Pattern and Copattern matching

Anton Setzer

Swansea University, Swansea UK

Leeds Logic Seminar, 13 May 2015

Iteration, Recursion, Induction

Coiteration, Corecursion

Bisimilarity and Coinduction

Proofs by Coinduction of Bisimilarity in Transition Systems

Mixed Patterns and Copatterns

Unnesting of Pattern/Copattern Matching

Iteration, Recursion, Induction

Coiteration, Corecursion

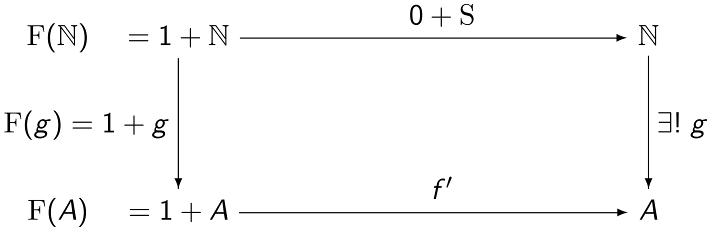
Bisimilarity and Coinduction

Proofs by Coinduction of Bisimilarity in Transition Systems

Mixed Patterns and Copatterns

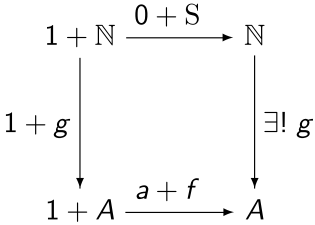
Unnesting of Pattern/Copattern Matching

$\mathbb{N}$ as an Initial Algebra

- $\mathbb{N}$ is initial algebra of the functor $\mathrm{F}(X) = 1 + X$
- 

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

Unique Iteration



Unique existence of g means **unique iteration**:

Given $a : A$ and $f : A \to A$ there exists a unique

$$
\begin{array}{lcl}
g : \mathbb{N} \to A & & \\
g\ 0 & = & a \\
g\ (\mathrm{S}\ n) & = & f\ (g\ n) \\
\text{i.e} & & \\
g\ (\mathrm{S}^n\ 0) & = & f^n\ a
\end{array}
$$

Unique Recursion

- From the principle of unique iteration we can prove the principle of unique (primitive) recursion:

  Given $a : A$ and $f : \mathbb{N} \to A \to A$ there exists a unique

$$
\begin{array}{lcl}
g : \mathbb{N} \to A & & \\
g\ 0 & = & a \\
g\ (\mathrm{S}\ n) & = & f\ n\ (g\ n)
\end{array}
$$

Induction

- From the principle of unique iteration we can prove the principle of induction:

  Assume $A : \mathbb{N} \to \mathrm{Set}$, $a : A\ 0$ and $f : (n : \mathbb{N}) \to A\ n \to A\ (\mathrm{S}\ n)$

  There exists a unique

$$
\begin{array}{lcl}
\multicolumn{3}{l}{g : (n : \mathbb{N}) \to A\ n} \\
g\ 0 & = & a \\
g\ (\mathrm{S}\ n) & = & f\ n\ (g\ n)
\end{array}
$$

- Using induction we can prove that if we have two solutions for a iteration or recursion principle, they are pointwise equal, i.e. uniqueness of iteration and recursion.

Pattern Matching

- The above means that we can define

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

  where in the second case we can use the **recursion hypothesis** or **induction hypothesis** $g\ n$.
- This means we can define $g\ n$ by **pattern matching** on $n : \mathbb{N}$.

Iteration, Recursion, Induction

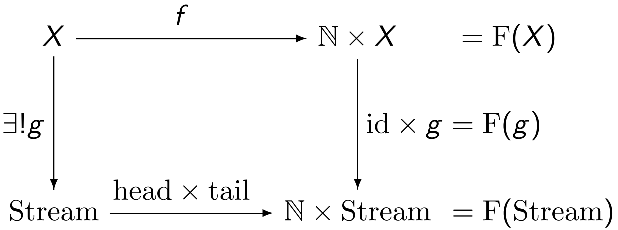
Theorem

Assume $\mathbb{N} : \mathrm{Set}$, $0 : \mathbb{N}$, $\mathrm{S} : \mathbb{N} \to \mathbb{N}$.
The following are equivalent

- *The principle of unique iteration.*
- *The principle of unique recursion.*
- *The principle of unique induction.*
- *The principle of induction.*

Streams as a Final Coalgebra

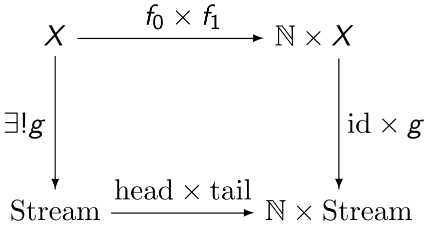
- Dual of $+$ is $\times$, so we use for clarity a functor using product rather than disjoint union:
- Stream is the final coalgebra of $\mathrm{F}(X) = \mathbb{N} \times X$



- We can decompose f as

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

Unique Coiteration



This corresponds to the principle of unique coiteration:
There exists a unique

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

Unique Coiteration

- 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.

$$\begin{array}{lllll} \mathrm{head} & (g\ x) & = & n & \text{for some } n : \mathbb{N} \text{ depending on } x \\ \mathrm{tail} & (g\ x) & = & g\ x' & \text{for some } x' : X \text{ depending on } x \end{array}$$

Unique Corecursion

- From unique coiteration we can derive **unique corecursion**:
  There exists a unique

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

- This means we can define $g\ x$ by **copattern matching**

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 not primitive but **defined** by corecursion

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

Examples

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

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

Bisimilarity

- Bisimilarity $\sim$ on Streams is an **indexed final coalgebra**.
- Consider the category $\mathrm{Set}^{\mathrm{Stream} \times \mathrm{Stream}}$ of binary relations

$$\varphi : \mathrm{Stream} \times \mathrm{Stream} \to \mathrm{Set}$$

- Let

$$\begin{array}{l} \mathrm{F}^{\sim} \ : \ \mathrm{Set}^{\mathrm{Stream} \times \mathrm{Stream}} \to \mathrm{Set}^{\mathrm{Stream} \times \mathrm{Stream}} \\ \mathrm{F}^{\sim}(\varphi, (s, s')) = (\mathrm{head}\ s = \mathrm{head}\ s') \times \varphi\ (\mathrm{tail}\ s, \mathrm{tail}\ s') \end{array}$$

Bisimilarity

- That $\sim$ is a $\mathrm{F}^{\sim}$ coalgebra means there exist

$$\begin{aligned}\mathrm{elim}_{\sim} : & (s, s' : \mathrm{Stream}) \\ & \to s \sim s' \\ & \to (\mathrm{head}\ s = \mathrm{head}\ s') \times (\mathrm{tail}\ s \sim \mathrm{tail}\ s')\end{aligned}$$

  i.e.

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

- Let $\mathrm{elim}^0_{\sim}$ and $\mathrm{elim}^1_{\sim}$ the two components of $\mathrm{elim}_{\sim}$,

$$\begin{aligned}\mathrm{elim}^0_{\sim} &: (s, s' : \mathrm{Stream}) \to s \sim s' \to \mathrm{head}\ s = \mathrm{head}\ s' \\ \mathrm{elim}^1_{\sim} &: (s, s' : \mathrm{Stream}) \to s \sim s' \to \mathrm{tail}\ s \sim \mathrm{tail}\ s'\end{aligned}$$

  and hide the first two arguments of $\mathrm{elim}^i_{\sim}$.

Bisimilarity

- That $\sim$ is a final $\mathrm{F}^{\sim}$-coalgebra means that it is the largest such relation:

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

- This means that

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

then

$$\forall s, s'.\varphi\,(s,s') \rightarrow s \sim s'$$

Bisimilarity

- So we have

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

and if

$$\forall s, s'.\varphi\,(s, s') \to \text{head } s = \text{head } s' \wedge \varphi\,(\text{tail } s, \text{tail } s')$$

then

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

Corecursive Proof of Bisimilarity

- Because $\sim$ is a final coalgebra we can compute proofs of it by corecursion:
- We can define

$$
\begin{array}{lcl}
f : (s, s' : \mathrm{Stream}) \to \varphi\ s\ s' \to s \sim s' \\
\mathrm{elim}^0_\sim\ (f\ s\ s'\ x) & = & \text{an element of } \mathrm{head}\ s = \mathrm{head}\ s' \\
\mathrm{elim}^0_\sim\ (f\ s\ s'\ x) & = & \text{an element of } (\mathrm{tail}\ s) \sim (\mathrm{tail}\ s')
\end{array}
$$

where in the last line we can use

- either a proof of $\mathrm{tail}\ s \sim \mathrm{tail}\ s'$ defined before
- or use the corecursion hypothesis $f\ (\mathrm{tail}\ s)\ (\mathrm{tail}\ s')\ x'$ for some $x' : \varphi\ (\mathrm{tail}\ s)\ (\mathrm{tail}\ s')$

Coinduction

Theorem

Assume $\mathrm{Stream} : \mathrm{Set}$, $\mathrm{head} : \mathrm{Stream} \to \mathbb{N}$, $\mathrm{tail} : \mathrm{Stream} \to \mathrm{Stream}$.
The following are equivalent

- *The principle of unique coiteration.*
- *The principle of unique corecursion.*
- *The principle of iteration together with the principle that bisimilarity $\sim$ implies equality*

$$\forall s, s' : \mathrm{Stream}. s \sim s' \to s = s'$$

Because of the possibility of defining elements of $s \sim s'$ the latter can be considered as a **principle of coinduction**.

Principle of Coinduction

- Let $\varphi : \mathrm{Stream} \to \mathrm{Stream} \to \mathrm{Set}$.
- We can prove

$$\forall s, s' : \mathrm{Stream}.\varphi\ s\ s' \to s = s'$$

by showing

$$\begin{array}{l}\forall s, s' : \mathrm{Stream}.\varphi\ s\ s' \to \mathrm{head}\ s = \mathrm{head}\ s' \\ \forall s, s' : \mathrm{Stream}.\varphi\ s\ s' \to \mathrm{tail}\ s = \mathrm{tail}\ s'\end{array}$$

where for proving $\mathrm{tail}\ s = \mathrm{tail}\ s'$ we can use the coinduction hypothesis that $\varphi\ (\mathrm{tail}\ s)\ (\mathrm{tail}\ s')$ implies $\mathrm{tail}\ s = \mathrm{tail}\ s'$.

Indexed Coinduction

- Instead of defining φ as a predicate $\mathrm{Stream} \to \mathrm{Stream} \to \mathrm{Set}$ we can assume

  $A : \mathrm{Set}$
  $s, t : A \to \mathrm{Stream}$
  and define
  $\varphi\ s'\ t' = (a : A) \times (s' = s\ a) \times (t' = t\ a)$

- Coinduction of φ becomes then the principle of indexed coinduction (see next slide)

Indexed Coinduction

- Assume

$$\begin{array}{lcl} A & : & \mathrm{Set} \\ s_0, s_1 & : & A \to \mathrm{Stream} \end{array}$$

- We can prove

$$\forall a : A. s_0\ a = s_1\ a$$

by showing

$$\begin{array}{l} \forall a : A. \mathrm{head}\ (s\ a) = \mathrm{head}\ (t\ a) \\ \forall a : A. \mathrm{tail}\ (s\ a) = \mathrm{tail}\ (t\ a) \end{array}$$

where for proving $\mathrm{tail}\ (s\ a) = \mathrm{tail}\ (t\ a)$ we can use that $\mathrm{tail}\ (s\ a) = s\ a'$ and $\mathrm{tail}\ (t\ a) = t\ a'$ and therefore by **coinduction-hypothesis** $s\ a' = t\ a'$.

Example Proof by Coinduction

- Remember

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

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

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

Example Proof by Coinduction

- We show
$$\forall n \in \mathbb{N}.\mathrm{inc}'\ n = \mathrm{inc}\ n \wedge \mathrm{inc}''\ n = \mathrm{inc}\ n$$
- Formally we would use in the above

$$\begin{array}{l} A = \mathbb{N} + \mathbb{N} \\ s\ (\mathrm{inl}\ n) = \quad \mathrm{inc}'\ n \\ s\ (\mathrm{inr}\ n) = \quad \mathrm{inc}''\ n \\ t\ (\mathrm{inl}\ n) = \quad \mathrm{inc}\ n \\ t\ (\mathrm{inr}\ n) = \quad \mathrm{inc}\ n \end{array}$$

and show

$$\forall a : A. s\ a = t\ a$$

Example Proof by Coinduction

- Proof of

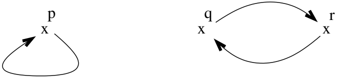
$$\forall n \in \mathbb{N}.\mathrm{inc}'\ n = \mathrm{inc}\ n \wedge \mathrm{inc}''\ n = \mathrm{inc}\ n$$

- Assume $n : \mathbb{N}$.

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

Bisimilarity

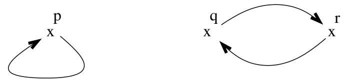
- Consider the following (unlabelled) transition system:



- Bisimilarity is the final coalgebra

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

Proof using the Definition of $\sim$



- We show $p \sim q \land p \sim r$ by coinduction:
- **Coinduction step for** $p \sim q$:
  - 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 $p \sim q$.

Traditional Argument of Proving Bisimiliarity

- The standard argument for showing $p \sim q \land p \sim r$ is as follows:
  Define a relation φ on states by

  $$\varphi(p', q') \Leftrightarrow p' = p \land (q' = q \lor q' = r)$$

  Show φ is a simulation:

  $$\forall p, p', q.\varphi(p, q) \land p \longrightarrow p' \Rightarrow \exists q'.q \longrightarrow q' \land \varphi(p', q')$$
  $$\forall p, q, q'.\varphi(p, q) \land q \longrightarrow q' \Rightarrow \exists p'.p \longrightarrow p' \land \varphi(p', q')$$

Comparison with Proofs by Induction

- We can compare both proofs to proofs by induction on natural number. Consider a proof of

$$\forall n, m, k . n + (m + k) = (n + m) + k$$

- The traditional proof would corresponds to defining a relation

$$R(k) \Leftrightarrow \forall n, m . n + (m + k) = (n + m) + k$$

and showing

$$R(0) \wedge \forall n . R(n) \rightarrow R(\mathrm{S}(n))$$

- Although this argument and the standard inductive proof using the induction hypothesis are equivalent, the standard inductive proof is more convenient and easier to follow.
- We hope that proofs by coinduction will similarly be easier if we do it by referring to the coinduction hypothesis.

Nested Pattern Matching

- Course of Value primitive recursion allows deep pattern matching. E.g. we can define the Fibonaccie numbers

$$
\begin{array}{lcl}
\mathrm{fib} : \mathbb{N} \to \mathbb{N} & & \\
\mathrm{fib}\ 0 & = & 1 \\
\mathrm{fib}\ (\mathrm{S}\ 0) & = & 1 \\
\mathrm{fib}\ (\mathrm{S}\ (\mathrm{S}\ n)) & = & \mathrm{fib}\ n + \mathrm{fib}\ (\mathrm{S}\ n)
\end{array}
$$

- We can now even mix pattern and copattern matching.

Example Mixed Pattern/Copattern Matching

- We can define now functions by patterns and copatterns.
- Example define stream:
  $f\ n =$
  $n, n, n-1, n-1, \ldots 0, 0, N, N, N-1, N-1, \ldots 0, 0, N, N, N-1, N-1,$

Example Mixed Pattern/Copattern Matching

$$f\ n = n, n, n-1, n-1, \ldots 0, 0, N, N, N-1, N-1, \ldots 0, 0, N, N, N-1, N-1,$$

$$\begin{array}{l} f : \mathbb{N} \to \mathrm{Stream} \\ f \quad = \quad ? \end{array}$$

Example Mixed Pattern/Copattern Matching

$f\ n = n, n, n-1, n-1, \ldots 0, 0, N, N, N-1, N-1, \ldots 0, 0, N, N, N-1, N-1,$

$$\begin{array}{l} f : \mathbb{N} \to \mathrm{Stream} \\ f \quad = \quad ? \end{array}$$

Copattern matching on $f : \mathbb{N} \to \mathrm{Stream}$:

$$\begin{array}{l} f : \mathbb{N} \to \mathrm{Stream} \\ f\ n \quad = \quad ? \end{array}$$

Example Mixed Pattern/Copattern Matching

$f\ n = n, n, n-1, n-1, \ldots 0, 0, N, N, N-1, N-1, \ldots 0, 0, N, N, N-1, N-1,$

$$\begin{array}{l} f : \mathbb{N} \to \mathrm{Stream} \\ f\ n \quad = \quad ? \end{array}$$

Copattern matching on $f\ n : \mathrm{Stream}$:

$$\begin{array}{lll} f : \mathbb{N} \to \mathrm{Stream} & & \\ \mathrm{head}\ (f\ n) & = & ? \\ \mathrm{tail}\ \ (f\ n) & = & ? \end{array}$$

Example Mixed Pattern/Copattern Matching

$f\ n = n, n, n-1, n-1, \ldots 0, 0, N, N, N-1, N-1, \ldots 0, 0, N, N, N-1, N-1,$

$$\begin{array}{l} f : \mathbb{N} \to \mathrm{Stream} \\ f\ n \quad = \quad ? \end{array}$$

Solve first case, copattern match on second case:

$$\begin{array}{llll} f : \mathbb{N} \to \mathrm{Stream} & & & \\ \mathrm{head} & (f\ n) & = & n \\ \mathrm{head}\ (\mathrm{tail} & (f\ n)) & = & ? \\ \mathrm{tail} \quad (\mathrm{tail} & (f\ n)) & = & ? \end{array}$$

Example Mixed Pattern/Copattern Matching

$f\ n = n, n, n-1, n-1, \ldots 0, 0, N, N, N-1, N-1, \ldots 0, 0, N, N, N-1, N-1,$

$$\begin{array}{l} f : \mathbb{N} \to \text{Stream} \\ f\ n \quad = \quad ? \end{array}$$

Solve second line, pattern match on n

$$\begin{array}{llll} f : \mathbb{N} \to \text{Stream} & & & \\ \text{head} & (f\ n) & = & n \\ \text{head} & (\text{tail}\ (f\ n)) & = & n \\ \text{tail} & (\text{tail}\ (f\ 0)) & = & ? \\ \text{tail} & (\text{tail}\ (f\ (\text{S}\ n))) & = & ? \end{array}$$

Example Mixed Pattern/Copattern Matching

$f\ n = n, n, n-1, n-1, \ldots 0, 0, N, N, N-1, N-1, \ldots 0, 0, N, N, N-1, N-1,$

$$\begin{array}{l} f : \mathbb{N} \to \mathrm{Stream} \\ f\ n \quad = \quad ? \end{array}$$

Solve remaining cases

$$\begin{array}{llll} f : \mathbb{N} \to \mathrm{Stream} & & & \\ \mathrm{head} & (f\ n) & = & n \\ \mathrm{head} & (\mathrm{tail}\ (f\ n)) & = & n \\ \mathrm{tail} & (\mathrm{tail}\ (f\ 0)) & = & f\ N \\ \mathrm{tail} & (\mathrm{tail}\ (f\ (\mathrm{S}\ n))) & = & f\ n \end{array}$$

Results of paper in POPL (2013)

- Development of a recursive simply typed calculus (no termination check).
- Allows to derive schemata for pattern/copattern matching.
- Proof that subject reduction holds.

$$t : A, \quad t \longrightarrow t' \text{ implies } t' : A$$

- Subject reduction fails when using codata types in combination with the equality type (e.g. in Coq and early versions of Agda).

Consider Example from above

$$
\begin{array}{llll}
f : \mathbb{N} \to \mathrm{Stream} & & & \\
\mathrm{head} & (f\ n) & = & n \\
\mathrm{head} & (\mathrm{tail}\ (f\ n)) & = & n \\
\mathrm{tail} & (\mathrm{tail}\ (f\ 0)) & = & f\ N \\
\mathrm{tail} & (\mathrm{tail}\ (f\ (\mathrm{S}\ n))) & = & f\ n
\end{array}
$$

We show how this example can be reduced to unnested (co)pattern matching.
In a second step (not shown today) one can reduce it to primitive (co)recursion operators.

Unnesting of Nested (Co)Pattern Matching

We follow the steps in the pattern matching:
We start with

$$
\begin{array}{lcl}
f : \mathbb{N} \to \mathrm{Stream} & & \\
\mathrm{head}\ (f\ \ n) & = & n \\
\mathrm{tail}\ \ \ (f\ \ n) & = & ?
\end{array}
$$

Copattern matching on $\text{tail}\ (f\ n)$:

$$
\begin{array}{lllcl}
f : \mathbb{N} \to \text{Stream} & & & & \\
\text{head} & & (f\ n) & = & n \\
\text{head} & (\text{tail} & (f\ n)) & = & n \\
\text{tail} & (\text{tail} & (f\ n)) & = & ?
\end{array}
$$

corresponds to

$$
\begin{array}{lllcl}
 & f : \mathbb{N} \to \text{Stream} & & & \\
 & \text{head}\ (f\ n) & & = & n \\
 & \text{tail}\ \ (f\ n) & & = & g\ n \\
 & & & & \\
 & g : \mathbb{N} \to \text{Stream} & & & \\
(\text{head}\ (\text{tail}\ (f\ n)) \quad =) & \text{head}\ (g\ n) & & = & n \\
(\text{tail}\ \ (\text{tail}\ (f\ n)) \quad =) & \text{tail}\ \ (g\ n) & & = & ?
\end{array}
$$

Pattern matching on $\text{tail}\ (\text{tail}\ (f\ n))$:

$$\begin{array}{llll} f : \mathbb{N} \to \text{Stream} & & & \\ \text{head} & (f\ n) & = & n \\ \text{head}\ (\text{tail}\ (f\ n)) & & = & n \\ \text{tail} & (\text{tail}\ (f\ 0)) & = & f\ N \\ \text{tail} & (\text{tail}\ (f\ (\text{S}\ n))) & = & f\ n \end{array}$$

corresponds to

$$\begin{array}{lllll} & f : \mathbb{N} \to \text{Stream} & & & \\ & \text{head}\ (f\ n) & & = & n \\ & \text{tail} & (f\ n) & = & g\ n \end{array}$$

$$\begin{array}{lllll} & g : \mathbb{N} \to \text{Stream} & & & \\ (\text{head}\ (\text{tail}\ (f\ n)) \quad =) & \text{head}\ (g\ n) & & = & n \\ (\text{tail}\ (\text{tail}\ (f\ n)) \quad =) & \text{tail} & (g\ n) & = & k\ n \end{array}$$

$$\begin{array}{lllll} & k : \mathbb{N} \to \text{Stream} & & & \\ (\text{tail}\ (\text{tail}\ (f\ 0)) \quad =) & k & 0 & = & f\ N \\ (\text{tail}\ (\text{tail}\ (f\ (\text{S}\ n))) \quad =) & k & (\text{S}\ n) & = & f\ n \end{array}$$

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.

Conclusion

- Copattern matching as the dual of pattern matching.
  - Pattern matching is an elimination principle for inductive types (initial algebras).
  - Copattern matching is an introduction principle for coinductive types (final coalgebras).
- Mixed pattern and copattern matching can be reduced to simple pattern and copattern matching.