Matching Logic Formulas

\[
\langle x = x - 3; \text{ if } (x > 0) x = 1; , x \mapsto a \rangle \land (a > \text{Int} - 7)
\]

\[
(\exists b)\langle \text{ if } (x > 0) x = 1; , x \mapsto b \rangle \land (b = \text{Int} a - \text{Int} 3 \land a > \text{Int} - 7)
\]

Models: \((\gamma, \rho)\), where \(\gamma\) is a concrete configuration, and \(\rho\) a valuation

Example: \(\rho(a) = 5, \rho(b) = 2\)
\[\gamma = \langle x = x - 3; \text{ if } (x > 0) x = 1; , x \mapsto 2 \rangle\]

We have \(\rho(b) = \rho(a) - \text{Int} 3, \rho(a) > \text{Int} - 7\), and
\[\rho(\langle \text{ if } (x > 0) x = 1; , x \mapsto b \rangle) = \gamma\]
Programming Language Semantics Specifications

\((\Sigma, \Pi, M, S)\), where is a set of pairs \(\varphi \Rightarrow \varphi'\)

\[\langle X = E; P, \sigma \rangle \Rightarrow \langle P, \sigma[X \mapsto \sigma[E]] \rangle\]

\[\langle \text{while } (E) \{ B \} P, \sigma \rangle \land \sigma[E] = \text{Bool} \ false \Rightarrow \langle P, \sigma \rangle\]

\[\langle \text{while } (E) \{ B \} P, \sigma \rangle \land \sigma[E] = \text{Bool} \ true \Rightarrow \langle B \text{ while } (E) \{ B \} P, \sigma \rangle\]
All Paths Reachability Logic

\[
\langle \text{while} \ (x \neq 0) \ \{ \ s = s+x; \ x = x-1; \ \}, x \mapsto a \ s \mapsto 0 \rangle
\]

\[
\Rightarrow \ (\exists b) \langle \bullet, x \mapsto 0 \ s \mapsto b \rangle \land b = \text{Int} \ \frac{a(a + \text{Int} 1)}{2}
\]

all paths reachability formula: \( \varphi \Rightarrow \varphi' \)

\[
(\gamma_0 \rightarrow \gamma_1 \rightarrow \gamma_\ldots \cdot \cdot , \rho) \models \varphi \Rightarrow \varphi' \text{ iff } (\gamma_0, \rho) \models \varphi \text{ and there is } j \geq 0 \text{ s.t. } (\gamma_j, \rho) \models \varphi'
\]

\( S \models \varphi \Rightarrow \varphi' \text{ iff for all finite and complete } (\tau, \rho) \text{ starting from } \varphi, (\tau, \rho) \models \varphi \Rightarrow \varphi' \)
Proving All Paths Reachability Specs

- there is a general proof systems (Roşu et al., RTA 2014); nice but not easy to handle it in practice

- a procedure inspired from circular coinduction technique (Rusu, Lucanu et al., 2015)

- here a more coinductive approach of this procedure
Derivative for RL Formulas

The derivative of a function ... measures the sensitivity to change of a quantity ... which is determined by another quantity ... (Wikipedia)

The derivative of a formula measures the sensitivity to change of a property determined by a transition step.

Semantic: $\varphi_1$ is a derivative of $\varphi$ iff all paths starting from $\varphi_1$ can be extended with one precedent step to paths starting from $\varphi$

If $\varphi$ is an ML formula then

$$\Delta_S(\varphi) = \{ (\exists \text{FreeVars}(\varphi_l, \varphi_r))(\varphi_l \land \varphi_r)^= \land \varphi_r \mid \varphi_l \Rightarrow \varphi_r \in S \}.$$ 

If $\varphi \Rightarrow \varphi'$ is an RL-formula then

$$\Delta_S(\varphi \Rightarrow \varphi') = \{ \varphi_1 \Rightarrow \varphi' \mid \varphi_1 \in \Delta_S(\varphi) \}.$$ 

$\varphi$ is $S$-derivable iff $\forall \varphi_1 \in \Delta_S(\varphi) \varphi_1^=$ is satisfiable.
A Set of Valid Formulas

**STEP**

\[
\text{[impl]} \quad \varphi \Rightarrow \varphi' \quad M \models \varphi \rightarrow \varphi' \quad \text{[der]} \quad \frac{\Delta S(\varphi \Rightarrow \varphi')}{\varphi \Rightarrow \varphi'} \quad \varphi \text{ $S$-derivable}
\]

Assume $S$ finite.

$S \models \nu \text{STEP}$
Trying to Prove an RL Formula

\[
\langle \text{while } (E) \{ B \} P, \sigma \rangle \land \sigma[E] =_{\text{Bool}} \text{false} \Rightarrow \langle P, \sigma \rangle
\]

\[
\langle \text{while } (E) \{ B \} P, \sigma \rangle \land \sigma[E] =_{\text{Bool}} \text{true} \Rightarrow \langle B \text{ while } (E) \{ B \} P, \sigma \rangle
\]

\[
\begin{array}{c}
T_1 \\
T_2
\end{array}
\]

\[
\langle \text{while } (x \neq 0) \{ s = s+x; \ x = x-1; \} \rangle P, x \mapsto a \ s \mapsto 0
\]

\[
\Rightarrow
\]

\[
(\exists b) \langle P, x \mapsto 0 \ s \mapsto b \rangle \land b =_{\text{Int}} \frac{a(a + \text{Int} 1)}{2}
\]
The Proof Tree $T_1$

\[
\langle\text{while }(E) \{ B \} P, \sigma \rangle \land \sigma[E] =_{\text{Bool}} \text{false} \Rightarrow \langle P, \sigma \rangle \\
\langle\text{while }(x \neq 0) \{ s = s+x; \ x = x-1; \} P, x \mapsto a \ s \mapsto 0 \rangle
\]

[impl] \[
\langle P, x \mapsto a \ s \mapsto 0 \rangle \land a =_{\text{Int}} 0 \Rightarrow (\exists b) \langle P, x \mapsto 0 \ s \mapsto b \rangle \land b =_{\text{Int}} \frac{a(a+1)}{2}
\]
The Proof Tree $T_2$ is Infinite . . . 😊

\[
\begin{align*}
\text{while } (x \neq 0) \{ s = s + x; \ x = x - 1; \} & \ P, \\
x \mapsto a - \text{Int} 1 \ s \mapsto a & \implies \\
\quad (\exists b) \langle P, x \mapsto 0 \ s \mapsto b \rangle \land b = \text{Int} \ \frac{a(a + \text{Int} 1)}{2} & \\
\end{align*}
\]

\[
\begin{align*}
\text{while } (x \neq 0) \{ s = s + x; \ x = x - 1; \} & \ P, \\
x \mapsto a \ s \mapsto a & \implies \\
\quad (\exists b) \langle P, x \mapsto 0 \ s \mapsto b \rangle \land b = \text{Int} \ \frac{a(a + \text{Int} 1)}{2} & \\
\end{align*}
\]

\[
\begin{align*}
\text{while } (x \neq 0) \{ s = s + x; \ x = x - 1; \} & \ P, \\
x \mapsto a \ s \mapsto 0 & \implies \\
\quad (\exists b) \langle P, x \mapsto 0 \ s \mapsto b \rangle \land b = \text{Int} \ \frac{a(a + \text{Int} 1)}{2} & \\
\end{align*}
\]
Circular Coinduction Helps Again

\[
\begin{align*}
{\circ}\quad \frac{\Delta \varphi_c \Rightarrow \varphi'_c (\varphi \Rightarrow \varphi')}{\varphi \Rightarrow \varphi'} \quad \varphi \rightarrow (\exists \text{FreeVars}(\varphi_c))\varphi_c
\end{align*}
\]

where \( \varphi_c \Rightarrow \varphi'_c \) is (one of the) initial goal(s)

The initial goal is written in a bit more general form:

\[
\langle \text{while} \ (x \neq 0) \ \{ \ s = s+x; \ x = x-1; \} \ P, x \mapsto a \ s \mapsto s_0 \rangle \\
\Rightarrow \\
(\exists b)\langle P, x \mapsto 0 \ s \mapsto b \rangle \land b = \text{Int} \frac{a(a + \text{Int} 1)}{2} + \text{Int} s_0
\]
The New Proof Tree $T_1$

\[ [\text{impl}] \quad \langle P, x \mapsto a, s \mapsto s_0 \rangle \land a = \text{int } 0 \]
\[ \Rightarrow \]
\[ (\exists b) \langle P, x \mapsto 0, s \mapsto b \rangle \land b = \text{int } \frac{a(a + \text{int } 1)}{2} + \text{int } s_0 \]
The Proof Tree $T_2$ Becomes Finite 😊

**[impl]**

$$\exists b \langle P, x \mapsto 0, s \mapsto b \rangle \land b = \text{int} \left( \frac{(a - \text{int} 1)a}{2} + \text{int} (s_0 + \text{int} a) \right)$$

$$\Rightarrow$$

$$\exists b \langle P, x \mapsto 0, s \mapsto b \rangle \land b = \text{int} \left( \frac{a(a + \text{int} 1)}{2} + \text{int} s_0 \right)$$

**[circ]**

$$\left\langle \begin{array}{l}
\text{while } (x \neq 0) \{ \ s = s + x; \ x = x - 1; \ \} P, \\
 x \mapsto a - \text{int} 1 \ s \mapsto s_0 + \text{int} a
\end{array} \right\rangle \land a \neq \text{int} 0$$

$$\Rightarrow$$

$$\exists b \langle P, x \mapsto 0, s \mapsto b \rangle \land b = \text{int} \left( \frac{a(a + \text{int} 1)}{2} + \text{int} s_0 \right)$$

**[der]**

$$\left\langle \begin{array}{l}
s = s + x; \ x = x - 1; \\
\text{while } (x \neq 0) \{ \ s = s + x; \ x = x - 1; \ \} P, \\
x \mapsto a \ s \mapsto s_0
\end{array} \right\rangle \land a \neq \text{int} 0$$

$$\Rightarrow$$

$$\exists b \langle P, x \mapsto 0, s \mapsto b \rangle \land b = \text{int} \left( \frac{a(a + \text{int} 1)}{2} + \text{int} s_0 \right)$$
A Simple *Sound* Proof System

(Lucanu & Rusu & Arusoaei & Nowak, 2015)

\[
[\text{impl}] \quad \frac{\varphi \Rightarrow \varphi'}{} \quad \models \varphi \rightarrow \varphi' \\
[\text{der}] \quad \frac{\Delta_S(\varphi \Rightarrow \varphi')}{} \quad \varphi \text{ is } S\text{-derivable}
\]

\[
[\text{circ}] \quad \frac{\Delta_{\varphi_c \Rightarrow \varphi'_c}(\varphi \Rightarrow \varphi')}{} \quad \models \varphi \rightarrow (\exists \text{var}(\varphi_c))\varphi_c, \varphi_c \Rightarrow \varphi'_c \in G
\]

where $G$ is the set of initial goals (reachability rules to be proved)

- a coinduction-based proof for soundness can be found in the techn. rep. [https://hal.inria.fr/hal-00766220v8](https://hal.inria.fr/hal-00766220v8)
- no completeness claim (vs. 8-rule complete system in [Roșu et al., RTA 2014])
- easy to be implemented with symbolic execution
Plan

1. Induction and Coinduction on Powerset

2. (Co)Inductive Sets Defined by Ground Inference Systems

3. Application to Program Verification

4. Conclusion
Conclusion

- circular coinduction is potentially a strong mechanism for proving behavioural properties expressed by reachability formulas
- already used as an inference rule in
  - proof systems for various versions of reachability logic (Roşu, Ştefănescu, Ciobâcă, Moore)
  - proof systems for program equivalence (Rusu, Ciobâcă, Roşu, Lucanu)
  - Coq verification procedure for one path reachability logic (Moore, Roşu)
- however, its practical potential not fully exploited
- there are efforts to incorporate it in various tools from K Framework