How to Reason Informally Coinductively

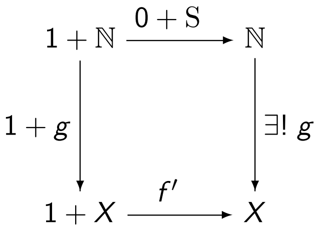
Anton Setzer

Swansea University, Swansea UK

Logic Colloquium 2012 (Manchester), 14 July 2012

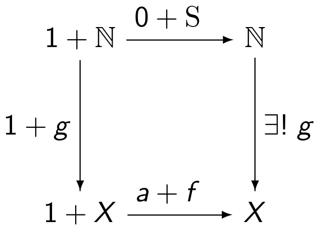
$\mathbb{N}$ as an Initial Algebra

- $\mathbb{N}$ is initial algebra of the functor $1 + \_$



f' can be decomposed as $f' = a + f$

$\mathbb{N}$ as an Initial Algebra



Existence of g corresponds to iteration:

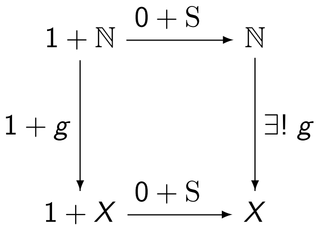
$$
\begin{aligned}
g(0) &= g((0+\mathrm{S})(\mathrm{inl})) &&= (a+f)((1+g)(\mathrm{inl})) \\
& &&= a \\
g(\mathrm{S}(n)) &= g((0+\mathrm{S})(\mathrm{inr}(n))) &&= (a+f)((1+g)(\mathrm{inr}(n))) \\
& &&= f(g(n)) \\
g(\mathrm{S}^n(0)) &= f^n(a)
\end{aligned}
$$

Proof by Induction

- We can derive using uniqueness of g the induction principle from it:
  - Assume $\varphi(0)$, $\forall n \in \mathbb{N}.\varphi(n) \to \varphi(\mathrm{S}(n))$. Then $\forall n \in \mathbb{N}.\varphi(n)$ hold.
- The induction principle is derived by taking

$$X := \{n \in \mathbb{N} \mid \varphi(n)\}$$

in



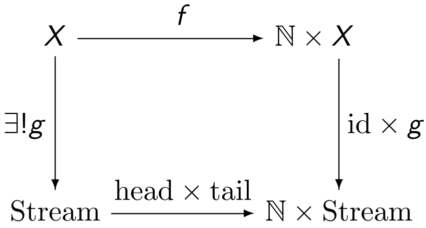
and then deriving that g as above is the identity.

Proof by Induction

- Actual proofs by induction are carried out as follows:
  Show $\forall n \in \mathbb{N}.\varphi(n)$ by Induction on n:
  - **Base case:**
    Prove $\varphi(0)$.
  - **Induction step:**
    Assume $n \in \mathbb{N}$. Prove $\varphi(\mathrm{S}(n))$ by using the IH $\varphi(n)$.
- So we don't define a set $X := \{n \in \mathbb{N}.\varphi(n)\}$ and show it is closed under 0, S, but reason using the schema of induction.
- We can use the IH in order to prove the proof obligation $\varphi(\mathrm{S}(n))$ in the induction step.
- **Goal:** Reason in a similar informal way about coalgebras, without having to construct the "X".

Streams

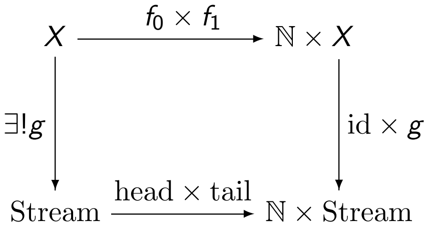
- Dual of $+$ is $\times$, so we use for clarity a functor using product rather than disjoint union:
- Stream is the final coalgebra of $\mathbb{N} \times \_$



- We can decompose f as

$$f \quad = \quad f_0 \times f_1$$

Streams



g above is uniquely defined by

$$
\begin{array}{lcllcl}
\mathrm{head}(g(x)) & = & \pi_0((\mathrm{head} \times \mathrm{tail})(g(x))) & & \\
 & = & \pi_0((\mathrm{id} \times g)(f_0 \times f_1)(x)) & = & f_0(x) \\
\mathrm{tail}(g(x)) & = & \pi_1((\mathrm{head} \times \mathrm{tail})(g(x))) & & \\
 & = & \pi_1((\mathrm{id} \times g)(f_0 \times f_1)(x)) & = & g(f_1(x))
\end{array}
$$

Guarded Recursion

- We had:

$$\begin{array}{lcl} \mathrm{head} & (g(x)) & = & f_0(x) \\ \mathrm{tail} & (g(x)) & = & g(f_1(x)) \end{array}$$

- By choosing f_0, f_1 we can define $g : X \to \mathrm{Stream}$ s.t.

$$\begin{array}{lcll} \mathrm{head} & (g(x)) & = & n & \text{depending on } x \\ \mathrm{tail} & (g(x)) & = & g(x') & \text{some } x' \in X \text{ depending on } x \end{array}$$

  So full recursion allowed after applying destructor to g.
- Guarded recursion in this form is exactly the same as the existence of g in the categorical diagram.

Guarded Recursion

- Generalisation: We can define g such that

$$
\begin{array}{llcll}
\mathrm{head} & (g(x)) & = & n & \text{depending on } x \\
\mathrm{tail} & (g(x)) & = & g(x') & \text{some } x' \in X \text{ depending on } x \\
 & & & \text{or} & \\
 & & = & s & \text{some } s \in \mathrm{Stream} \text{ depending on } x
\end{array}
$$

Examples

- We can define

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

Note: cons defined by guarded recursion

$$
\begin{array}{llcl}
\mathrm{inc} & & : & \mathbb{N} \to \mathrm{Stream} \\
\mathrm{head} & (\mathrm{inc}(n)) & = & n \\
\mathrm{tail} & (\mathrm{inc}(n)) & = & \mathrm{inc}(n + 1)
\end{array}
$$

Examples

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

- We want to show that inc, inc' are bisimilar and therefore, because Stream is a final (and not weakly final coalgebra) equal.

Bisimilarity

- Bisimilarity $\sim$ is the largest fixed point and therefore a dependent final coalgebra:

$$
\begin{array}{ccc}
\varphi(s, s') & \xrightarrow{\ f\ } & \mathrm{head}(s) = \mathrm{head}(s') \wedge \varphi(\mathrm{tail}(s), \mathrm{tail}(s')) \\
{\scriptstyle \exists! g}\downarrow & & \downarrow{\scriptstyle \mathrm{id} \wedge g} \\
s \sim s' & \xrightarrow{\ \mathrm{elim}_{\sim}\ } & \mathrm{head}(s) = \mathrm{head}(s') \wedge \mathrm{tail}(s) \sim \mathrm{tail}(s')
\end{array}
$$

- As a proof principle this reads:
  - Assume
    $\forall s, s' \in \mathrm{Stream}.\varphi(s, s') \to \mathrm{head}(s) = \mathrm{head}(s') \wedge \varphi(\mathrm{tail}(s), \mathrm{tail}(s'))$.
  - Then $\forall s, s' \in \mathrm{Stream}.\varphi(s, s') \to s \sim s'$.
    (And then $\forall s, s' \in \mathrm{Stream}.\varphi(s, s') \to s = s'$).

Generalisation

- We can generalise this to
  - Assume

$$\forall s, s' \in \mathrm{Stream}.\varphi(s, s') \rightarrow \mathrm{head}(s) = \mathrm{head}(s') \wedge (\varphi(\mathrm{tail}(s), \mathrm{tail}(s')) \vee \mathrm{tail}(s) \sim \mathrm{tail}(s'))$$

  - Then $\forall s, s' \in \mathrm{Stream}.\varphi(s, s') \rightarrow s \sim s'$.
- The coinduction step requires us to prove, assuming $\varphi(s, s')$

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

  and we can use the co-IH

$$\varphi(\mathrm{tail}(s), \mathrm{tail}(s')) \rightarrow \mathrm{tail}(s) \sim \mathrm{tail}(s')$$

  in order to prove the right conjunct.
- Similar to induction where we could use the IH as an additional assumption in the proof obligation of the induction step.

Example Proof by Coinduction

- We show $\forall n \in \mathbb{N}.\mathrm{inc}(n) \sim \mathrm{inc}'(n) \wedge \mathrm{inc}(n) \sim \mathrm{inc}''(n)$.
- Formally we can argue by using

$$\varphi(s, s') := \exists n \in \mathbb{N}.s = \mathrm{inc}(n) \wedge s' \in \{\mathrm{inc}'(n), inc''(n)\}$$

and then showing

$$\begin{aligned}\forall s, s' \in \mathrm{Stream}.\varphi(s, s') \to \mathrm{head}(s) = \mathrm{head}(s') \wedge \\ (\varphi(\mathrm{tail}(s), \mathrm{tail}(s')) \vee \mathrm{tail}(s) \sim \mathrm{tail}(s'))\end{aligned}$$

Informal Proof by Coinduction

- We show $\mathrm{inc}(n) \sim \mathrm{inc}'(n) \land \mathrm{inc}(n) \sim \mathrm{inc}''(n)$ by coinduction on $\sim$:
- **Coinduction step** for $\mathrm{inc}(n) \sim \mathrm{inc}'(n)$:
  We need to prove

$$\mathrm{head}(\mathrm{inc}(n)) = \mathrm{head}(\mathrm{inc}'(n)) \land \mathrm{tail}(\mathrm{inc}(n)) \sim \mathrm{tail}(\mathrm{inc}'(n))$$

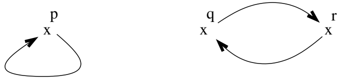
  and can use the co-IH for the second conjunct.
  Follows by:

$$\begin{array}{llclcl} \mathrm{head} & (\mathrm{inc}(n)) & = & n & = & \mathrm{head}(\mathrm{inc}'(n)) \\ \mathrm{tail} & (\mathrm{inc}(n)) & = & \mathrm{inc}(n+1) & \stackrel{co-IH}{\sim} & \mathrm{inc}''(n+1) = \mathrm{tail}(\mathrm{inc}'(n)) \end{array}$$

- **Coinduction step** for $\mathrm{inc}(n) \sim \mathrm{inc}''(n)$: Similarly.

Bisimilarity

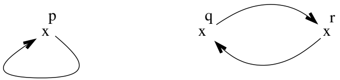
- Consider the following (unlabelled) transition system:



- 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 $p \sim q \land p \sim r$ by coinduction:
- **Coinduction step for** $p \sim q$:
  - Every transition of p is simulated by a transition of q:
    Only transition of p is $p \longrightarrow p$.
    We choose for q transition $q \longrightarrow r$,
    and get by co-IH $p \sim r$.
  - Every transition of q is simulated by a transition of p:
    Only transition of q is $q \longrightarrow r$.
    We choose for p transition $p \longrightarrow p$,
    and get by co-IH $p \sim r$.
- **Coinduction step for** $p \sim r$: Similar.

Conclusion

- Principle of induction is well established and makes proofs much easier.
- In theoretical computer science coinductive principles occur frequently.
  - Main reason: interactive programs running continuously in various frameworks (imperative, object-oriented, process-calculi)
- Coalgebras as being defined by their eliminators rather than infinite applications of constructors makes clear when recursive calls are allowed.
- Proofs by coinduction in the above situation can be carried out similarly as proofs by induction.
- Main difficulty: when are we allowed to apply co-IH?
  - In the corecursion step we have a proof obligation, and can use the co-IH to prove it.