On the exquisite pleasure of doing coinduction and corecursion in Isabelle

Andrei Popescu

University of Sheffield

Special Session on Proof Assistants
CALCO & MFPS, 21 June 2023

On the exquisite pleasure of doing coinduction and corecursion in Isabelle

Sheffield . . . ✈ . . . Bloomington, Indiana

Andrei Popescu

University of Sheffield

Special Session on Proof Assistants
CALCO & MFPS, 21 June 2023

On the exquisite pleasure of doing coinduction and corecursion in Isabelle

Sheffield . . . ✈ . . . Bloomington, Indiana

On the terrible pain of doing coinduction and corecursion in Isabelle

Andrei Popescu

University of Sheffield

Special Session on Proof Assistants
CALCO & MFPS, 21 June 2023

Inductive definition example: the sublist relation

$subl : \mathsf{List}(A) \to \mathsf{List}(A) \to \mathsf{Bool}$ defined inductively by the following rules:

$$\frac{\cdot}{subl\ []\ as}\ \text{(Nil)} \qquad\qquad \frac{subl\ as\ as'}{subl\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subl\ as\ as'}{subl\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

Inductive definition example: the sublist relation

$subl$: List(A) → List(A) → Bool defined inductively by the following rules:

$$\frac{\cdot}{subl\ []\ as}\ \text{(Nil)} \qquad \frac{subl\ as\ as'}{subl\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subl\ as\ as'}{subl\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

The inductive interpretation means:

1. smallest relation closed under the above rules

Inductive definition example: the sublist relation

$subl$: List$(A) \to$ List$(A) \to$ Bool defined inductively by the following rules:

$$\frac{\cdot}{subl\ []\ as}\ \text{(Nil)} \qquad\qquad \frac{subl\ as\ as'}{subl\ as\ (a\#as')}\ \text{(ConsR)}$$

$$\frac{subl\ as\ as'}{subl\ (a\#as)\ (a\#as')}\ \text{(Cons)}$$

The inductive interpretation means:

1. smallest relation closed under the above rules
2. relation provable by the above rules using finite proof trees

Coiductive definition example: the sub-lazy-list relation[1]

Given a set A, let LazyList(A) be the set of "lazy lists" (finite or infinite lists) with elements in A – they have the form $[a_1, a_2, \ldots, a_n]$ or $[a_1, a_2, \ldots]$. We write $a\#as$ for the lazy list obtained by consing a to as, and $bs \mathbin{@} as$ for the concatenation of a (finite) list bs and a lazy list as. $\mathit{subll} : \mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}$ is defined coinductively by the following rules:

$$\frac{\cdot}{\mathit{subll}\ []\ as}\ \text{(Nil)} \qquad \frac{\mathit{subll}\ as\ as'}{\mathit{subll}\ (a\#as)\ (bs \mathbin{@} (a\#as'))}\ \text{(Cons)}$$

[1] These rules are a modified version of what I showed at the conference. I thank Paul Levy for pointing out that my original definition was not correctly capturing sub-lazy-lists – which is a timely illustration of what Assia Mahboubi mentioned in her talk: that formality/rigour does not guarantee correctness.

Coiductive definition example: the sub-lazy-list relation[1]

Given a set A, let LazyList(A) be the set of "lazy lists" (finite or infinite lists) with elements in A – they have the form $[a_1, a_2, \ldots, a_n]$ or $[a_1, a_2, \ldots]$. We write $a\#as$ for the lazy list obtained by consing a to as, and $bs \mathbin{@} as$ for the concatenation of a (finite) list bs and a lazy list as. $subll : \mathsf{LazyList}(A) \to \mathsf{LazyList}(A) \to \mathsf{Bool}$ is defined <u>coinductively</u> by the following rules:

$$\frac{\cdot}{subll\ []\ as}\ (\text{Nil}) \qquad \frac{subll\ as\ as'}{subll\ (a\#as)\ (bs \mathbin{@} (a\#as'))}\ (\text{Cons})$$

The coinductive interpretation means:

1. <u>largest</u> relation <u>consistent with</u> (i.e., backwards-closed under) the above rules
2. relation provable by the above rules using <u>finite or infinite proof trees</u>

[1] These rules are a modified version of what I showed at the conference. I thank Paul Levy for pointing out that my original definition was not correctly capturing sub-lazy-lists – which is a timely illustration of what Assia Mahboubi mentioned in her talk: that formality/rigour does not guarantee correctness.

Fixpoints versus proof trees

The semantic foundations for induction and coinduction are perfectly dual – via Knaster-Tarski:

- induction: least (pre-)fixpoint
- coinduction: greatest (post-)fipoint

But they have quite different intuitions:

- induction – whatever can be proved using a finite number of rule applications
- coinduction – whatever can be proved using a finite or (countably) infinite number of rule applications

Fixpoints versus proof trees

The semantic foundations for induction and coinduction are perfectly dual – via Knaster-Tarski:

- induction: least (pre-)fixpoint
- coinduction: greatest (post-)fipoint

But they have quite different intuitions:

- induction – whatever can be proved using a finite number of rule applications
- coinduction – whatever can be proved using a finite or (countably) infinite number of rule applications

Will use Isabelle to prove the equivalence of the two views

Links to the Isabelle theories used in the demo:
`https://www.andreipopescu.uk/MFPS_CALCO_2023/Isabelle_files.zip`

An (obviously incomplete ☺) list of good sources of learning about induction and coinduction

Jacobs and Rutten 1997. A tutorial on coalgebra and coinduction

Paulson 2000. A fixedpoint approach to (co)inductive and (co)datatype definitions

Pierce 2002. Types and Programming Languages (Section 21.1. Induction and Coinduction)

Bertot 2008. CoInduction in Coq

Blanchette, Popescu & Traytel 2015. Witnessing (Co)datatypes

Kozen & Silva 2017. Practical coinduction

Chlipala 2019. Certified Programming with Dependent Types (Chapter 5. Infinite data and proofs)

Isabelle's (co)induction and (co)recursion infrastructure

(Co)inductive predicates, initial datatype package

- Paulson 1994. A Fixedpoint Approach to Implementing (Co)Inductive Definitions.
- Berghofer & Wenzel 1999. Inductive Datatypes in HOL - Lessons Learned in Formal-Logic Engineering.

Compositional (co)datatypes

- Traytel, Popescu, Blanchette 2012. Foundational, Compositional (Co)datatypes for Higher-Order Logic
- Blanchette, Hölzl, Lochbihler, Panny, Popescu, Traytel 2014. Truly Modular (Co)datatypes for Isabelle/HOL
- Blanchette, Meier, Popescu, Traytel 2017. Foundational Nonuniform (Co)datatypes for Higher-Order Logic.

Expressive corecursion

- Blanchette, Popescu, Traytel 2015. Foundational extensible corecursion: a proof assistant perspective.
- Blanchette, Bouzy, Lochbihler, Popescu, Traytel 2017. Friends with Benefits – Implementing Corecursion in Foundational Proof Assistants.

(Co)datatypes with bindings

- Blanchette, Gheri, Popescu, Traytel 2019. Bindings as bounded natural functors.