How to Reason Informally Coinductively

Anton Setzer

Swansea University

Swansea PCV Seminar, 16 June 2015

With contributions from Peter Hancock, Thorsten Altenkirch, Andreas Abel, Brigitte Pientka and David Thibodeau.

Goal

[1]

Inductive Definition	Coinductive Definition
Determined by Introduction	?
Iteration	?
Primitive Recursion	?
Pattern matching	?
Induction	?
Induction-Hypothesis	?

[1] Part of this table is due to Peter Hancock, see acknowledgements at the end.

Introduction/Elimination of Inductive/Coinductive Sets

- Introduction rules for Natural numbers means that we have

$$0 \in \mathbb{N}$$
$$\mathrm{S} : \mathbb{N} \to \mathbb{N}$$

- Dually, coinductive sets are given by their elimination rules i.e. by **observations**.

  As an example we consider Stream:

$$\begin{array}{lcl} \mathrm{head} & : & \mathrm{Stream} \to \mathbb{N} \\ \mathrm{tail} & : & \mathrm{Stream} \to \mathrm{Stream} \end{array}$$

Duality

Inductive Definition	Coinductive Definition
Determined by Introduction	Determined by Observation
Iteration	?
Primitive Recursion	?
Pattern matching	?
Induction	?
Induction-Hypothesis	?

Unique Iteration

- That $(\mathbb{N}, 0, \mathrm{S})$ are minimal can be given by:
  - Assume another $\mathbb{N}$-structure (X, z, s), i.e.

$$\begin{array}{l} z \in X \\ s : X \to X \end{array}$$

  - Then there exist a **unique homomorphism** $g : (\mathbb{N}, 0, \mathrm{S}) \to (X, z, s)$:

$$\begin{array}{lcl} g : \mathbb{N} \to X & & \\ g(0) & = & z \\ g(\mathrm{S}(n)) & = & s(g(n)) \end{array}$$

  - This means we can define uniquely

$$\begin{array}{lcll} g : \mathbb{N} \to X & & & \\ g(0) & = & x & \text{for some } x \in X \\ g(\mathrm{S}(n)) & = & x' & \text{for some } x' \in X \text{ depending on } g(n) \end{array}$$

Unique Coiteration

- Dually, that $(\mathrm{Stream}, \mathrm{head}, \mathrm{tail})$ is maximal can be given by:
  - Assume another Stream-structure (X, h, t):

$$\begin{array}{lcl} h & : & X \to \mathbb{N} \\ t & : & X \to X \end{array}$$

  - Then there exist a **unique homomorphism** $g : (X, h, t) \to (\mathrm{Stream}, \mathrm{head}, \mathrm{tail})$:

$$\begin{array}{lcl} \multicolumn{3}{l}{g : X \to \mathrm{Stream}} \\ \mathrm{head}(g(x)) & = & h(x) \\ \mathrm{tail}(g(x)) & = & g(t(x)) \end{array}$$

- Means we can define uniquely

$$\begin{array}{lcll} \multicolumn{4}{l}{g : X \to \mathrm{Stream}} \\ \mathrm{head}(g(x)) & = & n & \text{for some } n \in \mathbb{N} \text{ depending on } x \\ \mathrm{tail}(g(x)) & = & g(x') & \text{for some } x' \in X \text{ depending on } x \end{array}$$

Comparison

- When using iteration the instance of g we can use is restricted, but we can apply an arbitrary function to it.
- When using coiteration we can choose which instance of g we want, but can use it only directly.

Duality

Inductive Definition	Coinductive Definition
Determined by Introduction	Determined by Observation
Iteration	Coiteration
Primitive Recursion	?
Pattern matching	?
Induction	?
Induction-Hypothesis	?

Unique Primitive Recursion

- From unique iteration we can derive principle of **unique primitive recursion**
  - We can define uniquely

$$
\begin{array}{lcll}
g : \mathbb{N} \to X & & & \\
g(0) & = & x & \text{for some } x \in X \\
g(\mathrm{S}(n)) & = & x' & \text{for some } x' \in X \text{ depending on } n,\ g(n)
\end{array}
$$

- Primitive **pattern matching**.

Unique Primitive Corecursion

- From unique coiteration we can derive principle of **unique primitive corecursion**
  - We can define uniquely

$$
\begin{array}{lcl}
g : X \to \mathrm{Stream} & & \\
\mathrm{head}(g(x)) & = & n \text{ for some } n \in \mathbb{N} \text{ depending on } x \\
\mathrm{tail}(g(x))) & = & g(x') \text{ for some } x' \in X \text{ depending on } x \\
 & & \text{or} \\
 & = & s \text{ for some } s \in \mathrm{Stream} \text{ depending on } x
\end{array}
$$

  - **Note:** No application of a function to $g(x')$ allowed.
  - Primitive **copattern matching**.

Example

$$\begin{array}{lcl} s \in \mathrm{Stream} & & \\ \mathrm{head}(s) & = & 0 \\ \mathrm{tail}(s) & = & s \end{array}$$

$$\begin{array}{lcl} s' : \mathbb{N} \to \mathrm{Stream} & & \\ \mathrm{head}(s'(n)) & = & 0 \\ \mathrm{tail}(s'(n)) & = & s'(n+1) \end{array}$$

$$\begin{array}{lcl} \mathrm{cons} : (\mathbb{N} \times \mathrm{Stream}) \to \mathrm{Stream} & & \\ \mathrm{head}(\mathrm{cons}(n,s)) & = & n \\ \mathrm{tail}(\mathrm{cons}(n,s)) & = & s \end{array}$$

Duality

Inductive Definition	Coinductive Definition
Determined by Introduction	Determined by Observation
Iteration	Coiteration
Primitive Recursion	Primitive Corecursion
Pattern matching	Copattern matching
Induction	?
Induction-Hypothesis	?

Induction

- From unique iteration one can derive principle of **induction**:

  We can prove $\forall n \in \mathbb{N}.\varphi(n)$ by proving
  $\varphi(0)$
  $\forall n \in \mathbb{N}.\varphi(n) \to \varphi(\mathrm{S}(n))$

- Using induction we can prove (assuming extensionality of functions) uniqueness of iteration and primitive recursion.

Equivalence

Theorem

Let $(\mathbb{N}, 0, \mathrm{S})$ *be an* $\mathbb{N}$*-algebra. The following is equivalent*

1. *The principle of unique iteration.*
2. *The principle of unique primitive recursion.*
3. *The principle of iteration* $+$ *induction.*
4. *The principle of primitive recursion* $+$ *induction.*

Coinduction

- Uniqueness in coiteration is equivalent to the principle:
  Bisimulation implies equality
- Bisimulation on Stream is the largest relation $\sim$ on Stream s.t.

$$s \sim s' \to \mathrm{head}(s) = \mathrm{head}(s') \land \mathrm{tail}(s) \sim \mathrm{tail}(s')$$

- Largest can be expressed as $\sim$ being an indexed coinductively defined set.
- Primitive corecursion over $\sim$ means:
  We can prove

$$\forall s, s'.X(s, s') \to s \sim s'$$

  by showing

$$\begin{array}{rcl} X(s, s') & \to & \mathrm{head}(s) = \mathrm{head}(s') \\ X(s, s') & \to & X(\mathrm{tail}(s), \mathrm{tail}(s')) \lor \mathrm{tail}(s) \sim \mathrm{tail}(s') \end{array}$$

Coinduction

- Combining
  - bisimulation implies equality
  - bisimulation can be shown corecursively

  we obtain the following principle of **coinduction**

Schema of Coinduction

- We can prove

$$\forall s, s'.X(s, s') \to s = s'$$

by showing

$$\begin{array}{lcl} \forall s, s'.X(s, s') & \to & \mathrm{head}(s) = \mathrm{head}(s') \\ \forall s, s'.X(s, s') & \to & \mathrm{tail}(s) = \mathrm{tail}(s') \end{array}$$

where $\mathrm{tail}(s) = \mathrm{tail}(s')$ can be derived
  - directly or
  - from a proof of

$$X(\mathrm{tail}(s), \mathrm{tail}(s'))$$

  invoking the **co-induction-hypothesis**

$$X(\mathrm{tail}(s), \mathrm{tail}(s')) \to \mathrm{tail}(s) = \mathrm{tail}(s')$$

- **Note:** Only direct use of co-IH allowed.

Indexed Coinduction

- For using coinduction, one typically wants to show for some $f, g : X \to \mathrm{Stream}$
$$\forall x \in X . f(x) = g(x)$$
- Using $X(s, s') = \{x \mid f(x) = s \wedge g(x) = s'\}$ we obtain the principle of **indexed coinduction**

Schema Indexed Coinduction

- We can prove
$$\forall x \in X. f(x) = g(x)$$
by showing
$$\begin{array}{rcl} \forall x \in X.\mathrm{head}(f(x)) & = & \mathrm{head}(g(x)) \\ \forall x \in X.\mathrm{tail}(f(x)) & = & \mathrm{tail}(g(x)) \end{array}$$
where $\mathrm{tail}(f(x)) = \mathrm{tail}(g(x))$ can be derived
  - directly or
  - by
$$\mathrm{tail}(f(x)) = f(x') \qquad \mathrm{tail}(g(x)) = g(x')$$
and using the **co-induction-hypothesis**
$$f(x') = g(x')$$
- Again **only direct use of co-IH** allowed
(otherwise you can derive $\mathrm{tail}(f(x)) = \mathrm{tail}(g(x))$ from $f(x) = g(x)$).
- In fact the above is the same as uniqueness of corecursion.

Equivalence

Theorem

Let $(\mathrm{Stream}, \mathrm{head}, \mathrm{tail})$ *be a* Stream-*coalgebra. The following is equivalent*

1. *The principle of unique coiteration.*
2. *The principle of unique primitive corecursion.*
3. *The principle of coiteration + coinduction*
4. *The principle of primitive corecursion + coinduction*
5. *The principle of coiteration + indexed coinduction.*
6. *The principle of primitive corecursion + indexed coinduction.*

Example

- Remember

$$\begin{array}{rcl} \mathrm{head}(s) & = & 0 \\ \mathrm{tail}\ \ (s) & = & s \end{array} \qquad \begin{array}{rcl} \mathrm{head}(s'(n)) & = & 0 \\ \mathrm{tail}\ \ (s'(n)) & = & s'(n+1) \end{array}$$

- We show $\forall n \in \mathbb{N}.s = s'(n)$ by indexed coinduction:
  - $\mathrm{head}(s) = 0 = \mathrm{head}(s'(n))$.
  - $\mathrm{tail}(s) = s \stackrel{\text{co-IH}}{=} s'(n+1) = \mathrm{tail}(s'(n))$.

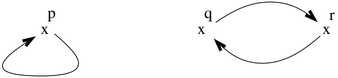
Example

$$\begin{array}{rcl} \mathrm{head}(s) & = & 0 \\ \mathrm{tail}\ (s) & = & s \end{array}$$

- We show $s = \mathrm{cons}(0, s)$ by indexed coinduction:
  - $\mathrm{head}(s) = 0 = \mathrm{head}(\mathrm{cons}(0, s))$.
  - $\mathrm{tail}(s) = s = \mathrm{tail}(\mathrm{cons}(0, s))$
    (no use of co-IH).

Proofs of Other Bisimilarity Relations

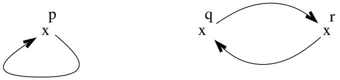
- The above can be used as well for proving other bisimilarity relations.
- Consider the following (unlabelled) transition system:



- Bisimilarity is the final coalgebra

$$\begin{aligned} p \sim q \to (\forall p'.p \longrightarrow p' \\ \to \exists q'.q \longrightarrow q' \wedge p' \sim q') \\ \wedge \cdots \text{symmetric case} \cdots \} \end{aligned}$$

Proof using the Definition of $\sim$



- We show $p \sim q \wedge p \sim r$ by indexed coinduction:
- **Coinduction step for** $p \sim q$:
  - Assume $p \longrightarrow p'$. Then $p' = p$.
    We have $q \longrightarrow r$ and by co-IH $p \sim r$.
  - Assume $q \longrightarrow q'$. Then $q' = r$.
    We have $p \longrightarrow p$ and by co-IH $p \sim r$.
- **Coinduction step for** $p \sim r$:
  - Assume $p \longrightarrow p'$. Then $p' = p$.
    We have $r \longrightarrow q$ and by co-IH $p \sim q$.
  - Assume $r \longrightarrow r'$. Then $r' = q$.
    We have $p \longrightarrow p$ and by co-IH $p \sim q$.

General Coinduction

- Schwichtenberg: I have done lots of coinductive proofs but it was never a proof of an equality.
- Answer:
  - What happens is that the predicate proved was defined coinductively.
  - The corecursion principle for this predicate corresponds to coinductive proofs of this formula.
  - Again the corecursion hypothesis forms the coinduction principles.
- It is necessary to do it like this because coinduction/corecursion is an **introduction principle**, **not** an **elimination principle**.

Conclusion

Inductive Definition	Coinductive Definition
Determined by Introduction	Determined by Observation
Iteration	Coiteration
Primitive Recursion	Primitive Corecursion
Pattern matching	Copattern matching
Induction	Coinduction (?)
Induction-Hypothesis	Coinduction-Hypothesis

Acknowledgements

- To look at iteration, recursion, induction in parallel with coiteration, corecursion, coinduction I learned from Peter Hancock, although we didn't resolve in our discussions what coinduction is and what the precise formulation of corecursion would be.
- How to derive from iteration recursion I learned from Thorsten Altenkirch, however that seems to be a well-known fact.

Bibliography

- Anton Setzer, Andreas Abel, Brigitte Pientka and David Thibodeau: Unnesting of Copatterns. In Gilles Dowek (Ed): Rewriting and Typed Lambda Calculi. Proceedings RTA-TLCA 2014. LNCS 8560, 2014, pp. 31 - 45. Doi 10.1007/978-3-319-08918-8_3. Bibtex.
- Andreas Abel, Brigitte Pientka, David Thibodeau and Anton Setzer: Copatterns: programming infinite structures by observations. Proceedings of POPL 2013, 2013, pp. 27 - 38. Doi 10.1145/2429069.2429075. Bibtex.
- Anton Setzer: Coalgebras as Types determined by their Elimination Rules. In: Peter Dybjer, Sten Lindström, Erik Palmgren, Göran Sundholm: Epistemology versus ontology: Essays on the foundations of mathematics in honour of Per Martin-Löf. Springer, 2012, pp. 351 – 369, Doi: 10.1007/978-94-007-4435-6_16. Bibtex

Appendix

Difficulty defining Pred Using Iteration

- Using iteration pred, the inverse of $0, \mathrm{S}$ is inefficient:

$$
\begin{array}{lcl}
\multicolumn{3}{l}{\mathrm{pred} : \mathbb{N} \to \{-1\} \cup \mathbb{N}} \\
\mathrm{pred}(0) & = & -1 \\
\mathrm{pred}(\mathrm{S}(n)) & = & \mathrm{S}'(\mathrm{pred}(n))
\end{array}
$$

where

$$
\begin{array}{lcl}
\multicolumn{3}{l}{\mathrm{S}' : \{-1\} \cup \mathbb{N} \to \mathbb{N}} \\
\mathrm{S}'(-1) & = & 0 \\
\mathrm{S}(n) & = & \mathrm{S}(n) \text{ if } n \in \mathbb{N}
\end{array}
$$

$$
\begin{array}{lclcl}
\mathrm{pred}(2) & = & \mathrm{S}'(\mathrm{pred}(1)) & = & \mathrm{S}'(\mathrm{S}'(\mathrm{pred}(0))) \\
 & = & \mathrm{S}'(\mathrm{S}'(-1)) & = & \mathrm{S}'(0) = \mathrm{S}(0) = 1
\end{array}
$$

Difficulty defining Cons Using Coiteration

- Using coiteration cons, the inverse of $\mathrm{head}, \mathrm{tail}$ is difficult to define

$$
\begin{array}{lcl}
\multicolumn{3}{l}{\mathrm{cons} : (\mathbb{N} \times \mathrm{Stream}) \to \mathrm{Stream}} \\
\mathrm{head}(\mathrm{cons}(n, s)) & = & n \\
\mathrm{tail}(\mathrm{cons}(n, s)) & = & \mathrm{cons}(\mathrm{head}(s), \mathrm{tail}(s))
\end{array}
$$

$$e.g.\mathrm{tail}(\mathrm{tail}(\mathrm{cons}(n, s))) = \mathrm{cons}(\mathrm{head}(\mathrm{tail}(s)), \mathrm{tail}(\mathrm{tail}(s)))$$