Unfolding Nested Patterns and Copatterns

Anton Setzer

(Swansea, UK)

Shonan Meeting on Coinduction
7 October 2013
Codata types and Decidable Equality

Pattern and Copattern Matching

Reduction of Mixed Pattern/Copattern Matching to Operators

Conclusion
Codata types and Decidable Equality

Pattern and Copattern Matching

Reduction of Mixed Pattern/Copattern Matching to Operators

Conclusion
Theorem Regarding Undecidability of Equality

Theorem

Assume the following:

- There exists a subset \( \text{Stream} \subseteq \mathbb{N} \),
- computable functions \( \text{head} : \text{Stream} \rightarrow \mathbb{N}, \text{tail} : \text{Stream} \rightarrow \text{Stream} \),
- a decidable equality \( _==_ \) on \( \text{Stream} \) which is congruence,
- the possibility to define elements of \( \text{Stream} \) by guarded recursion based on primitive recursive functions \( f, g : \mathbb{N} \rightarrow \mathbb{N} \), such that the standard equalities related to guarded recursion hold.

Then it is not possible to fulfil the following condition:

\[
\forall s, s' : \text{Stream}. \text{head} s = \text{head} s' \land \text{tail} s == \text{tail} s' \rightarrow s == s' \quad (\ast)
\]

\(^1\)Thanks to somebody in the audience (M. Hofmann?) pointed out during the talk that \( \text{Stream} \) needs not to be decidable.
Consequences for Codata Approach

Remark

Condition (∗) is fulfilled if we have an operation
cons : \(\mathbb{N} \rightarrow \text{Stream} \rightarrow \text{Stream} \) preserving equalities s.t.

\[
\forall s : \text{Stream}. s = \text{cons} (\text{head} s) (\text{tail} s)
\]

So we cannot have a type theory with streams, decidable type checking and decidable equality on streams such that

\[
\forall s. \exists n, s'. s == \text{cons} n s'
\]

as assumed by the codata approach.
Proof of Theorem

- Assume we had the above.
- By
  \[ s \approx n_0 :: n_1 :: n_2 :: \cdots n_k :: s' \]
  we mean the equations using head, tail expressing that \( s \) behaves as the stream indicated on the right hand side.
- Define by guarded recursion \( / : \text{Stream} \)
  \[ / \approx 1 :: 1 :: 1 :: \cdots \]
Proof of Theorem

- For e code for a Turing machine define by guarded recursion based on primitive recursion functions \( f, g \) s.t. if \( e \) terminates after \( n \) steps and returns result \( k \) then

\[
f(e) \approx \begin{cases} 
0 :: 0 :: 0 :: \cdots :: 0 :: l \\
\text{n times}
\end{cases}
\]

\[
g(e) \approx \begin{cases} 
0 :: 0 :: 0 :: \cdots :: 0 :: l & \text{if } k = 0 \\
\text{n times} \\
0 :: 0 :: 0 :: \cdots :: 0 :: l & \text{if } k > 0 \\
\text{n+1 times}
\end{cases}
\]
Proof of Theorem

\[ f \ e \ \approx \ \begin{cases} 0 :: 0 :: 0 :: \ldots :: 0 :: l \\ n \text{ times} \end{cases} \]

\[ g \ e \ \approx \ \begin{cases} 0 :: 0 :: 0 :: \ldots :: 0 :: l & \text{if } k = 0 \\ 0 :: 0 :: 0 :: \ldots :: 0 :: l & \text{if } k > 0 \\ n+1 \text{ times} \end{cases} \]

- If \( e \) terminates after \( n \) steps with result 0 then
  \[ f \ e =\!\!\!\approx g \ e \]

- If \( e \) terminates after \( n \) steps with result \( > 0 \) then
  \[ \neg(f \ e =\!\!\!\approx g \ e) \]
Proof of Theorem

- So

\[ \lambda e. (f \ e \ == \ g \ e) \]

separates the TM with result 0 from those with result \( \geq 0 \).

- But these two sets are inseparable.
During the talk a related article by Conor McBride was discussed:
- Let’s see how things unfold: Reconciling the infinite with the intensional. Proceedings of CALCO’09, LNCS, 2009, 113 – 126.
- While this paper contains the idea we believe that we state a more precise theorem and provide a more formal proof.
- We were not able to reduce the result directly to the undecidability of the Turing Halting problem as suggested in that paper.
Codata types and Decidable Equality

Pattern and Copattern Matching

Reduction of Mixed Pattern/Copattern Matching to Operators

Conclusion
Coalgebras defined by Elimination Rules

\[
\text{coalg Stream : Set where}
\]
\[
\begin{align*}
\text{head} & : \text{Stream} \rightarrow \mathbb{N} \\
\text{tail} & : \text{Stream} \rightarrow \text{Stream}
\end{align*}
\]

Copattern matching:

\[
g : A \rightarrow \text{Stream}
\]
\[
\begin{align*}
\text{head } (g \ a) & = f_0 \ a \\
\text{tail } (g \ a) & = g \ (f_1 \ a)
\end{align*}
\]
or
\[
\begin{align*}
\text{tail } (g \ a) & = f_2 \ a
\end{align*}
\]
Patterns and Copatterns

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

$f : \mathbb{N} \rightarrow \text{Stream}$

$f = ?$
Patterns and Copatterns

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

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

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

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

\[ f \ n = ? \]
Patterns and Copatterns

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

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

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

\[ f : \mathbb{N} \rightarrow \text{Stream} \]
\[ \text{head} (f \ n) = ? \]
\[ \text{tail} (f \ n) = ? \]
Patterns and Copatterns

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

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

Solve first case, copattern match on second case:

\[ f : \mathbb{N} \rightarrow \text{Stream} \]
\[ \text{head} (f \ n) = n \]
\[ \text{head} (\text{tail} (f \ n)) = ? \]
\[ \text{tail} (\text{tail} (f \ n)) = ? \]
Patterns and Copatterns

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

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

\[ f \, n = ? \]

Solve second line, pattern match on \( n \)

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

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

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

\[ \text{tail} \ (\text{tail} \ (f \, 0)) = ? \]

\[ \text{tail} \ (\text{tail} \ (f \ (S \ n))) = ? \]
Patterns and Copatterns

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

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

**Solve remaining cases**

\[ f : \mathbb{N} \rightarrow \text{Stream} \]
\[ \text{head} \ (f \ n) = n \]
\[ \text{head} \ (\text{tail} \ (f \ n)) = n \]
\[ \text{tail} \ (\text{tail} \ (f \ 0)) = f \ N \]
\[ \text{tail} \ (\text{tail} \ (f \ (S \ n))) = f \ n \]
Codata types and Decidable Equality

Pattern and Copattern Matching

Reduction of Mixed Pattern/Copattern Matching to Operators

Conclusion
Operators for Primitive (Co)Recursion

\[ P_{N,A} : A \rightarrow (N \rightarrow A \rightarrow A) \rightarrow N \rightarrow A \]
\[ P_{N,A} \text{ step}_0 \text{ step}_S 0 = \text{ step}_0 \]
\[ P_{N,A} \text{ step}_0 \text{ step}_S (S \ n) = \text{ step}_S \ n \ (P_{N,A} \text{ step}_0 \text{ step}_S \ n) \]

\[ \text{coP}_{\text{Stream},A} : (A \rightarrow N) \rightarrow (A \rightarrow (\text{Stream} + A)) \rightarrow A \rightarrow \text{Stream} \]
\[ \text{head} \ (\text{coP}_{\text{Stream},A} \text{ step}_{\text{head}} \text{ step}_{\text{tail}} \ a) = \text{ step}_{\text{head}} \ a \]
\[ \text{tail} \ (\text{coP}_{\text{Stream},A} \text{ step}_{\text{head}} \text{ step}_{\text{tail}} \ a) = \]
\[ \text{case}_{\text{Stream},A,\text{Stream id}} \ (\text{coP}_{\text{Stream},A} \text{ step}_{\text{head}} \text{ step}_{\text{tail}}) \ (\text{step}_{\text{tail}} \ a) \]
Operators for full/primitive (co)recursion

\[\text{coP}_{\text{Stream},A} : (A \rightarrow \mathbb{N}) \rightarrow (A \rightarrow (\text{Stream} + A)) \rightarrow A \rightarrow \text{Stream}\]

\[
\text{head} \left( \text{coP}_{\text{Stream},A} \text{ step}_{\text{head}} \text{ step}_{\text{tail}} a \right) = \text{ step}_{\text{head}} a
\]

\[
\text{tail} \left( \text{coP}_{\text{Stream},A} \text{ step}_{\text{head}} \text{ step}_{\text{tail}} a \right) = \\
\text{ case}_{\text{Stream},A,\text{Stream id}} \left( \text{coP}_{\text{Stream},A} \text{ step}_{\text{head}} \text{ step}_{\text{tail}} \right) \left( \text{step}_{\text{tail}} a \right)
\]

\[\text{coR}_{\text{Stream},A} : ((A \rightarrow \text{Stream}) \rightarrow A \rightarrow \mathbb{N}) \rightarrow ((A \rightarrow \text{Stream}) \rightarrow A \rightarrow \text{Stream}) \rightarrow \text{Stream}\]

\[
\text{head} \left( \text{coR}_{\text{Stream},A} \text{ step}_{\text{head}} \text{ step}_{\text{tail}} a \right) = \text{ step}_{\text{head}} \\
\left( \text{coR}_{\text{Stream},A} \text{ step}_{\text{head}} \text{ step}_{\text{tail}} \right) a
\]

\[
\text{tail} \left( \text{coR}_{\text{Stream},A} \text{ step}_{\text{head}} \text{ step}_{\text{tail}} a \right) = \\
\text{ step}_{\text{tail}} \\
\left( \text{coR}_{\text{Stream},A} \text{ step}_{\text{head}} \text{ step}_{\text{tail}} \right) a
\]
Consider Example from above

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

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

This example can be reduced to primitive (co)recursion.

**Step 1:** Following the development of the (co)pattern matching definition, unfold it into simultaneous non-nested (co)pattern matching definitions.
Step 1: Unnesting of Nested (Co)Pattern Matching

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

\[ f : \mathbb{N} \rightarrow \text{Stream} \]
\[ \text{head } (f \ n) = n \]
\[ \text{tail } (f \ n) = ? \]
**Copattern matching** on \( \text{tail} \ (f \ n) \):

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

corresponds to

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

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

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

corresponds to

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

\[
\begin{align*}
\text{head} & (g\ n) = n \\
\text{tail} & (g\ n) = k\ n
\end{align*}
\]

g : \mathbb{N} \rightarrow \text{Stream}

\[
\begin{align*}
(\text{head} & (\text{tail}\ (f\ n))) =) \text{head} (g\ n) = n \\
(\text{tail} & (\text{tail}\ (f\ n))) =) \text{tail} (g\ n) = k\ n
\end{align*}
\]

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

\[
\begin{align*}
(\text{tail} & (\text{tail}\ (f\ 0))) =) k\ 0 = f\ N \\
(\text{tail} & (\text{tail}\ (f\ (S\ n)))) =) k\ (S\ n) = f\ n
\end{align*}
\]
Step 2: Reduction to Primitive (Co)recursion

- This can now easily be reduced to full (co)recursion.
- In this example we can reduce it to primitive (co)recursion.
- First combine $f, g$ into one function $f + g$. 
\( f : \mathbb{N} \rightarrow \text{Stream} \)
\[
f \ n \quad = \quad (f + g) \ (\bar{f} \ n)
\]

\((f + g) : (\bar{f}(\mathbb{N}) + g(\mathbb{N})) \rightarrow \text{Stream}\)
\[
\text{head} \ ((f + g) (\bar{f} \ n)) \quad = \quad n
\]
\[
\text{head} \ ((f + g) (\bar{g} \ n)) \quad = \quad n
\]
\[
\text{tail} \ ((f + g) (\bar{f} \ n)) \quad = \quad (f + g) (\bar{g} \ n)
\]
\[
\text{tail} \ ((f + g) (\bar{f} \ n)) \quad = \quad k \ n
\]

\( k : \mathbb{N} \rightarrow \text{Stream} \)
\[
k \ 0 \quad = \quad (f + g) \ (\bar{f} \ N)
\]
\[
k \ (S \ n) \quad = \quad (f + g) \ (\bar{f} \ n)
\]
Unfolding of the Pattern Matchings

- The call of $k$ has result always of the form $(f + g)(fbf\ n))$. So we can replace the recursive call $k\ n$ by $(f + g)(f (k'\ n))$. 

Anton Setzer
Unfolding Nested (Co)Patterns
\[ f : \mathbb{N} \rightarrow \text{Stream} \]
\[ f \ n = (f + g) (\overline{f} \ n) \]

\[(f + g) : (f(\overline{\mathbb{N}}) + g(\overline{\mathbb{N}})) \rightarrow \text{Stream} \]
\[ \text{head} \ ((f + g) (\overline{f} \ n)) = n \]
\[ \text{head} \ ((f + g) (\overline{g} \ n)) = n \]
\[ \text{tail} \ ((f + g) (\overline{f} \ n)) = (f + g) (\overline{g} \ n) \]
\[ \text{tail} \ ((f + g) (\overline{f} \ n)) = (f + g) (\overline{f} (k' \ n)) \]

\[ k' : \mathbb{N} \rightarrow \mathbb{N} \]
\[ k \ 0 = N \]
\[ k \ (S \ n) = n \]
Unfolding of the Pattern Matchings

- $(f + g)$ can be defined by primitive corecursion.
- $k'$ can be defined by primitive recursion.
\[ f : \mathbb{N} \rightarrow \text{Stream} \]

\[ f \ n = (f + g) (\underline{f} \ n) \]

\[ (f + g) : (\underline{f}(\mathbb{N}) + g(\mathbb{N})) \rightarrow \text{Stream} \]

\[ (f + g) = \]

\[ \coP_{\text{Stream},(\underline{f}(\mathbb{N})+g(\mathbb{N}))} (\lambda x. \text{case}_r(x) \text{ of} \]

\[ (\underline{f} \ n) \rightarrow n \]

\[ (g \ n) \rightarrow n \)

\[ (\lambda x. \text{case}_r(x) \text{ of} \]

\[ (\underline{f} \ n) \rightarrow \underline{g} \ n \]

\[ (g \ n) \rightarrow \underline{f} (k' \ n)) \]

\[ k' : \mathbb{N} \rightarrow \mathbb{N} \]

\[ k' = P_{\mathbb{N},\mathbb{N}} \ n (\lambda n, ih. n) \]
The case distinction can be trivially replaced by the case distinction operator.
\[ f : \mathbb{N} \to \text{Stream} \]
\[ f \ n = (f + g) (\overline{f} \ n) \]

\[(f + g) : (f(\mathbb{N}) + g(\mathbb{N})) \to \text{Stream} \]
\[(f + g) = \]
\[\text{coP}_{\text{Stream}, f(\mathbb{N})+g(\mathbb{N})} \left( \text{case}_{f(\mathbb{N})+g(\mathbb{N})} \ id \ id \right) \]
\[\quad (\text{case}_{f(\mathbb{N})+g(\mathbb{N})} \ g \ (f \circ k')) \]

\[ k' : \mathbb{N} \to \mathbb{N} \]
\[ k' = P_{\mathbb{N},\mathbb{N}} n \ (\lambda n, \text{ih} \cdot n) \]
Codata types and Decidable Equality

Pattern and Copattern Matching

Reduction of Mixed Pattern/Copattern Matching to Operators

Conclusion
Conclusion

▶ Codata types make the assumption

\[ \forall s : \text{Stream.} \exists n, s'. s = \text{cons } n s' \]

which cannot be combined with a decidable equality.

▶ One can reduce certain cases of recursive nested (co)pattern matching to primitive (co)recursion.

▶ Systematic treatment needs still to be done.
▶ Cases which can be reduced should be those to be accepted by a termination checker.
▶ If the reduction succeeds we get a normalising version (by Mendler and Geuvers).
▶ Therefore a termination checked version of the calculus is normalising.