Nested induction and coinduction

Nils Anders Danielsson

2014-06-13

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement n° 247219. It has also received financial support from EPSRC and (perhaps) the Swedish Research Council. This presentation does not necessarily reflect the views of the ERC or the EU. The EU is not liable for any use of the presented information.

Introduction

- Programming: Introducing hand-crafted types.
- Mathematics/logic: Defining sets/types.

Introduction

- Induction: Finite.
- Coinduction: (Potentially) infinite.
- Combinations.
  Example: Liveness properties.

Induction

Induction

Rough idea

Values of a certain type are constructed by applying certain rules. The rules must only be applied a finite number of times.

Example: The natural numbers

- $0, 1, 2, 3, ...$
- Two rules:

$$\frac{}{0 \,:\, \mathbb{N}} \qquad\qquad \frac{n \,:\, \mathbb{N}}{1 + n \,:\, \mathbb{N}}$$

Example: The natural numbers

Two *constructors*, zero and successor:

$$\frac{}{\mathsf{zero} : \mathbb{N}} \qquad \frac{n : \mathbb{N}}{\mathsf{suc}\, n : \mathbb{N}}$$

Example: The natural numbers

Two *constructors*, zero and successor:

$$\frac{}{\mathsf{zero} : \mathbb{N}} \qquad \frac{n : \mathbb{N}}{\mathsf{suc}\ n : \mathbb{N}}$$

zero
suc zero
suc (suc zero)
suc (suc (suc zero))
⋮

Example: Finite lists

$$\frac{}{\mathsf{nil} : List\ A} \qquad \frac{x : A \qquad xs : List\ A}{\mathsf{cons}\ x\ xs : List\ A}$$

nil
$\mathsf{cons}\ 0\ \mathsf{nil}$
$\mathsf{cons}\ 1\ \mathsf{nil}$
$\mathsf{cons}\ 0\ (\mathsf{cons}\ 1\ \mathsf{nil})$
$\vdots$

Writing programs

Destruction

$$program : Inductive \to Whatever$$

Values in inductive types can be destructed using *iteration*, in which each constructor is uniformly replaced by a (total) function.

Writing programs

Destruction

$$program : Inductive \to Whatever$$

Values in inductive types can be destructed using *iteration*, in which each constructor is uniformly replaced by a (total) function.

$$\textsf{cons}\ 1\ (\textsf{cons}\ 2\ (\textsf{cons}\ 3\ \textsf{nil}))$$
$$\Downarrow$$
$$add\ 1\ (add\ 2\ (add\ 3\ 0))$$

Writing programs

Destruction

$$program : Inductive \to Whatever$$

Values in inductive types can be destructed using *iteration*, in which each constructor is uniformly replaced by a (total) function.

$$\text{cons}\ 1\ (\text{cons}\ 2\ (\text{cons}\ 3\ \text{nil}))$$
$$\Downarrow$$
$$1 + (2 + (3 + 0))$$

Example: Destructing a list

$$
\begin{aligned}
sum &: List\,\mathbb{N} \to \mathbb{N} \\
sum\ \mathsf{nil} &= 0 \\
sum\ (\mathsf{cons}\ x\ xs) &= x + sum\ xs
\end{aligned}
$$

Example: Destructing a list

$$\begin{aligned} sum &: List\ \mathbb{N} \to \mathbb{N} \\ sum\ \textsf{nil} &= 0 \\ sum\ (\textsf{cons}\ x\ xs) &= x + sum\ xs \end{aligned}$$

$$sum\ (\underline{\textsf{cons}}\ 5\ (\textsf{cons}\ 3\ \textsf{nil})) =$$

Example: Destructing a list

$$\begin{array}{lcl} sum & : & List\ \mathbb{N} \to \mathbb{N} \\ sum\ \mathsf{nil} & = & 0 \\ sum\ (\mathsf{cons}\ x\ xs) & = & x + sum\ xs \end{array}$$

$$\begin{array}{ll} sum\ (\underline{\mathsf{cons}}\ 5\ (\mathsf{cons}\ 3\ \mathsf{nil})) & = \\ 5 + sum\ (\underline{\mathsf{cons}}\ 3\ \mathsf{nil}) & = \end{array}$$

Example: Destructing a list

$$
\begin{aligned}
&sum \;:\; List\,\mathbb{N} \to \mathbb{N} \\
&sum\ \mathsf{nil} &&= 0 \\
&sum\ (\mathsf{cons}\ x\ xs) &&= x + sum\ xs
\end{aligned}
$$

$$
\begin{aligned}
&sum\ (\underline{\mathsf{cons}}\ 5\ (\mathsf{cons}\ 3\ \mathsf{nil})) &&= \\
&5 + sum\ (\underline{\mathsf{cons}}\ 3\ \mathsf{nil}) &&= \\
&5 + (3 + sum\ \underline{\mathsf{nil}}) &&=
\end{aligned}
$$

Example: Destructing a list

$$\begin{aligned} sum &: List\ \mathbb{N} \to \mathbb{N} \\ sum\ \mathsf{nil} &= 0 \\ sum\ (\mathsf{cons}\ x\ xs) &= x + sum\ xs \end{aligned}$$

$$\begin{aligned} sum\ (\underline{\mathsf{cons}}\ 5\ (\mathsf{cons}\ 3\ \mathsf{nil})) &= \\ 5 + sum\ (\underline{\mathsf{cons}}\ 3\ \mathsf{nil}) &= \\ 5 + (3 + sum\ \underline{\mathsf{nil}}) &= \\ 5 + (3 + 0) &= \end{aligned}$$

Example: Destructing a list

$$\begin{array}{lcl} sum & : & List\ \mathbb{N} \to \mathbb{N} \\ sum\ \textsf{nil} & = & 0 \\ sum\ (\textsf{cons}\ x\ xs) & = & x + sum\ xs \end{array}$$

$$\begin{array}{ll} sum\ (\underline{\textsf{cons}}\ 5\ (\textsf{cons}\ 3\ \textsf{nil})) & = \\ 5 + sum\ (\underline{\textsf{cons}}\ 3\ \textsf{nil}) & = \\ 5 + (3 + sum\ \underline{\textsf{nil}}) & = \\ 5 + (3 + 0) & = \\ 8 & \end{array}$$

Iteration

Scheme for lists:

$$\begin{array}{lcl} f \ : \ List\ A \to X & & \\ f\ \mathsf{nil} & = & n \\ f\ (\mathsf{cons}\ x\ xs) & = & c\ x\ (f\ xs) \end{array}$$

$$\begin{array}{l} f\ (\mathsf{cons}\ 5\ (\mathsf{cons}\ 3\ \mathsf{nil})) \ = \\ \quad c \qquad 5\ (c \qquad 3\ n) \end{array}$$

Example: Destructing a list

$$
\begin{aligned}
&primes \;:\; List\,\mathbb{N} \to List\,\mathbb{N}\\
&primes\ \mathsf{nil} = \mathsf{nil}\\
&primes\ (\mathsf{cons}\ x\ xs) =\\
&\quad \textbf{if}\ prime\ x\ \textbf{then}\ \mathsf{cons}\ x\ (primes\ xs)\\
&\quad \phantom{\textbf{if}\ prime\ x\ }\textbf{else}\ \phantom{\mathsf{cons}\ x\ (} primes\ xs
\end{aligned}
$$

Non-example: Destructing a list

$$
\begin{aligned}
&bad : List\ A \to \mathbb{N} \\
&bad\ \mathsf{nil} &&= 0 \\
&bad\ (\mathsf{cons}\ x\ xs) &&= bad\ (\mathsf{cons}\ x\ xs)
\end{aligned}
$$

Non-example: Destructing a list

$$
\begin{aligned}
&bad \;:\; List\ A \to \mathbb{N} \\
&bad\ \textsf{nil} \;=\; 0 \\
&bad\ (\textsf{cons}\ x\ xs) \;=\; bad\ (\textsf{cons}\ x\ xs)
\end{aligned}
$$

$$
bad\ (\underline{\textsf{cons}}\ 0\ \textsf{nil}) \;=
$$

Non-example: Destructing a list

$$\begin{array}{lcl} bad & : & List\ A \to \mathbb{N} \\ bad\ \textsf{nil} & = & 0 \\ bad\ (\textsf{cons}\ x\ xs) & = & bad\ (\textsf{cons}\ x\ xs) \end{array}$$

$$\begin{array}{lc} bad\ (\underline{\textsf{cons}}\ 0\ \textsf{nil}) & = \\ bad\ (\underline{\textsf{cons}}\ 0\ \textsf{nil}) & = \end{array}$$

Non-example: Destructing a list

$$
\begin{array}{lcl}
bad & : & List\ A \to \mathbb{N} \\
bad\ \textsf{nil} & = & 0 \\
bad\ (\textsf{cons}\ x\ xs) & = & bad\ (\textsf{cons}\ x\ xs)
\end{array}
$$

$$
\begin{array}{l}
bad\ (\underline{\textsf{cons}}\ 0\ \textsf{nil})\ = \\
bad\ (\underline{\textsf{cons}}\ 0\ \textsf{nil})\ = \\
\vdots
\end{array}
$$

Non-termination

- Especially bad in (certain) logics: $2 + 2 = 5$.
- Iteration guarantees termination.
- Iteration can be awkward.
  Many other recursion schemes exist.

Induction

Inductive definitions are very common in computer science:

- Data types (functional programming).
- Predicates used to state program correctness: "the list xs contains only primes".
- Semantics (meaning) of programs.
- Syntax of programs.
- Type systems.
- ...

Coinduction

Coinduction

Dual to induction:

	Induction	Coinduction
Basic concept	Constructors	Destructors
Programs	Destruct values (recursion)	Construct values (corecursion)

Coinduction

Rough idea

Values of a certain type are *destructed* by applying certain rules. The rules must only be applied a finite number of times.

Example: Infinite streams

Two *destructors*, head and tail:

$$\frac{xs \ : \ Stream\ A}{\textsf{head}\ xs \ : \ A} \qquad \frac{xs \ : \ Stream\ A}{\textsf{tail}\ xs \ : \ Stream\ A}$$

$$\begin{array}{lcl} \textsf{head}\ xs & = & 0 \\ \textsf{head}\ (\textsf{tail}\ xs) & = & 1 \\ \textsf{head}\ (\textsf{tail}\ (\textsf{tail}\ xs)) & = & 2 \\ \vdots & & \end{array}$$

$$xs \ = \ 0, 1, 2, ...$$

Writing programs

Construction

$$program : Whatever \to Coinductive$$

Values in coinductive types can be constructed using *coiteration*, in which each destructor is uniformly replaced by a (total) function.

Coiteration

Scheme for streams:

$$
\begin{aligned}
f &: X \to \mathit{Stream}\ A \\
\textsf{head}\ (f\ x) &= h\ x \\
\textsf{tail}\ (f\ x) &= f\ (t\ x)
\end{aligned}
$$

$$
\textsf{head}\ (\textsf{tail}\ (\textsf{tail}\ (f\ x))) =
$$

$$
h\ (t\ (t\ x))
$$

Example: Constructing a stream

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

$$nats\ n \qquad\qquad = \ n, \ 1 + n, \ 2 + n, \ \ldots$$

Example: Constructing a stream

$$
\begin{aligned}
&nats \;:\; \mathbb{N} \to \mathit{Stream}\;\mathbb{N} \\
&\textsf{head}\;(nats\;n) \;=\; n \\
&\textsf{tail}\;\;\,(nats\;n) \;=\; nats\;(1+n)
\end{aligned}
$$

$$
\begin{aligned}
nats\;n \quad &= \; n,\; 1+n,\; 2+n,\; \ldots \\
nats\;(1+n) \; &= \; 1+n,\; 2+n,\; \ldots
\end{aligned}
$$

Example: Constructing a stream

$$\begin{aligned}
&\mathit{nats} \;:\; \mathbb{N} \to \mathit{Stream}\;\mathbb{N} \\
&\textsf{head}\;(\mathit{nats}\;n) \;=\; n \\
&\textsf{tail}\;\;(\mathit{nats}\;n) \;=\; \mathit{nats}\;(1+n)
\end{aligned}$$

$$\begin{aligned}
\mathit{nats}\;n \quad &= \; n,\; 1+n,\; 2+n,\; \ldots \\
\mathit{nats}\;(1+n) &= \qquad 1+n,\; 2+n,\; \ldots
\end{aligned}$$

$$\textsf{head}\;(\qquad \textsf{tail}\;(\qquad \underline{\textsf{tail}}\;(\mathit{nats}\;0))) \;=$$

Example: Constructing a stream

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

$$\begin{aligned}
nats\ n &= n,\ 1 + n,\ 2 + n,\ \ldots \\
nats\ (1 + n) &= 1 + n,\ 2 + n,\ \ldots
\end{aligned}$$

$$\begin{aligned}
\textsf{head}\ (\textsf{tail}\ (\underline{\textsf{tail}}\ (nats\ 0))) &= \\
\textsf{head}\ (\underline{\textsf{tail}}\ (nats\ (1 + 0))) &=
\end{aligned}$$

Example: Constructing a stream

$$
\begin{aligned}
&nats : \mathbb{N} \to \mathit{Stream}\ \mathbb{N} \\
&\mathsf{head}\ (nats\ n) = n \\
&\mathsf{tail}\ \ (nats\ n) = nats\ (1 + n)
\end{aligned}
$$

$$
\begin{aligned}
nats\ n &= n,\ 1 + n,\ 2 + n,\ \ldots \\
nats\ (1 + n) &= 1 + n,\ 2 + n,\ \ldots
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{head}\ (\quad \mathsf{tail}\ (\quad \underline{\mathsf{tail}}\ (nats\ 0))) = \\
&\mathsf{head}\ (\quad \underline{\mathsf{tail}}\ (nats\ (1 + \quad 0))) = \\
&\underline{\mathsf{head}}\ (nats\ (1 + \quad (1 + \quad 0))) =
\end{aligned}
$$

Example: Constructing a stream

$$
\begin{aligned}
&nats \;:\; \mathbb{N} \to \mathit{Stream}\;\mathbb{N} \\
&\textsf{head}\;(nats\;n) \;=\; n \\
&\textsf{tail}\;\;(nats\;n) \;=\; nats\;(1+n)
\end{aligned}
$$

$$
\begin{aligned}
nats\;n &= n,\;\; 1+n,\;\; 2+n,\;\; \ldots \\
nats\;(1+n) &= 1+n,\;\; 2+n,\;\; \ldots
\end{aligned}
$$

$$
\begin{aligned}
&\textsf{head}\;(\;\textsf{tail}\;(\;\underline{\textsf{tail}}\;(nats\;0))) \;= \\
&\textsf{head}\;(\;\underline{\textsf{tail}}\;(nats\;(1+0))) \;= \\
&\underline{\textsf{head}}\;(nats\;(1+(1+0))) \;= \\
&1+(1+0)
\end{aligned}
$$

Example: Constructing a stream

$$\begin{aligned} &inc \;:\; Stream\,\mathbb{N} \to Stream\,\mathbb{N} \\ &\textsf{head}\;(inc\;xs) \;=\; 1 + \textsf{head}\;xs \\ &\textsf{tail}\;\;\;(inc\;xs) \;=\; inc\;(\textsf{tail}\;xs) \end{aligned}$$

Example: Constructing a stream

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

$$\textsf{head}\ (\underline{\textsf{tail}}\ (inc\ (nats\ 0))) =$$

Example: Constructing a stream

$$\begin{aligned}
inc &: Stream\,\mathbb{N} \to Stream\,\mathbb{N} \\
\textsf{head}\ (inc\ xs) &= 1 + \textsf{head}\ xs \\
\textsf{tail}\ \ (inc\ xs) &= inc\ (\textsf{tail}\ xs)
\end{aligned}$$

$$\begin{aligned}
\textsf{head}\ (\underline{\textsf{tail}}\ (inc\ (nats\ 0))) &= \\
\underline{\textsf{head}}\ (inc\ (\textsf{tail}\ (nats\ 0))) &=
\end{aligned}$$

Example: Constructing a stream

$$\begin{aligned} inc &: Stream\,\mathbb{N} \rightarrow Stream\,\mathbb{N} \\ \textsf{head}\;(inc\;xs) &= 1 + \textsf{head}\;xs \\ \textsf{tail}\;(inc\;xs) &= inc\;(\textsf{tail}\;xs) \end{aligned}$$

$$\begin{aligned} &\textsf{head}\;(\underline{\textsf{tail}}\;(inc\;(nats\;0))) &= \\ &\underline{\textsf{head}}\;(inc\;(\textsf{tail}\;(nats\;0))) &= \\ &1 + \textsf{head}\;(\textsf{tail}\;(nats\;0)) &= \end{aligned}$$

Example: Constructing a stream

$$
\begin{aligned}
inc &: Stream\,\mathbb{N} \rightarrow Stream\,\mathbb{N} \\
\textsf{head}\ (inc\ xs) &= 1 + \textsf{head}\ xs \\
\textsf{tail}\ \ (inc\ xs) &= inc\ (\textsf{tail}\ xs)
\end{aligned}
$$

$$
\begin{aligned}
&\textsf{head}\ (\underline{\textsf{tail}}\ (inc\ (nats\ 0))) &=\\
&\underline{\textsf{head}}\ (inc\ (\textsf{tail}\ (nats\ 0))) &=\\
&1 + \textsf{head}\ (\textsf{tail}\ (nats\ 0)) &=\\
&1 + 1 &
\end{aligned}
$$

Non-example: Constructing a stream

$$
\begin{aligned}
bad &: Stream\,\mathbb{N} \\
\textsf{head}\ bad &= 0 \\
\textsf{tail}\ \ bad &= \textsf{tail}\ bad
\end{aligned}
$$

Non-example: Constructing a stream

$$\begin{aligned} bad &: Stream\, \mathbb{N} \\ \textsf{head}\; bad &= 0 \\ \textsf{tail}\;\; bad &= \textsf{tail}\; bad \end{aligned}$$

$$\textsf{head}\; (\underline{\textsf{tail}}\; bad) \;=$$

Non-example: Constructing a stream

$$
\begin{aligned}
&bad \ : \ Stream\,\mathbb{N} \\
&\textsf{head}\ bad \ = \ 0 \\
&\textsf{tail}\ \ bad \ = \ \textsf{tail}\ bad
\end{aligned}
$$

$$
\begin{aligned}
&\textsf{head}\ (\underline{\textsf{tail}}\ bad) \ = \\
&\textsf{head}\ (\underline{\textsf{tail}}\ bad) \ =
\end{aligned}
$$

Non-example: Constructing a stream

$$
\begin{aligned}
&bad \;:\; Stream\,\mathbb{N} \\
&\textsf{head}\;bad \;=\; 0 \\
&\textsf{tail}\;\;\;bad \;=\; \textsf{tail}\;bad
\end{aligned}
$$

$$
\begin{aligned}
&\textsf{head}\;(\underline{\textsf{tail}}\;bad) \;= \\
&\textsf{head}\;(\underline{\textsf{tail}}\;bad) \;= \\
&\vdots
\end{aligned}
$$

Coinduction using constructors

Coinduction

Rough idea

Values of a certain type are constructed by applying certain rules.

Example: Infinite streams

$$\frac{x : A \qquad xs : Stream\ A}{\mathsf{cons}\ x\ xs : Stream\ A}$$

$$\mathsf{cons}\ 0\ (\mathsf{cons}\ 1\ (\mathsf{cons}\ 2\ (\mathsf{cons}\ 3\ ...)))$$
$$\mathsf{cons}\ 2\ (\mathsf{cons}\ 3\ (\mathsf{cons}\ 5\ (\mathsf{cons}\ 7\ ...)))$$
$$\vdots$$

Destructors

$$\begin{aligned} \mathsf{head}\ (\mathsf{cons}\ x\ xs) &= x \\ \mathsf{tail}\ (\mathsf{cons}\ x\ xs) &= xs \end{aligned}$$

Coiteration

Scheme for streams:

$$\begin{aligned} f &: X \to \mathit{Stream}\ A \\ f\ x &= \textsf{cons}\ (h\ x)\ (f\ (t\ x)) \end{aligned}$$

$$\begin{aligned} &f\ x &= \\ &\textsf{cons}\ (h\ x)\ (f\ (t\ x)) &= \\ &\textsf{cons}\ (h\ x)\ (\textsf{cons}\ (h\ (t\ x))\ (f\ (t\ (t\ x)))) &= \\ &\vdots \end{aligned}$$

Coiteration

Productivity

It is always possible to compute the next constructor in a finite number of steps.

Example: Constructing a stream

$$\begin{aligned} nats &: \mathbb{N} \to Stream\, \mathbb{N} \\ nats\, n &= \textsf{cons}\, n\, (nats\, (1 + n)) \end{aligned}$$

Example: Constructing a stream

$$
\begin{aligned}
nats &: \mathbb{N} \to Stream\,\mathbb{N} \\
nats\,n &= \textsf{cons}\,n\,(nats\,(1+n))
\end{aligned}
$$

$$
nats\,0 \qquad\qquad =
$$

Example: Constructing a stream

$$nats : \mathbb{N} \to Stream\ \mathbb{N}$$
$$nats\ n = \textsf{cons}\ n\ (nats\ (1 + n))$$

$$nats\ 0 =$$
$$\textsf{cons}\ 0\ (nats\ 1) =$$

Example: Constructing a stream

$$nats : \mathbb{N} \to Stream\ \mathbb{N}$$
$$nats\ n = \textsf{cons}\ n\ (nats\ (1 + n))$$

$$nats\ 0 =$$
$$\textsf{cons}\ 0\ (nats\ 1) =$$
$$\textsf{cons}\ 0\ (\textsf{cons}\ 1\ (nats\ 2)) =$$

Example: Constructing a stream

$$\begin{aligned} nats &: \mathbb{N} \to Stream\ \mathbb{N} \\ nats\ n &= \textsf{cons}\ n\ (nats\ (1 + n)) \end{aligned}$$

$$\begin{aligned} &nats\ 0 &= \\ &\textsf{cons}\ 0\ (nats\ 1) &= \\ &\textsf{cons}\ 0\ (\textsf{cons}\ 1\ (nats\ 2)) &= \\ &\textsf{cons}\ 0\ (\textsf{cons}\ 1\ (\textsf{cons}\ 2\ (nats\ 3))) &= \end{aligned}$$

Example: Constructing a stream

$$
\begin{aligned}
nats &: \mathbb{N} \to \mathit{Stream}\ \mathbb{N} \\
nats\ n &= \textsf{cons}\ n\ (nats\ (1 + n))
\end{aligned}
$$

$$
\begin{aligned}
&nats\ 0 &= \\
&\textsf{cons}\ 0\ (nats\ 1) &= \\
&\textsf{cons}\ 0\ (\textsf{cons}\ 1\ (nats\ 2)) &= \\
&\textsf{cons}\ 0\ (\textsf{cons}\ 1\ (\textsf{cons}\ 2\ (nats\ 3))) &= \\
&\vdots
\end{aligned}
$$

Example: Constructing a stream

$$inc \ : \ Stream\,\mathbb{N} \rightarrow Stream\,\mathbb{N}$$
$$inc\ (\textsf{cons}\ x\ xs) \ = \ \textsf{cons}\ (1 + x)\ (inc\ xs)$$

Example: Constructing a stream

$$\begin{aligned} inc &: Stream\,\mathbb{N} \to Stream\,\mathbb{N} \\ inc\,(\textsf{cons}\;x\;xs) &= \textsf{cons}\;(1+x)\;(inc\;xs) \end{aligned}$$

$$\begin{aligned} inc &: Stream\,\mathbb{N} \to Stream\,\mathbb{N} \\ \textsf{head}\;(inc\;xs) &= 1 + \textsf{head}\;xs \\ \textsf{tail}\;\;(inc\;xs) &= inc\;(\textsf{tail}\;xs) \end{aligned}$$

Non-example: Constructing a stream

$$
\begin{array}{l}
\mathit{primes} \ : \ \mathit{Stream}\ \mathbb{N} \to \mathit{Stream}\ \mathbb{N} \\
\mathit{primes}\ (\mathsf{cons}\ x\ xs) \ = \\
\quad \textbf{if}\ \mathit{prime}\ x\ \textbf{then}\ \mathsf{cons}\ x\ (\mathit{primes}\ xs) \\
\qquad\qquad\qquad \textbf{else} \qquad\qquad \mathit{primes}\ xs
\end{array}
$$

Example: Potentially infinite lists

$$\frac{}{\text{nil} : Colist\ A} \qquad \frac{x : A \qquad xs : Colist\ A}{\text{cons}\ x\ xs : Colist\ A}$$

$$\begin{aligned}
&\text{nil}\\
&\text{cons}\ 0\ \text{nil}\\
&\text{cons}\ 0\ (\text{cons}\ 1\ \text{nil})\\
&\vdots\\
&\text{cons}\ 0\ (\text{cons}\ 1\ (\text{cons}\ 2\ (\text{cons}\ 3\ ...)))\\
&\vdots
\end{aligned}$$

Coinduction

Examples of uses of coinductive definitions in computer science:

- Data types.
- Predicates used to state program correctness: "the stream xs contains only primes".
- Modelling of abstract data types.
- Non-terminating programs in total languages.
- Semantics (meaning) of non-termination.
- ...

Nested induction and coinduction

Example: Stream processors

- SP: Representation of stream processors.
- $run : SP \to Stream\ Bit \to Stream\ Bit$

Example: Stream processors

$$\frac{x \;:\; SP \qquad y \;:\; SP}{\mathsf{get}\; x\; y \;:\; SP} \qquad \frac{b \;:\; Bit \qquad x \;:\; SP}{\mathsf{put}\; b\; x \;:\; SP}$$

- get x y: Read one bit, continue as x if 0, y if 1.
- put b x: Write b, continue as x.

Example: Stream processors

$$\begin{aligned} copy &: SP \\ copy &= \mathsf{get}\ (\mathsf{put}\ 0\ copy)\ (\mathsf{put}\ 1\ copy) \end{aligned}$$

$$\begin{aligned} not &: SP \\ not &= \mathsf{get}\ (\mathsf{put}\ 1\ not)\ (\mathsf{put}\ 0\ not) \end{aligned}$$

Example: Stream processors

$$\begin{aligned} copy &: SP \\ copy &= \mathsf{get}\ (\mathsf{put}\ 0\ copy)\ (\mathsf{put}\ 1\ copy) \end{aligned}$$

$$\begin{aligned} not &: SP \\ not &= \mathsf{get}\ (\mathsf{put}\ 1\ not)\ (\mathsf{put}\ 0\ not) \end{aligned}$$

Are these definitions OK?

Example: Stream processors

How should this mixed definition be interpreted?

$$\frac{x : SP \qquad y : SP}{\textsf{get}\ x\ y : SP} \qquad \frac{b : Bit \qquad x : SP}{\textsf{put}\ b\ x : SP}$$

Example: Stream processors

How should this mixed definition be interpreted?

$$\frac{x : SP \qquad y : SP}{\mathsf{get}\ x\ y : SP} \qquad \frac{b : Bit \qquad x : SP}{\mathsf{put}\ b\ x : SP}$$

Outer inductive definition, inner coinductive one:

- Only finite number of gets.
- Cannot define $copy$ or not.

Example: Stream processors

How should this mixed definition be interpreted?

$$\frac{x : SP \qquad y : SP}{\textsf{get}\ x\ y : SP} \qquad \frac{b : Bit \qquad x : SP}{\textsf{put}\ b\ x : SP}$$

Outer coinductive definition, inner inductive one:

- Only finite number of *consecutive* gets.
- Total number of gets can be infinite.
- Can define $copy$ and not.

Example: Stream processors

$$
\begin{aligned}
run &: SP \to Stream\ Bit \to Stream\ Bit \\
run\ (\mathsf{get}\ x\ y)\ (\mathsf{cons}\ 0\ bs) &= run\ x\ bs \\
run\ (\mathsf{get}\ x\ y)\ (\mathsf{cons}\ 1\ bs) &= run\ y\ bs \\
run\ (\mathsf{put}\ b\ x)\ bs &= \mathsf{cons}\ b\ (run\ x\ bs)
\end{aligned}
$$

Example: Stream processors

What if get were coinductive?

$$\frac{x : SP \qquad y : SP}{\textsf{get}\; x\; y : SP} \qquad \frac{b : Bit \qquad x : SP}{\textsf{put}\; b\; x : SP}$$

Could define $sink$:

$$\begin{aligned} sink &: SP \\ sink &= \textsf{get}\; sink\; sink \end{aligned}$$

Could not define run: not productive.

Example: Stream processors

By combining induction and coinduction,
rather than using only coinduction:

- Fewer stream processors allowed.
- But can define run.

Trade-off: Less data, more functions.

Nested induction and coinduction

Other examples:

- Parser combinators.
- Program equivalences.
- Subtyping.
- ...

Summary

- Induction: Finite.
- Coinduction: (Potentially) infinite.
- Nested induction and coinduction:
  Both finite and infinite.
  Precise control over size of data.

Bonus slides

Example: Potentially infinite lists

$$\frac{}{\mathsf{nil} \;:\; Colist_{\mathsf{i}}\ A\ O} \qquad \frac{x \;:\; A \qquad xs \;:\; O}{\mathsf{cons}\ x\ xs \;:\; Colist_{\mathsf{i}}\ A\ O}$$

$$\frac{xs \;:\; Colist\ A}{\mathsf{destruct}\ xs \;:\; Colist_{\mathsf{i}}\ A\ (Colist\ A)}$$

$$\begin{aligned}
\mathsf{destruct}\ xs &= \mathsf{cons}\ 0\ ys \\
\mathsf{destruct}\ ys &= \mathsf{cons}\ 1\ zs \\
\mathsf{destruct}\ zs &= \mathsf{nil}
\end{aligned}$$

$$xs = 0, 1$$

Example: Stream processors

Outer coinductive definition, inner inductive one:

$$\frac{x \,:\, SP \text{¡}\, O \qquad y \,:\, SP \text{¡}\, O}{\mathsf{get}\; x\; y \,:\, SP \text{¡}\, O} \qquad \frac{b \,:\, Bit \qquad x \,:\, O}{\mathsf{put}\; b\; x \,:\, SP \text{¡}\, O}$$

$$\frac{x \,:\, SP}{\mathsf{destruct}\; x \,:\, SP \text{¡}\, SP}$$

Example: Stream processors

Outer inductive definition, inner coinductive one:

$$\frac{x : O \qquad y : O}{\mathsf{get}\ x\ y : SP_{\text{¡}}\ O} \qquad \frac{b : Bit \qquad x : SP_{\text{¡}}\ O}{\mathsf{put}\ b\ x : SP_{\text{¡}}\ O}$$

$$\frac{x : SP_{\text{¡}}\ SP}{\mathsf{construct}\ x : SP}$$

Example:
Bit streams with a finite number of ones

Outer inductive definition, inner coinductive one:

$$\frac{xs \;:\; Stream}{\textsf{cons-zero}\; xs \;:\; Stream} \qquad \frac{xs \;:\; Stream}{\textsf{cons-one}\; xs \;:\; Stream}$$