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

\[ N \text{ as an Initial Algebra} \]

- \( N \) is initial algebra of the functor \( F(X) = 1 + X \)

\[
\begin{align*}
F(N) &= 1 + N \\
F(g) &= 1 + g \\
F(A) &= 1 + A \\
\exists! g
\end{align*}
\]

\( f' \) can be decomposed as \( f' = a + f \)
Unique Iteration

1 + N \xrightarrow{0 + S} N
1 + g \xrightarrow{\exists! \ g} 1 + A \xrightarrow{a + f} A

Unique existence of \( g \) means unique iteration:

Given \( a : A \) and \( f : A \to A \), there exists a unique
\[
g : N \to A
\]
\[
g(0) = a
\]
\[
g(Sn) = f(gn)
\]
i.e.
\[
g(S^n 0) = f^n a
\]

Induction

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

Assume \( A : N \to \text{Set} \), \( a : A \ 0 \) and \( f : (n : N) \to A \ n \to A \ (Sn) \)
There exists a unique
\[
g : (n : N) \to A \ n
\]
\[
g(0) = a
\]
\[
g(Sn) = f(n \ g \ n)
\]

Pattern Matching

The above means that we can define
\[
g : (n : N) \to A \ n
\]
\[
g(0) = a \text{ for some } a : A
\]
\[
g(Sn) = a' \text{ for some } a' : A \text{ depending on } n
\]
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 \( n : N \).
Theorem

Assume $\mathbb{N} : \text{Set}$, $0 : \mathbb{N}$, $S : \mathbb{N} \to \mathbb{N}$.
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 $\oplus$ is $\times$, so we use for clarity a functor using product rather than disjoint union:
- Stream is the final coalgebra of $F(X) = \mathbb{N} \times X$

$$
\begin{array}{c}
X \\
\downarrow f \\
\uparrow \exists! g \\
\text{Stream} \\
\downarrow \text{head} \times \text{tail} \\
\mathbb{N} \times \text{Stream} = F(\text{Stream})
\end{array}
\quad
\begin{array}{c}
X \\
\downarrow f_0 \times f_1 \\
\uparrow \exists! g \\
\text{Stream} \\
\downarrow \text{head} \times \text{tail} \\
\mathbb{N} \times \text{Stream} = F(\text{Stream})
\end{array}
$$

- We can decompose $f$ as

$$f = f_0 \times f_1$$

Unique Coiteration

This corresponds to the principle of unique coiteration:
There exists a unique $g : A \to \text{Stream}$

$$
\begin{align*}
g : A &\to \text{Stream} \\
\text{head} (g \ x) &= f_0 \ x \\
\text{tail} (g \ x) &= g \ (f_1 \ x)
\end{align*}
$$
Unique Coiteration

We had:

\[
\begin{align*}
\text{head } (g \ x) &= f_0 \times \\
\text{tail } (g \ x) &= g \ (f_1 \ x)
\end{align*}
\]

By choosing \( f_0, f_1 \) we can define \( g : X \rightarrow \text{Stream} \) s.t.

\[
\begin{align*}
\text{head } (g \ x) &= n \quad \text{for some } n : \mathbb{N} \text{ depending on } x \\
\text{tail } (g \ x) &= g \ x' \quad \text{for some } x' : X \text{ depending on } x
\end{align*}
\]

Unique Corecursion

From unique coiteration we can derive **unique corecursion**:

There exists a unique \( g : A \rightarrow \text{Stream} \)

\[
\begin{align*}
\text{head } (g \ x) &= n \quad \text{for some } n : \mathbb{N} \text{ depending on } x \\
\text{tail } (g \ x) &= g \ x' \quad \text{for some } x' : X \text{ depending on } x \\
&\text{or} \\
&= s \quad \text{for some } s : \text{Stream} \text{ depending on } x
\end{align*}
\]

This means we can define \( g \ x \) by **copattern matching**

Examples

We can define

\[
\begin{align*}
\text{cons} &: (\mathbb{N} \times \text{Stream}) \rightarrow \text{Stream} \\
\text{head } (\text{cons} \ (n, s)) &= n \\
\text{tail } (\text{cons} \ (n, s)) &= s
\end{align*}
\]

Note: \text{cons} not primitive but **defined** by corecursion

\[
\begin{align*}
\text{inc} &: \mathbb{N} \rightarrow \text{Stream} \\
\text{head } (\text{inc} \ n) &= n \\
\text{tail } (\text{inc} \ n) &= \text{inc} \ (n + 1)
\end{align*}
\]

\[
\begin{align*}
\text{inc'} &: \mathbb{N} \rightarrow \text{Stream} \\
\text{head } (\text{inc'} \ n) &= n \\
\text{tail } (\text{inc'} \ n) &= \text{inc''} \ (n + 1)
\end{align*}
\]

\[
\begin{align*}
\text{inc''} &: \mathbb{N} \rightarrow \text{Stream} \\
\text{head } (\text{inc''} \ n) &= n \\
\text{tail } (\text{inc''} \ n) &= \text{inc'} \ (n + 1)
\end{align*}
\]

Anton Setzer (Swansea)  Pattern and Copattern matching  13/ 46

Anton Setzer (Swansea)  Pattern and Copattern matching  14/ 46

Anton Setzer (Swansea)  Pattern and Copattern matching  15/ 46

Anton Setzer (Swansea)  Pattern and Copattern matching  16/ 46
Bisimilarity and Coinduction

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

Bisimilarity

That \( \sim \) is a final coalgebra means there exist

\[
\text{elim}_\sim : (s, s' : \text{Stream}) \\
\rightarrow s \sim s' \\
\rightarrow (\text{head } s = \text{head } s') \times (\text{tail } s \sim \text{tail } s')
\]

i.e.

\( s \sim s' \rightarrow (\text{head } s = \text{head } s') \land ((\text{tail } s) \sim (\text{tail } s')) \)

Let \( \text{elim}_0^\sim \) and \( \text{elim}_1^\sim \) the two components of \( \text{elim}_\sim \).

\[
\text{elim}_0^\sim : (s, s' : \text{Stream}) \rightarrow s \sim s' \rightarrow \text{head } s = \text{head } s' \\
\text{elim}_1^\sim : (s, s' : \text{Stream}) \rightarrow s \sim s' \rightarrow \text{tail } s \sim \text{tail } s'
\]

and hide the first two arguments of \( \text{elim}_i^\sim \).

Bisimilarity

That \( \sim \) is an indexed final coalgebra.

Consider the category \( \text{Set}^{\text{Stream} \times \text{Stream}} \) of binary relations

\( \varphi : \text{Stream} \times \text{Stream} \rightarrow \text{Set} \)

Let

\[
F^\sim : \text{Set}^{\text{Stream} \times \text{Stream}} \rightarrow \text{Set}^{\text{Stream} \times \text{Stream}} \\
F^\sim (\varphi, (s, s')) = (\text{head } s = \text{head } s') \times \varphi (\text{tail } s, \text{tail } s')
\]

This means that

\[
\forall s, s'. \varphi (s, s') \rightarrow \text{head } s = \text{head } s' \land \varphi (\text{tail } s, \text{tail } s')
\]

then

\[
\forall s, s'. (s, s') \rightarrow s \sim s'
\]
Bisimilarity

- So we have
  \[ s \sim s' \rightarrow \text{head} s = \text{head} s' \land (\text{tail} s) \sim (\text{tail} s') \]

and if

\[ \forall s, s'. \varphi (s, s') \rightarrow \text{head} s = \text{head} s' \land \varphi (\text{tail} s, \text{tail} s') \]

then

\[ \forall s, s'. \varphi (s, s') \rightarrow s \sim s' \]

Corecursive Proof of Bisimilarity

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

\[
\begin{align*}
  f : (s, s') : \text{Stream} &\rightarrow \varphi s s' \rightarrow s \sim s' \\
  \text{elim}_0 (f s s' x) &\quad = \quad \text{an element of head} s = \text{head} s' \\
  \text{elim}_0 (f s s' x) &\quad = \quad \text{an element of} (\text{tail} s) \sim (\text{tail} s')
\end{align*}
\]

where in the last line we can use
- either a proof of \( \text{tail} s \sim \text{tail} s' \) defined before
- or use the corecursion hypothesis \( f (\text{tail} s) (\text{tail} s') x' \) for some \( x' : \varphi (\text{tail} s) (\text{tail} s') \)

Principle of Coinduction

- Let \( \varphi : \text{Stream} \rightarrow \text{Stream} \rightarrow \text{Set} \).
- We can prove

\[ \forall s, s' : \text{Stream}. \varphi s s' \rightarrow s = s' \]

by showing

\[ \forall s, s' : \text{Stream}. \varphi s s' \rightarrow \text{head} s = \text{head} s' \]

\[ \forall s, s' : \text{Stream}. \varphi s s' \rightarrow \text{tail} s = \text{tail} s' \]

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

- Instead of defining \( \varphi \) as a predicate \( \text{Stream} \to \text{Stream} \to \text{Set} \) we can assume
  
  \[
  A : \text{Set} \\
  s, t : A \to \text{Stream}
  \]
  
  and define
  
  \[
  \varphi \ s' \ t' = (a : A) \times (s' = s \ a) \times (t' = t \ a)
  \]

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

Example Proof by Coinduction

- Remember

\[
\begin{align*}
\text{inc} & : \mathbb{N} \to \text{Stream} \\
\text{head}(\text{inc} \ n) & = n \\
\text{tail}(\text{inc} \ n) & = \text{inc} \ (n+1)
\end{align*}
\]

\[
\begin{align*}
\text{inc}' & : \mathbb{N} \to \text{Stream} \\
\text{head}(\text{inc}'(n)) & = n \\
\text{tail}(\text{inc}'(n)) & = \text{inc}''(n+1)
\end{align*}
\]

\[
\begin{align*}
\text{inc}'' & : \mathbb{N} \to \text{Stream} \\
\text{head}(\text{inc}''(n)) & = n \\
\text{tail}(\text{inc}''(n)) & = \text{inc}'(n+1)
\end{align*}
\]

- We show

\[
\forall n \in \mathbb{N}. \text{inc}' \ n = \text{inc} \ n \land \text{inc}'' \ n = \text{inc} \ n
\]

- Formally we would use in the above

\[
\begin{align*}
A & = \mathbb{N} + \mathbb{N} \\
\text{s (inl} \ n) & = \text{inc}' \ n \\
\text{s (inr} \ n) & = \text{inc}'' \ n \\
\text{t (inl} \ n) & = \text{inc} \ n \\
\text{t (inr} \ n) & = \text{inc} \ n
\end{align*}
\]

and show

\[
\forall a : A. s \ a = t \ a
\]
Example Proof by Coinduction

Proof of
\[\forall n \in \mathbb{N}. \text{inc}' n = \text{inc} n \land \text{inc}'' n = \text{inc} n\]

Assume \(n : \mathbb{N}\).

\[
\begin{align*}
\text{head (inc}' n \text{)} &= n = \text{head (inc} n \text{)} \\
\text{head (inc}'' n \text{)} &= n = \text{head (inc} n \text{)} \\
\text{tail (inc}' n \text{)} &= \text{inc}'' (n + 1) \overset{\text{co-IH}}{=} \text{inc} (n + 1) = \text{tail (inc} n \text{)} \\
\text{tail (inc}'' n \text{)} &= \text{inc}' (n + 1) \overset{\text{co-IH}}{=} \text{inc} (n + 1) = \text{tail (inc} n \text{)}
\end{align*}
\]

Proof using the Definition of \(\sim\)

We show \(p \sim q \land p \sim r\) by coinduction:

**Coinduction step for \(p \sim q\):**
- Assume \(p \rightarrow p'\). Then \(p' = p\).
  We have \(q \rightarrow r\) and by co-IH \(p \sim r\).
- Assume \(q \rightarrow q'\). Then \(q' = r\).
  We have \(p \rightarrow p\) and by co-IH \(p \sim r\).

**Coinduction step for \(p \sim r\):**
- Assume \(p \rightarrow p'\). Then \(p' = p\).
  We have \(r \rightarrow q\) and by co-IH \(p \sim q\).
- Assume \(r \rightarrow r'\). Then \(r' = q\).
  We have \(p \rightarrow p\) and by co-IH \(p \sim q\).
Traditional Argument of Proving Bisimilarity

The standard argument for showing \( p \sim q \land p \sim r \) is as follows:

Define a relation \( \varphi \) on states by

\[
\varphi(p', q') \iff p' = p \land (q' = q \lor q' = r)
\]

Show \( \varphi \) is a simulation:

\[
\forall p, p', q. \varphi(p, q) \land p \rightarrow p' \Rightarrow \exists q'. q \rightarrow q' \land \varphi(p', q')
\]

\[
\forall p, q, q'. \varphi(p, q) \land q \rightarrow q' \Rightarrow \exists p'. p \rightarrow p' \land \varphi(p', q')
\]

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 correspond to defining a relation

\[
R(k) \iff \forall n. n + (m + k) = (n + m) + k
\]

and showing

\[
R(0) \land \forall n. R(n) \rightarrow R(S(n))
\]

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 Fibonacci numbers

\[
\begin{align*}
fib & : \mathbb{N} \rightarrow \mathbb{N} \\
fib(0) & = 1 \\
fib(S(0)) & = 1 \\
fib(S(S(n))) & = fib(n) + fib(S(n))
\end{align*}
\]

We can now even mix pattern and copattern matching.
We can define now functions by patterns and copatterns.

Example define stream:

\[
  f \ n = n, n, n-1, n-1, \ldots 0, 0, N, N, N-1, N-1, \ldots 0, 0, N, N, N-1, N-1, 
\]

Copattern matching on \( f : \mathbb{N} \rightarrow \text{Stream} \):

\[
  f = ?
\]

\[
  f : \mathbb{N} \rightarrow \text{Stream}
\]

\[
  f = ?
\]

\[
  \text{head} (f \ n) = ?
\]

\[
  \text{tail} (f \ n) = ?
\]

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.

\[
  t : A, \quad t \rightarrow t' \text{ implies } t' : A
\]

- 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

\[ f : \mathbb{N} \to \text{Stream} \]
\[
\begin{align*}
\text{head } (f \ n) &= n \\
\text{head } (\text{tail } (f \ n)) &= n \\
\text{tail } (\text{tail } (f \ 0)) &= f \ N \\
\text{tail } (\text{tail } (f \ (S \ n))) &= f \ n
\end{align*}
\]

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.

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

\[ f : \mathbb{N} \to \text{Stream} \]
\[
\begin{align*}
\text{head } (f \ n) &= n \\
\text{head } (\text{tail } (f \ n)) &= n \\
\text{tail } (\text{tail } (f \ n)) &= ?
\end{align*}
\]

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

\[ f : \mathbb{N} \to \text{Stream} \]
\[
\begin{align*}
\text{head } (f \ n) &= n \\
\text{tail } (f \ n) &= ?
\end{align*}
\]

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

\[ f : \mathbb{N} \to \text{Stream} \]
\[
\begin{align*}
\text{head } (f \ n) &= n \\
\text{head } (\text{tail } (f \ n)) &= n \\
\text{tail } (\text{tail } (f \ 0)) &= f \ N \\
\text{tail } (\text{tail } (f \ (S \ n))) &= f \ n
\end{align*}
\]

corresponds to

\[ g : \mathbb{N} \to \text{Stream} \]
\[
\begin{align*}
\text{head } (g \ n) &= n \\
\text{tail } (g \ n) &= g \ n
\end{align*}
\]

\[ k : \mathbb{N} \to \text{Stream} \]
\[
\begin{align*}
\text{tail } (k \ 0) &= f \ N \\
\text{tail } (k \ (S \ n)) &= f \ n
\end{align*}
\]
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.

- 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.