# Pattern und Copattern Matching

Anton Setzer
Swansea, UK

(Sect 1 - 3 joint work with Andreas Abel, Brigitte Pientka, and David Thibodeau)

Advances in Proof Theory Bern, 13 – 14 December 2013
To celebrate Gerhard Jäger's 60th birthday

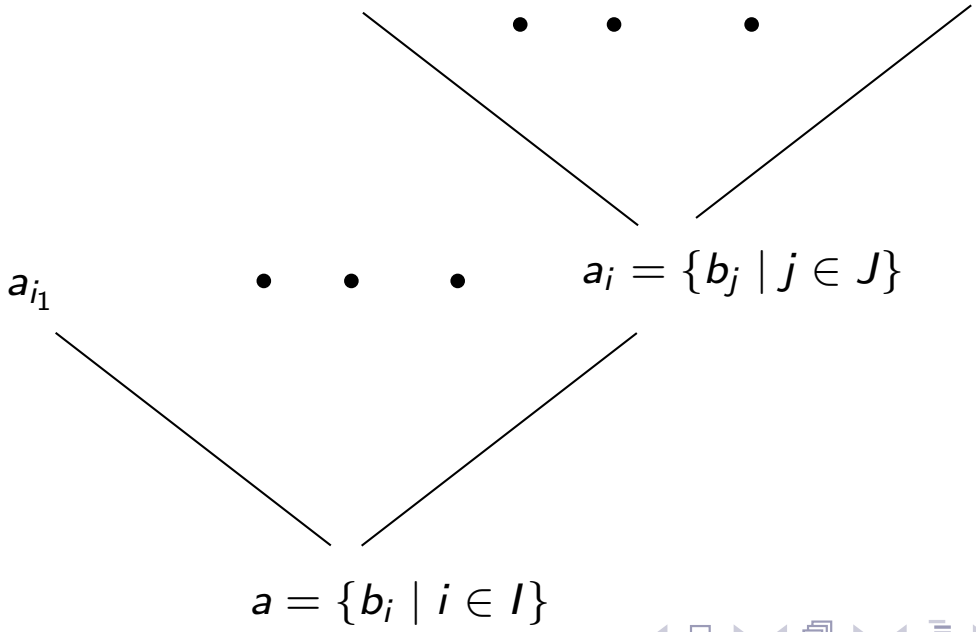
# Happy Birthday

# One Key Achievement of G. Jäger

- Shift in Proof Theory from Subsystems of Analysis to Kripke-Platek set theory.

# One Key Achievement of G. Jäger

- Shift in Proof Theory from Subsystems of Analysis to Kripke-Platek set theory.
- Reason for its success:
  - Allows to principles from recursion theory very well
    Therefore provide an excellent layering of theories of different proof theoretic strength.
  - Sets have a natural tree like structure, which corresponds well to proofs which are trees as well.



# Second Key Shift by Gerhard Jäger

- Use of Feferman's Theory of Explicit Mathematics as a laboratory for the formalisation and proof theoretic analysis of recursion principles.
- Feferman are very flexible for this purpose.
  - One example by R. Kahle and A.S.:
    Mahlo universe can be completely predicatively described in Feferman systems (extended predicative Mahlo).
- It would be nice to try out the use of EM as a programming language.

# PX

- One implementation of Feferman Systems is the PX system.
- Hayashi, Susumu and Nakano, Hiroshi: PX: a computational logic. MIT Press, 1988.

# PX

- One implementation of Feferman Systems is the PX system.
- Hayashi, Susumu and Nakano, Hiroshi: PX: a computational logic. MIT Press, 1988.
- My third year student Nathan Smith contacted the creator Susumu Hayashi:

# PX

- One implementation of Feferman Systems is the PX system.
- Hayashi, Susumu and Nakano, Hiroshi: PX: a computational logic. MIT Press, 1988.
- My third year student Nathan Smith contacted the creator Susumu Hayashi:

# PX

- One implementation of Feferman Systems is the PX system.
- Hayashi, Susumu and Nakano, Hiroshi: PX: a computational logic. MIT Press, 1988.
- My third year student Nathan Smith contacted the creator Susumu Hayashi:


Dear Nathan,

Thank you for your mail. I stopped to maintain PX system about a quarter century ago. I do not even keep the codes of the system. Sorry, but I have left computer science and software engineering long time ago.

Best regards,
Susumu Hayashi

# One more Achievement by Jäger

- **Metaprediacativity**
  - Exploration of the limit of predicative methods in proof theory.

From Codata to Coalgebras

Algebras and Coalgebras

Patterns and Copatterns

Codata types and Decidable Equality

Conclusion

Appendix: Example of Objects as Coalgebras

Appendix: Reduction of Mixed Pattern/Copattern Matching to Operators

# From Codata to Coalgebras

Algebras and Coalgebras

Patterns and Copatterns

Codata types and Decidable Equality

Conclusion

Appendix: Example of Objects as Coalgebras

Appendix: Reduction of Mixed Pattern/Copattern Matching to Operators

# Dependent Type Theory with Decidable Type Checking

- We explore the use of coalgebras in the context of dependent type theory with **decidable type checking**.
- This allows to program in dependent type theory in a similar way as when programming in other typed languages.
  - No need to **derive** that something is a correct program.
- Type checking in **dependent** type theory requires checking of (definitional) equality.
- Such an equality cannot be extensional.
- Instead, two functions $f, g : A \to B$ are equal if they (or their program codes) reduce to the same normal form.

# Codata Type

- Idea of Codata Types non-well-founded versions of inductive data types:

$$\begin{aligned} &\text{codata Stream} : \text{Set where} \\ &\quad \text{cons} \quad : \quad \mathbb{N} \to \text{Stream} \to \text{Stream} \end{aligned}$$

- Same definition as inductive data type but we are allowed to have infinite chains of constructors

$$\text{cons } n_0 \ (\text{cons } n_1 \ (\text{cons } n_2 \ \cdots))$$

- **Problem 1:** Non-normalisation.
- **Problem 2:** Equality between streams is equality between all $n_i$, and therefore undecidable.
- **Problem 3:** Underlying assumption is

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

which results in undecidable equality.

# Subject Reduction Problem

- In order to repair problem of normalisation restrictions on reductions were introduced.
- Resulted in Coq in a long known problem of **subject reduction**.
- In order to avoid this, in Agda dependent elimination for coalgebras disallowed.
  - Makes it difficult to use.

# Coalgebraic Formulation of Coalgebras

- Solution is to follow the long established categorical formulation of coalgebras.
- Final coalgebras will be replaced by weakly final coalgebras.
- Two streams will be equal if the programs producing them reduce to the same normal form.

From Codata to Coalgebras

Algebras and Coalgebras

Patterns and Copatterns

Codata types and Decidable Equality

Conclusion

Appendix: Example of Objects as Coalgebras

Appendix: Reduction of Mixed Pattern/Copattern Matching to Operators

# Algebras and Coalgebras

Elaboration of notions of (co)iteration, (co)recursion, induction is result of discussions with Peter Hancock.

- Algebraic data types correspond to initial algebras.
  - $\mathbb{N}$ as an algebra can be represented as introduction rules for $\mathbb{N}$:

$$\begin{array}{lcl} 0 & : & \mathbb{N} \\ \mathrm{S} & : & \mathbb{N} \to \mathbb{N} \end{array}$$

- Coalgebra obtained by "reversing the arrows".
  - $\mathrm{Stream}$ as a coalgebra can be expressed as as elimination rules for it:

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

# Weakly Initial Algebras and Final Coalgebras

- $\mathbb{N}$ as a weakly initial algebra corresponds to iteration (elimination rule): For $A : \mathrm{Set}$, $a : A$, $f : A \to A$ there exists

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

  (or $g\ n = f^n\ a$).

- $\mathrm{Stream}$ as a weakly final coalgebra corresponds to coiteration or guarded iteration (introduction rule):
  For $A : \mathrm{Set}$, $f_0 : A \to \mathbb{N}$, $f_1 : A \to A$ there exists $g$ s.t.

$$\begin{aligned} & g : A \to \mathrm{Stream} \\ & \mathrm{head}\ (g\ a) \quad = \quad f_0\ a \\ & \mathrm{tail}\ (g\ a) \quad = \quad g\ (f_1\ a) \end{aligned}$$

# Example

- Using coiteration we can define

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

# Recursion and Corecursion

- $\mathbb{N}$ as an initial algebra corresponds to uniqueness of $g$ above.
  - Allows to derive primitive recursion:
    For $A : \mathrm{Set}$, $a : A$, $f : (\mathbb{N} \times A) \to A$ there exists

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

- $\mathrm{Stream}$ as a final coalgebra corresponds to uniqueness of $h$.
  - Allows to derive primitive corecursion:
    For $A : \mathrm{Set}$, $f_0 : A \to \mathbb{N}$, $f_1 : A \to (\mathrm{Stream} + A)$ there exists

$$\begin{array}{lcll} g : A \to \mathrm{Stream} & & & \\ \mathrm{head}\ (g\ a) & = & f_0\ a & \\ \mathrm{tail}\ (g\ a) & = & s & \text{if } f_1\ a = \mathrm{inl}\ s \\ \mathrm{tail}\ (g\ a) & = & g\ a' & \text{if } f_1\ a = \mathrm{inr}\ a' \end{array}$$

# Recursion vs Iteration

- Using recursion we can define inverse case of the constructors of $\mathbb{N}$ as follows:

$$\begin{aligned} &\mathrm{case} : \mathbb{N} \to (1 + \mathbb{N}) \\ &\mathrm{case}\ 0 &&= \mathrm{inl} \\ &\mathrm{case}\ (\mathrm{S}\ n) &&= \mathrm{inr}\ n \end{aligned}$$

- Using iteration, we cannot make use of $n$ and therefore $\mathrm{case}$ is defined inefficiently:

$$\begin{aligned} &\mathrm{case} : \mathbb{N} \to (1 + \mathbb{N}) \\ &\mathrm{case}\ 0 &&= \mathrm{inl} \\ &\mathrm{case}\ (\mathrm{S}\ n) &&= \mathrm{caseaux}\ (\mathrm{case}\ n) \end{aligned}$$

$$\begin{aligned} &\mathrm{caseaux} : (1 + \mathbb{N}) \to (1 + \mathbb{N}) \\ &\mathrm{caseaux}\ \mathrm{inl} &&= \mathrm{inr}\ 0 \\ &\mathrm{caseaux}\ (\mathrm{inr}\ n) &&= \mathrm{inr}\ (\mathrm{S}\ n) \end{aligned}$$

# Defininition of pred

- In the talk given we defined pred in a wrong way (using iteration). One way of defining pred by iteration is by defining first case and then to define

$$\begin{aligned} &\text{predaux} : (1 + \mathbb{N}) \to \mathbb{N} \\ &\text{predaux inl} \quad = \quad 0 \\ &\text{predaux (inr } n\text{)} \quad = \quad n \end{aligned}$$

$$\begin{aligned} &\text{pred} : \mathbb{N} \to \mathbb{N} \\ &\text{pred } n = \text{predaux (case } n\text{)} \end{aligned}$$

# Corecursion vs Coiteration

- Definition of $\mathrm{cons}$ (inverse of the destructors) using coiteration inefficient:

$$\begin{aligned}
&\mathrm{cons} : \mathbb{N} \to \mathrm{Stream} \to \mathrm{Stream} \\
&\mathrm{head}\ (\mathrm{cons}\ n\ s) \quad = \quad n \\
&\mathrm{tail}\ (\mathrm{cons}\ n\ s) \quad = \quad \mathrm{cons}\ (\mathrm{head}\ s)\ (\mathrm{tail}\ s)
\end{aligned}$$

  - Using primitive corecursion we can define more easily

$$\begin{aligned}
&\mathrm{cons} : \mathbb{N} \to \mathrm{Stream} \to \mathrm{Stream} \\
&\mathrm{head}\ (\mathrm{cons}\ n\ s) \quad = \quad n \\
&\mathrm{tail}\ (\mathrm{cons}\ n\ s) \quad = \quad s
\end{aligned}$$

# Induction - Coinduction?

- Induction is dependent primitive recursion:
  For $A : \mathbb{N} \to \mathrm{Set}$, $a : A\ 0$, $f : (n : \mathbb{N}) \to A\ n \to A\ (\mathrm{S}\ n)$ there exists

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

  - Equivalent to uniqueness of arrows with respect to propositional equality and interpreting equality on arrows extensionally.
- Uniqueness of arrows in final coalgebras expresses that equality is bisimulation equality.
  - How to dualise **dependent** primitive recursion?

# Weakly Final Coalgebra

- Equality for final coalgebras is undecidable:
  Two streams
  $$\begin{array}{rcccccccc} s & = & (a_0 & , & a_1 & , & a_2 & , & \dots \\ t & = & (b_0 & , & b_1 & , & b_2 & , & \dots \end{array}$$
  are equal iff $a_i = b_i$ for all $i$.
- Even the weak assumption
  $$\forall s.\exists n, s'.s = \mathrm{cons}\ n\ s'$$
  results in an undecidable equality.
- Weakly final coalgebras obtained by omitting uniqueness of $g$ in diagram for coalgebras.
- However, one can extend schema of coiteration as above, and still preserve decidability of equality.
  - Those schemata are usually not derivable in weakly final coalgebras.

# Definition of Coalgebras by Observations

- We see now that elements of coalgebras are defined by their observations:
  An element $s$ of $\mathrm{Stream}$ is anything for which we can define

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

- This generalises the function type.
  Functions are as well determined by observations.
  - An $f : A \to B$ is any program which if applied to $a : A$ returns some $b : B$.
- **Inductive data types** are defined by **construction**
  **coalgebraic data types** and **functions** by **observations**.

# Relationship to Objects in Object-Oriented Programming

- Objects in Object-Oriented Programming are types which are defined by their observations.
- Therefore objects are coalgebraic types by nature.

From Codata to Coalgebras

Algebras and Coalgebras

Patterns and Copatterns

Codata types and Decidable Equality

Conclusion

Appendix: Example of Objects as Coalgebras

Appendix: Reduction of Mixed Pattern/Copattern Matching to Operators

# Patterns and Copatterns

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

# Patterns and Copatterns

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

$$f : \mathbb{N} \to \mathrm{Stream}$$
$$f \quad = \quad ?$$

# Patterns and Copatterns

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

$$f : \mathbb{N} \to \text{Stream}$$
$$f \quad = \quad ?$$

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

$$f : \mathbb{N} \to \text{Stream}$$
$$f\ n \quad = \quad ?$$

# Patterns and Copatterns

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

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

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

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

# Patterns and Copatterns

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

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

**Solve first case, copattern match on second case**:

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

# Patterns and Copatterns

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

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

**Solve second line, pattern match on $n$**

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

# Patterns and Copatterns

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

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

**Solve remaining cases**

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

# 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.

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

From Codata to Coalgebras

Algebras and Coalgebras

Patterns and Copatterns

Codata types and Decidable Equality

Conclusion

Appendix: Example of Objects as Coalgebras

Appendix: Reduction of Mixed Pattern/Copattern Matching to Operators

# Result

- We show that a decidable equality on $\mathrm{Stream}$ is incompatible with the assumption

$$\forall s.\exists n, s'.s == \mathrm{cons}\ n\ s'$$

# Theorem Regarding Undecidability of Equality

## Theorem

*Assume the following:*

- *There exists a subset* $\mathrm{Stream} \subseteq \mathbb{N}$,
- *computable functions*
  $\mathrm{head} : \mathrm{Stream} \to \mathbb{N}$, $\mathrm{tail} : \mathrm{Stream} \to \mathrm{Stream}$,
- *a decidable binary relation* $\_ == \_$ *on* $\mathrm{Stream}$ *which is an equivalence relation,*
- *the possibility to define elements of* $\mathrm{Stream}$ *by primitive corecursion based on primitive recursive functions* $f, g : \mathbb{N} \to \mathbb{N}$, *such that the equalities related to guarded recursion hold.*
- *All operation above respect equality* $\_ == \_$.

*Then the following does not hold:*

$$\forall s, s' \in \mathrm{Stream}.\mathrm{head}\ s == \mathrm{head}\ s' \wedge \mathrm{tail}\ s == \mathrm{tail}\ s' \to s == s' \qquad (*)$$

# Consequences for Codata Approach

## Remark

*Condition* $(*)$ *is fulfilled if we have an operation*
$\mathrm{cons} : \mathbb{N} \to \mathrm{Stream} \to \mathrm{Stream}$ *preserving equalities s.t.*

$$\forall s : \mathrm{Stream}.s == \mathrm{cons}\ (\mathrm{head}\ s)\ (\mathrm{tail}\ s)$$

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

$$\forall s.\exists n, s'.s == \mathrm{cons}\ n\ s'$$

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 $\mathrm{head}$, $\mathrm{tail}$ expressing that $s$ behaves as the stream indicated on the right hand side.

- Define by guarded recursion $I : \mathrm{Stream}$

$$I \approx 1 :: 1 :: 1 :: \cdots$$

# 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

$$f\ e \approx \underbrace{0 :: 0 :: 0 :: \cdots :: 0}_{n \text{ times}} :: I$$

$$g\ e \approx \begin{cases} \underbrace{0 :: 0 :: 0 :: \cdots :: 0}_{n \text{ times}} :: I & \text{if } k = 0 \\ \underbrace{0 :: 0 :: 0 :: \cdots :: 0}_{n+1 \text{ times}} :: I & \text{if } k > 0 \end{cases}$$

# Proof of Theorem

$$f\ e \quad \approx \quad \underbrace{0 :: 0 :: 0 :: \cdots :: 0}_{n \text{ times}} :: l$$

$$g\ e \quad \approx \quad \begin{cases} \underbrace{0 :: 0 :: 0 :: \cdots :: 0}_{n \text{ times}} :: l & \text{if } k = 0 \\ \underbrace{0 :: 0 :: 0 :: \cdots :: 0}_{n+1 \text{ times}} :: l & \text{if } k > 0 \end{cases}$$

- If $e$ terminates after $n$ steps with result 0 then

$$f\ e == g\ e$$

- If $e$ terminates after $n$ steps with result $> 0$ then

$$\neg(f\ e == g\ e)$$

# Proof of Theorem

- So

$$\lambda e.(f\ e == g\ e)$$

  separates the TM with result 0 from those with result $> 0$.
- But these two sets are inseparable.
  (See e.g. Odifreddi, 1999, p. 148, Theorem II.2.5.)

From Codata to Coalgebras

Algebras and Coalgebras

Patterns and Copatterns

Codata types and Decidable Equality

Conclusion

Appendix: Example of Objects as Coalgebras

Appendix: Reduction of Mixed Pattern/Copattern Matching to Operators

# Conclusion

- Symmetry between
  - algebras and coalgebras,
  - iteration and coiteration,
  - recursion and corecursion,
  - patterns and copatterns.
- Final algebras are defined by construction, coalgebras and function types by observation.
- Codata types make the implicit assumption

$$\forall s : \mathrm{Stream}.\exists n, s'.s = \mathrm{cons}\ n\ s'$$

  which cannot be combined with a decidable equality.
- Weakly final coalgebras solve this problem.
  - Wheras in attempts to use codata types sophisticated reduction rules were used which depend on context, in the coalgebra approach we have recursion rules which can always be applied independent of context.

From Codata to Coalgebras

Algebras and Coalgebras

Patterns and Copatterns

Codata types and Decidable Equality

Conclusion

Appendix: Example of Objects as Coalgebras

Appendix: Reduction of Mixed Pattern/Copattern Matching to Operators

# Example

```
class Angle
  { int angle;

    Angle(int myangle) {angle = myangle mod 360;};

    void set(int myangle){angle = myangle mod 360; return;};

    int get(){return angle;};
  }
```

Interface of Angle (which is the true type) can be given as

$$\begin{array}{l} \text{coalg Angle where} \\ \quad \begin{array}{lcl} \text{set} & : & \text{Angle} \to \mathbb{N} \to \text{Angle} \\ \text{get} & : & \text{Angle} \to \mathbb{N} \end{array} \end{array}$$

# Example

```
class Angle
  { int angle;

    Angle(int myangle){angle = myangle mod 360;};

    void set(int myangle){angle = myangle mod 360; return;};

    int get(){return angle;};
  }
```

Implementation of the methods can be given as
(argument $\mathbb{N}$ corresponds to the instance variable):

$$\begin{aligned} &\text{angleImpl} : \mathbb{N} \to \text{Angle} \\ &\quad \text{set}\ (\text{angleImpl}\ n)\ m \quad = \quad \text{angleImpl}\ (m \mod 360) \\ &\quad \text{get}\ (\text{angleImpl}\ n) \quad = \quad n \end{aligned}$$

# Example

```
class Angle
  { int angle;                                                    }

    Angle(int myangle){angle = myangle mod 360;};

    void set(int myangle){angle = myangle mod 360; return;};

    int get(){return angle;};
  }
```

Implementation of the constructor can be given as

$$\begin{aligned} &\mathrm{createAngle} : \mathbb{N} \to \mathrm{Angle} \\ &\mathrm{createAngle}\ n = \mathrm{angleImpl}\ n \end{aligned}$$

From Codata to Coalgebras

Algebras and Coalgebras

Patterns and Copatterns

Codata types and Decidable Equality

Conclusion

Appendix: Example of Objects as Coalgebras

Appendix: Reduction of Mixed Pattern/Copattern Matching to Operators

# Operators for Primitive (Co)Recursion

$$\mathrm{P}_{\mathbb{N},A} : A \to (\mathbb{N} \to A \to A) \to \mathbb{N} \to A$$
$$\mathrm{P}_{\mathbb{N},A}\ \mathrm{step}_0\ \mathrm{step}_\mathrm{S}\ 0 \quad = \quad \mathrm{step}_0$$
$$\mathrm{P}_{\mathbb{N},A}\ \mathrm{step}_0\ \mathrm{step}_\mathrm{S}\ (\mathrm{S}\ n) \quad = \quad \mathrm{step}_\mathrm{S}\ n\ (\mathrm{P}_{\mathbb{N},A}\ \mathrm{step}_0\ \mathrm{step}_\mathrm{S}\ n)$$

$$\mathrm{coP}_{\mathrm{Stream},A} : (A \to \mathbb{N}) \to (A \to (\mathrm{Stream} + A)) \to A \to \mathrm{Stream}$$
$$\mathrm{head}\ (\mathrm{coP}_{\mathrm{Stream},A}\ \mathrm{step}_\mathrm{head}\ \mathrm{step}_\mathrm{tail}\ a) \quad = \quad \mathrm{step}_\mathrm{head}\ a$$
$$\mathrm{tail}\ \ (\mathrm{coP}_{\mathrm{Stream},A}\ \mathrm{step}_\mathrm{head}\ \mathrm{step}_\mathrm{tail}\ a) \quad =$$
$$\mathrm{case}_{\mathrm{Stream},A,\mathrm{Stream}}\ \mathrm{id}\ (\mathrm{coP}_{\mathrm{Stream},A}\ \mathrm{step}_\mathrm{head}\ \mathrm{step}_\mathrm{tail})\ (\mathrm{step}_\mathrm{tail}\ a)$$

# Operators for full/primitive (co)recursion

$$
\begin{aligned}
&\mathrm{R}_{\mathbb{N},A} : ((\mathbb{N} \to A) \to A) \to ((\mathbb{N} \to A) \to \mathbb{N} \to A) \to \mathbb{N} \to A \\
&\mathrm{R}_{\mathbb{N},A}\ \mathrm{step}_0\ \mathrm{step}_\mathrm{S}\ 0 \quad = \quad \mathrm{step}_0\ (\mathrm{R}_{\mathbb{N},A}\ \mathrm{step}_0\ \mathrm{step}_\mathrm{S}) \\
&\mathrm{R}_{\mathbb{N},A}\ \mathrm{step}_0\ \mathrm{step}_\mathrm{S}\ (\mathrm{S}\ n) \quad = \quad \mathrm{step}_\mathrm{S}\ (\mathrm{R}_{\mathbb{N},A}\ \mathrm{step}_0\ \mathrm{step}_\mathrm{S})\ n
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{coR}_{\mathrm{Stream},A} : ((A \to \mathrm{Stream}) \to A \to \mathbb{N}) \\
&\qquad\qquad \to ((A \to \mathrm{Stream}) \to A \to \mathrm{Stream}) \\
&\qquad\qquad \to \mathrm{Stream} \\
&\mathrm{head}\ (\mathrm{coR}_{\mathrm{Stream},A}\ \mathrm{step}_\mathrm{head}\ \mathrm{step}_\mathrm{tail}\ a) \quad = \quad \mathrm{step}_\mathrm{head} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\mathrm{coR}_{\mathrm{Stream},A}\ \mathrm{step}_\mathrm{head}\ \mathrm{step}_\mathrm{tail})\ a \\
&\mathrm{tail}\ (\mathrm{coR}_{\mathrm{Stream},A}\ \mathrm{step}_\mathrm{head}\ \mathrm{step}_\mathrm{tail}\ a) \quad = \quad \mathrm{step}_\mathrm{tail} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\mathrm{coR}_{\mathrm{Stream},A}\ \mathrm{step}_\mathrm{head}\ \mathrm{step}_\mathrm{tail})\ a
\end{aligned}
$$

# Consider Example from above

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

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

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

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

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

corresponds to

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

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

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

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

corresponds to

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

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

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

# 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 $f + g$.

$$
\begin{aligned}
&f : \mathbb{N} \to \text{Stream} \\
&f\ n \qquad\qquad\qquad\qquad\qquad\qquad = \quad (f + g)\ (\underline{\text{f}}\ n)
\end{aligned}
$$

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

$$
\begin{aligned}
&k : \mathbb{N} \to \text{Stream} \\
&k\ 0 \qquad\qquad\qquad\qquad\qquad\qquad = \quad (f + g)\ (\underline{\text{f}}\ N) \\
&k\ (\text{S}\ n) \qquad\qquad\qquad\qquad\qquad = \quad (f + g)\ (\underline{\text{f}}\ n)
\end{aligned}
$$

# 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 $(f + g)(\underline{\mathrm{f}}\ (k'\ n))$.

$$
\begin{aligned}
& f : \mathbb{N} \to \text{Stream} \\
& f\ n && = && (f + g)\ (\underline{\text{f}}\ n)
\end{aligned}
$$

$$
\begin{aligned}
& (f + g) : (\underline{\text{f}}(\mathbb{N}) + \underline{\text{g}}(\mathbb{N})) \to \text{Stream} \\
& \text{head} \quad ((f + g)\ (\underline{\text{f}}\ n)) && = && n \\
& \text{head} \quad ((f + g)\ (\underline{\text{g}}\ n)) && = && n \\
& \text{tail} \quad ((f + g)\ (\underline{\text{f}}\ n)) && = && (f + g)\ (\underline{\text{g}}\ n) \\
& \text{tail} \quad ((f + g)\ (\underline{\text{f}}\ n)) && = && (f + g)\ (\underline{\text{f}}\ (k'\ n))
\end{aligned}
$$

$$
\begin{aligned}
& k' : \mathbb{N} \to \mathbb{N} \\
& k\ 0 && = && N \\
& k\ (\text{S}\ n) && = && n
\end{aligned}
$$

# Unfolding of the Pattern Matchings

- $(f + g)$ can be defined by primitive corecursion.
- $k'$ can be defined by primitive recursion.

$$f : \mathbb{N} \to \text{Stream}$$
$$f\ n \quad = \quad (f + g)\ (\underline{\text{f}}\ n)$$

$$(f + g) : (\underline{\text{f}}(\mathbb{N}) + \underline{\text{g}}(\mathbb{N})) \to \text{Stream}$$
$$(f + g) =$$
$$\text{coP}_{\text{Stream},(\underline{\text{f}}(\mathbb{N})+\underline{\text{g}}(\mathbb{N})}\ (\lambda x.\text{case}_r(x)\ \text{of}$$
$$(\underline{\text{f}}\ n) \quad \longrightarrow \quad n$$
$$(\underline{\text{g}}\ n) \quad \longrightarrow \quad n)$$
$$(\lambda x.\text{case}_r(x)\ \text{of}$$
$$(\underline{\text{f}}\ n) \quad \longrightarrow \quad \underline{\text{g}}\ n$$
$$(\underline{\text{g}}\ n) \quad \longrightarrow \quad \underline{\text{f}}\ (k'\ n))$$

$$k' : \mathbb{N} \to \mathbb{N}$$
$$k' = \text{P}_{\mathbb{N},\mathbb{N}}\ N\ (\lambda n, ih.n)$$

# Reduction to Primitive (Co)Recursion

- The case distinction can be trivially replaced by the case distinction operator.

$$f : \mathbb{N} \to \text{Stream}$$
$$f\ n \quad = \quad (f + g)\ (\underline{\text{f}}\ n)$$

$$(f + g) : (\underline{\text{f}}(\mathbb{N}) + \underline{\text{g}}(\mathbb{N})) \to \text{Stream}$$
$$(f + g) =$$
$$\text{coP}_{\text{Stream},\underline{\text{f}}(\mathbb{N})+\underline{\text{g}}(\mathbb{N})}\ (\text{case}_{\underline{\text{f}}(\mathbb{N})+\underline{\text{g}}(\mathbb{N})}\ \text{id}\ \text{id})$$
$$(\text{case}_{\underline{\text{f}}(\mathbb{N})+\underline{\text{g}}(\mathbb{N})}\ \underline{\text{g}}\ (\underline{\text{f}} \circ k'))$$

$$k' : \mathbb{N} \to \mathbb{N}$$
$$k' = \text{P}_{\mathbb{N},\mathbb{N}}\ N\ (\lambda n, ih.n)$$