Lecture 4: Inductive and Coinductive Datatypes

Andrei Popescu

University of Sheffield

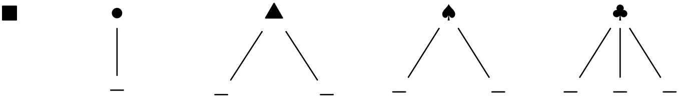
MGS 21

16 April, 2021

Bounded Natural Functors (BNFs)

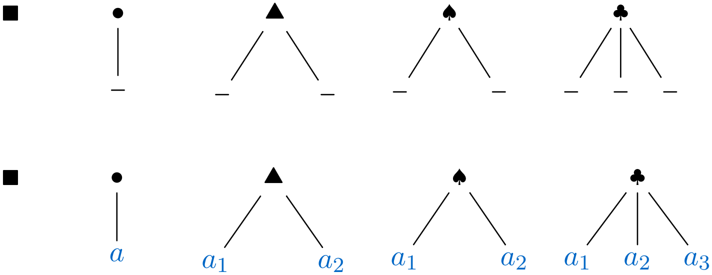
Preliminariers: It's All About Shape and Content

Shapes



Preliminariers: It's All About Shape and Content

Shapes



Shapes filled with content from a set $A = \{a_1, a_2, \ldots\}$

Natural Functors on Set

Set = the class of all sets

Natural Functors on Set

$\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ is a natural functor if:

Natural Functors on Set

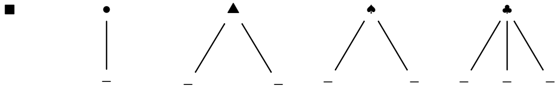
$\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ is a natural functor if:

It comes with a set of shapes

Natural Functors on Set

$F : \mathsf{Set} \to \mathsf{Set}$ is a natural functor if:

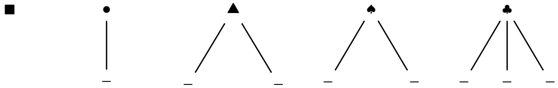
It comes with a set of shapes, say



Natural Functors on Set

$\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ is a natural functor if:

It comes with a set of shapes, say



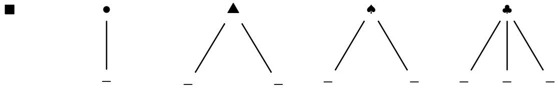
Each element $x \in \mathsf{F}\ A$ consists of:

a choice of a shape

Natural Functors on Set

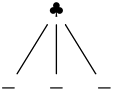
F : Set → Set is a natural functor if:

It comes with a set of shapes, say



Each element $x \in \mathsf{F}\, A$ consists of:

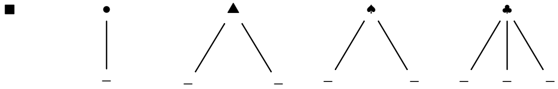
a choice of a shape, say



Natural Functors on Set

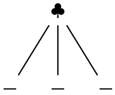
F : Set → Set is a natural functor if:

It comes with a set of shapes, say



Each element $x \in \mathsf{F}\,A$ consists of:

a choice of a shape, say

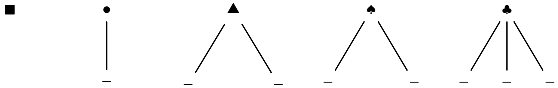


a filling with content from A

Natural Functors on Set

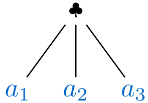
F : Set → Set is a natural functor if:

It comes with a set of shapes, say



Each element $x \in \mathsf{F}\,A$ consists of:

a choice of a shape, say



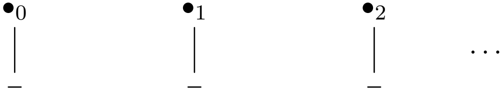
a filling with content from A, say

Examples of Natural Functors

$\mathsf{F}\ A = \mathbb{N} \times A$

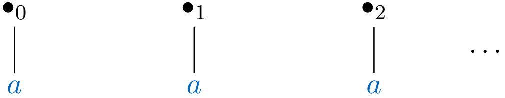
Examples of Natural Functors

$\mathsf{F}\ A = \mathbb{N} \times A$



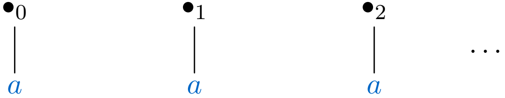
Examples of Natural Functors

$\mathsf{F}\ A = \mathbb{N} \times A$



Examples of Natural Functors

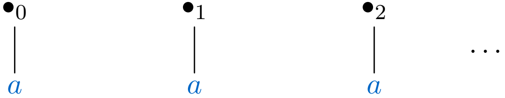
$\mathsf{F}\ A = \mathbb{N} \times A$



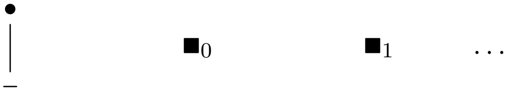
$\mathsf{F}\ A = \mathbb{N} + A$

Examples of Natural Functors

$\mathsf{F}\ A = \mathbb{N} \times A$

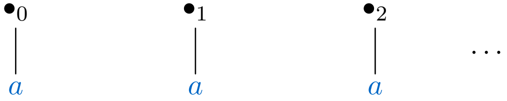


$\mathsf{F}\ A = \mathbb{N} + A$

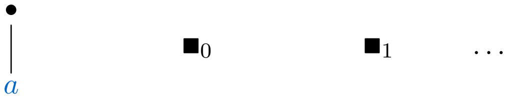


Examples of Natural Functors

$\mathsf{F}\ A = \mathbb{N} \times A$

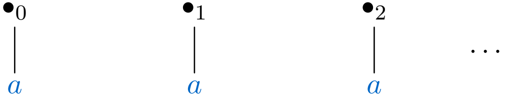


$\mathsf{F}\ A = \mathbb{N} + A$

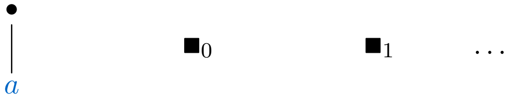


Examples of Natural Functors

$\mathsf{F}\ A = \mathbb{N} \times A$



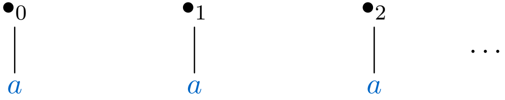
$\mathsf{F}\ A = \mathbb{N} + A$



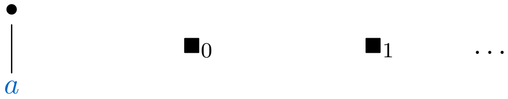
$\mathsf{F}\ A = \mathsf{List}\ A$

Examples of Natural Functors

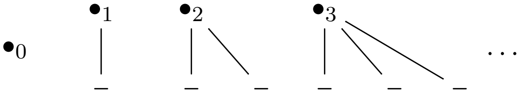
F A = $\mathbb{N} \times A$



F A = $\mathbb{N} + A$

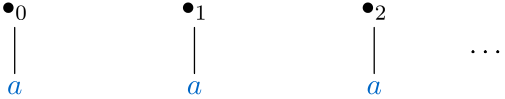


F A = List A

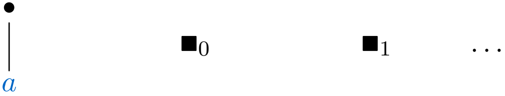


Examples of Natural Functors

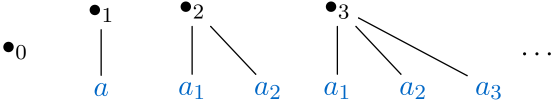
F A = $\mathbb{N} \times A$



F A = $\mathbb{N} + A$

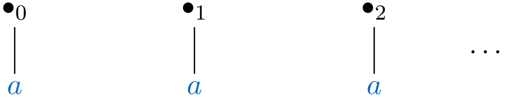


F A = List A

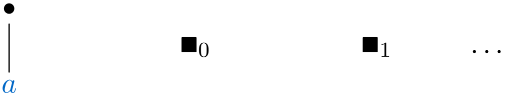


Examples of Natural Functors

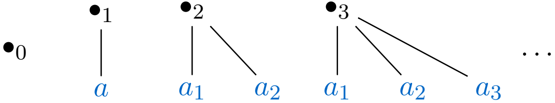
$\mathsf{F}\ A = \mathbb{N} \times A$



$\mathsf{F}\ A = \mathbb{N} + A$



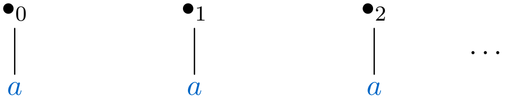
$\mathsf{F}\ A = \mathsf{List}\ A$



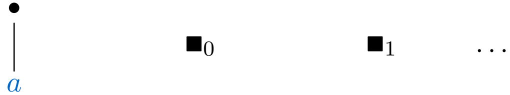
$\mathsf{F}\ A = \mathsf{Stream}\ A$?

Examples of Natural Functors

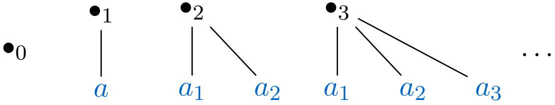
$\mathsf{F}\ A = \mathbb{N} \times A$



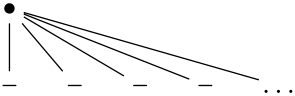
$\mathsf{F}\ A = \mathbb{N} + A$



$\mathsf{F}\ A = \mathsf{List}\ A$

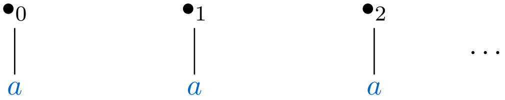


$\mathsf{F}\ A = \mathsf{Stream}\ A$

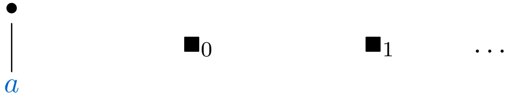


Examples of Natural Functors

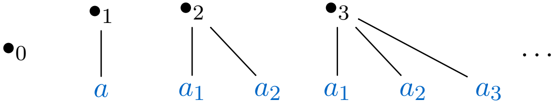
$\mathsf{F}\ A = \mathbb{N} \times A$



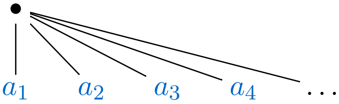
$\mathsf{F}\ A = \mathbb{N} + A$



$\mathsf{F}\ A = \mathsf{List}\ A$



$\mathsf{F}\ A = \mathsf{Stream}\ A$

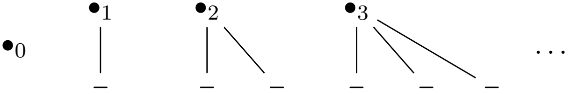


Examples of Natural Functors

$\mathsf{F}\ A = \mathsf{LazyList}\ A$?

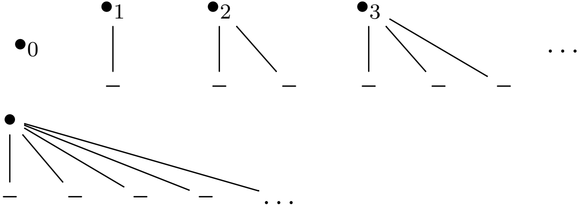
Examples of Natural Functors

F A = LazyList A = List A



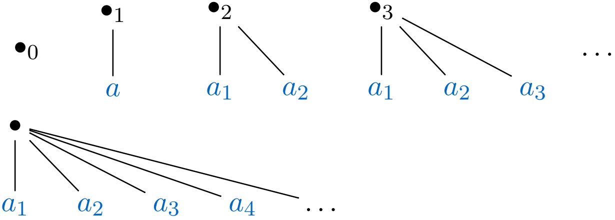
Examples of Natural Functors

F A = LazyList A = List A ∪ Stream A



Examples of Natural Functors

$\mathsf{F}\ A = \mathsf{LazyList}\ A\ \ = \mathsf{List}\ A \cup \mathsf{Stream}\ A$

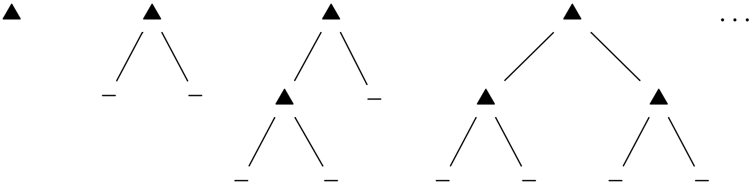


Examples of Natural Functors

F A = BTree A (Full Binary Trees with leaves in A)

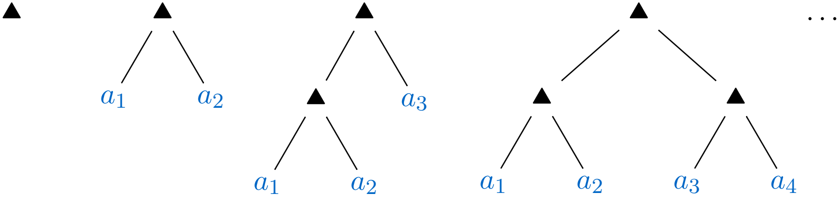
Examples of Natural Functors

F A = BTree A (Full Binary Trees with leaves in A)



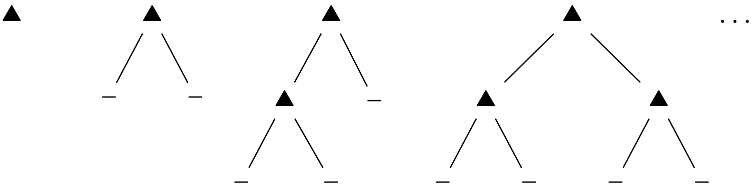
Examples of Natural Functors

F A = BTree A (Full Binary Trees with leaves in A)



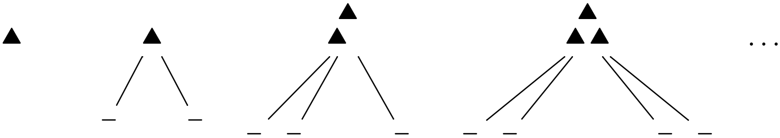
Examples of Natural Functors

F A = BTree A (Full Binary Trees with leaves in A)



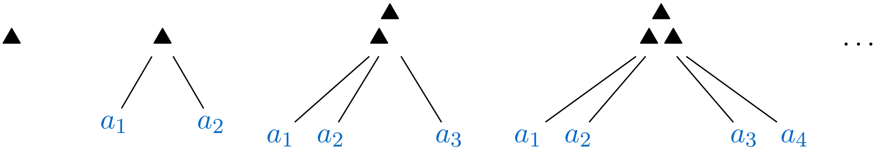
Examples of Natural Functors

F A = BTree A (Full Binary Trees with leaves in A)



Examples of Natural Functors

F A = BTree A (Full Binary Trees with leaves in A)



Functorial Action (Mapper)

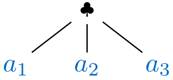
$$\begin{tikzcd}
A \arrow[d, "f"] \\
B
\end{tikzcd}$$

Functorial Action (Mapper)

$$\begin{tikzcd} A \arrow[d, "f"] & \mathsf{F}\,A \arrow[d, "\mathsf{F}\,f"] \\ B & \mathsf{F}\,B \end{tikzcd}$$

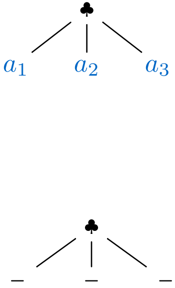
Functorial Action (Mapper)

$$\begin{tikzcd}
A \arrow[d, "f"] & \mathsf{F}\,A \arrow[d, "\mathsf{F}\,f"] \\
B & \mathsf{F}\,B
\end{tikzcd}$$



Functorial Action (Mapper)

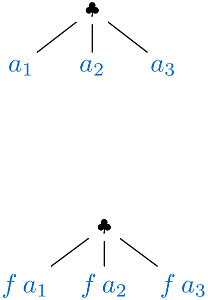
$$\begin{tikzcd} A \arrow[d, "f"] & \mathsf{F}\,A \arrow[d, "\mathsf{F}\,f"] \\ B & \mathsf{F}\,B \end{tikzcd}$$



Keep the same shape

Functorial Action (Mapper)

$$\begin{tikzcd} A \arrow[d, "f"] \\ B \end{tikzcd} \qquad \begin{tikzcd} \mathsf{F}\, A \arrow[d, "\mathsf{F}\, f"] \\ \mathsf{F}\, B \end{tikzcd}$$

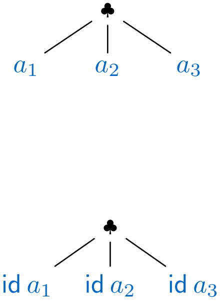


Keep the same shape

Apply f to the content

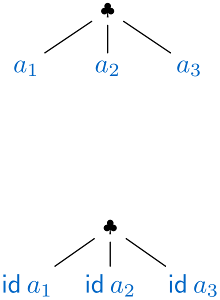
Commutation with the Identity Function

$$\begin{tikzcd}
A \arrow[d, "\mathsf{id}"] & \mathsf{F}\,A \arrow[d, "\mathsf{F}\,\mathsf{id}"] \\
A & \mathsf{F}\,A
\end{tikzcd}$$



Commutation with the Identity Function

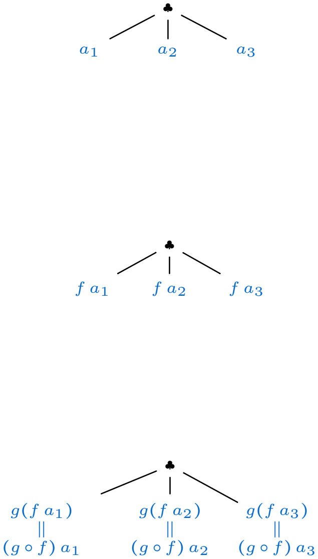
$$\begin{tikzcd}
A \arrow[d, "\mathsf{id}"] & \mathsf{F}\,A \arrow[d, "\mathsf{F}\,\mathsf{id} = \mathsf{id}"] \\
A & \mathsf{F}\,A
\end{tikzcd}$$



Commutation with Function Composition

$$\begin{tikzcd}
A \arrow[d, "f"] \\
B \arrow[d, "g"] \\
C
\end{tikzcd}$$

$$\begin{tikzcd}
\mathsf{F}\,A \arrow[d, "\mathsf{F}\,f"] \arrow[dd, bend right, "\mathsf{F}\,(g\circ f)"'] \\
\mathsf{F}\,B \arrow[d, "\mathsf{F}\,g"] \\
\mathsf{F}\,C
\end{tikzcd}$$


♣
a_1
a_2
a_3
♣
$f\ a_1$
$f\ a_2$
$f\ a_3$
♣
$g(f\ a_1)$ $\|$ $(g\circ f)\ a_1$
$g(f\ a_2)$ $\|$ $(g\circ f)\ a_2$
$g(f\ a_3)$ $\|$ $(g\circ f)\ a_3$

Bottom Line

$\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

For all $A \xrightarrow{f} B$, we have $\mathsf{F}\,A \xrightarrow{\mathsf{F}\,f} \mathsf{F}\,B$ such that:

$\mathsf{F}\,\mathsf{id}_A = \mathsf{id}_{\mathsf{F}A}$
$\mathsf{F}\,(g \circ f) = \mathsf{F}\,g \circ \mathsf{F}\,f$

Bottom Line

$\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

For all $A \xrightarrow{f} B$, we have $\mathsf{F}\,A \xrightarrow{\mathsf{F}\,f} \mathsf{F}\,B$ such that:

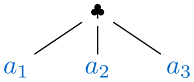
$$\begin{aligned} &\mathsf{F}\,\mathsf{id}_A = \mathsf{id}_{\mathsf{F}A} \\ &\mathsf{F}\,(g \circ f) = \mathsf{F}\,g \circ \mathsf{F}\,f \end{aligned}$$

Functoriality

Atoms

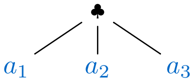
$$\begin{tikzcd}\mathsf{F}\,A \arrow[r, "\mathsf{Fset}_A"] & \mathcal{P}\,A\end{tikzcd}$$

Atoms



$$\begin{tikzcd}\mathsf{F}\,A \arrow[r, "\mathsf{Fset}_A"] & \mathcal{P}\,A\end{tikzcd}$$

Atoms



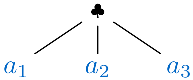
$\{a_1, a_2, a_3\}$

$$\begin{tikzcd}
\mathsf{F}\,A \arrow[r, "\mathsf{Fset}_A"] & \mathcal{P}\,A
\end{tikzcd}$$

Atoms

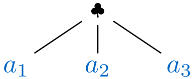
$$\begin{tikzcd}
\mathsf{F}\,A \arrow[r, "\mathsf{Fset}_A"] \arrow[d, "\mathsf{F}\,f"'] & \mathcal{P}\,A \arrow[d, "\mathsf{image}\,f"] \\
\mathsf{F}\,B \arrow[r, "\mathsf{Fset}_B"] & \mathcal{P}\,B
\end{tikzcd}$$

Atoms



$$\begin{tikzcd}
\mathsf{F}\,A \arrow[r, "\mathsf{Fset}_A"] \arrow[d, "\mathsf{F}\,f"'] & \mathcal{P}\,A \arrow[d, "\mathsf{image}\,f"] \\
\mathsf{F}\,B \arrow[r, "\mathsf{Fset}_B"] & \mathcal{P}\,B
\end{tikzcd}$$

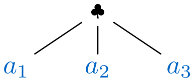
Atoms



$\{a_1, a_2, a_3\}$

$$\begin{tikzcd}
\mathsf{F}\,A \arrow[r, "\mathsf{Fset}_A"] \arrow[d, "\mathsf{F}\,f"'] & \mathcal{P}\,A \arrow[d, "\mathsf{image}\,f"] \\
\mathsf{F}\,B \arrow[r, "\mathsf{Fset}_B"] & \mathcal{P}\,B
\end{tikzcd}$$

Atoms

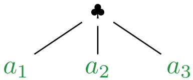


$\{a_1, a_2, a_3\}$

$$\begin{tikzcd}
\mathsf{F}\,A \arrow[r, "\mathsf{Fset}_A"] \arrow[d, "\mathsf{F}\,f"'] & \mathcal{P}\,A \arrow[d, "\mathsf{image}\,f"] \\
\mathsf{F}\,B \arrow[r, "\mathsf{Fset}_B"] & \mathcal{P}\,B
\end{tikzcd}$$

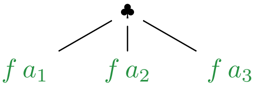
$\{f\,a_1, f\,a_2, f\,a_3\}$

Atoms



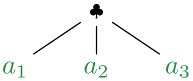
$\{a_1, a_2, a_3\}$

$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, "\mathsf{Fset}_A"] \arrow[d, "\mathsf{F}\ f"'] & \mathcal{P}\ A \arrow[d, "\mathsf{image}\ f"] \\
\mathsf{F}\ B \arrow[r, "\mathsf{Fset}_B"] & \mathcal{P}\ B
\end{tikzcd}$$



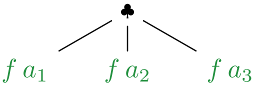
$\{f\ a_1, f\ a_2, f\ a_3\}$

Atoms



$\{a_1, a_2, a_3\}$

$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, "\mathsf{Fset}_A"] \arrow[d, "\mathsf{F}\ f"'] & \mathcal{P}\ A \arrow[d, "\mathsf{image}\ f"] \\
\mathsf{F}\ B \arrow[r, "\mathsf{Fset}_B"] & \mathcal{P}\ B
\end{tikzcd}$$



$\{f\ a_1, f\ a_2, f\ a_3\}$

Bottom Line

$\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

For all $A \overset{f}{\to} B$, we have $\mathsf{F}\,A \xrightarrow{\mathsf{F}\,f} \mathsf{F}\,B$ such that:

$$\begin{aligned} &\mathsf{F}\,\mathsf{id}_A = \mathsf{id}_{\mathsf{F}A} \\ &\mathsf{F}\,(g \circ f) = \mathsf{F}\,g \circ \mathsf{F}\,f \end{aligned}$$

Functoriality

Bottom Line

$\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

For all $A \xrightarrow{f} B$, we have $\mathsf{F}\,A \xrightarrow{\mathsf{F}\,f} \mathsf{F}\,B$ such that:

$$\begin{aligned}\mathsf{F}\,\mathsf{id}_A &= \mathsf{id}_{\mathsf{F}A}\\ \mathsf{F}\,(g \circ f) &= \mathsf{F}\,g \circ \mathsf{F}\,f\end{aligned}$$

Functoriality

For all A, we have $\mathsf{F}\,A \xrightarrow{\mathsf{Fset}_A} \mathcal{P}\,A$ such that, for all $A \xrightarrow{f} B$:

$$\mathsf{image}\,f \circ \mathsf{Fset}_A = \mathsf{Fset}_B \circ \mathsf{image}\,f$$

Bottom Line

$\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

For all $A \xrightarrow{f} B$, we have $\mathsf{F}\,A \xrightarrow{\mathsf{F}\,f} \mathsf{F}\,B$ such that:

$$\begin{aligned} &\mathsf{F}\,\mathsf{id}_A = \mathsf{id}_{\mathsf{F}A} \\ &\mathsf{F}\,(g \circ f) = \mathsf{F}\,g \circ \mathsf{F}\,f \end{aligned}$$

Functoriality

For all A, we have $\mathsf{F}\,A \xrightarrow{\mathsf{Fset}_A} \mathcal{P}\,A$ such that, for all $A \xrightarrow{f} B$:

$$\mathsf{image}\ f \circ \mathsf{Fset}_A = \mathsf{Fset}_B \circ \mathsf{image}\ f$$

Naturality

Bottom Line: Natural Functors

$\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

For all $A \xrightarrow{f} B$, we have $\mathsf{F}\,A \xrightarrow{\mathsf{F}\,f} \mathsf{F}\,B$ such that:

$$\begin{aligned} \mathsf{F}\,\mathsf{id}_A &= \mathsf{id}_{\mathsf{F}\,A} \\ \mathsf{F}\,(g \circ f) &= \mathsf{F}\,g \circ \mathsf{F}\,f \end{aligned} \qquad \text{Functoriality}$$

For all A, we have $\mathsf{F}\,A \xrightarrow{\mathsf{Fset}_A} \mathcal{P}\,A$ such that, for all $A \xrightarrow{f} B$:

$$\mathsf{image}\ f \circ \mathsf{Fset}_A = \mathsf{Fset}_B \circ \mathsf{image}\ f \qquad \text{Naturality}$$

Examples of Natural Functors

$$A \xrightarrow{f} B$$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\,A \xrightarrow{\mathsf{F}\,f} \mathsf{F}\,B$$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$\mathsf{F}\ A = \mathbb{N} \times A$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$$\mathsf{F}\ A = \mathbb{N} \times A \qquad \mathsf{F} f\ (n, a) = (n, f\ a)$$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$\mathsf{F}\ A = \mathbb{N} \times A$

$\mathsf{F} f\ (n, a) = (n, f\ a)$
$\mathsf{Fset}\ (n, a) = \{a\}$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$\mathsf{F}\ A = \mathbb{N} \times A$

$$\mathsf{F} f\ (n, a) = (n, f\ a)$$
$$\mathsf{Fset}\ (n, a) = \{a\}$$

$\mathsf{F}\ A = \mathbb{N} + A$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$\mathsf{F}\ A = \mathbb{N} \times A$

$\mathsf{F} f\ (n, a) = (n, f\ a)$
$\mathsf{Fset}\ (n, a) = \{a\}$

$\mathsf{F}\ A = \mathbb{N} + A$

$\mathsf{F} f\ (\mathsf{Left}\ n) = \mathsf{Left}\ n \qquad \mathsf{F} f\ (\mathsf{Right}\ a) = \mathsf{Right}\ (f\ a)$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$\mathsf{F}\ A = \mathbb{N} \times A$

$\mathsf{F} f\ (n, a) = (n, f\ a)$

$\mathsf{Fset}\ (n, a) = \{a\}$

$\mathsf{F}\ A = \mathbb{N} + A$

$\mathsf{F} f\ (\mathsf{Left}\ n) = \mathsf{Left}\ n \qquad \mathsf{F} f\ (\mathsf{Right}\ a) = \mathsf{Right}\ (f\ a)$

$\mathsf{Fset}\ (\mathsf{Left}\ n) = \varnothing \qquad \mathsf{Fset}\ (\mathsf{Right}\ a) = \{a\}$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$\mathsf{F}\ A = \mathbb{N} \times A$

$\mathsf{F}\, f\ (n, a) = (n, f\ a)$
$\mathsf{Fset}\ (n, a) = \{a\}$

$\mathsf{F}\ A = \mathbb{N} + A$

$\mathsf{F}\, f\ (\mathsf{Left}\, n) = \mathsf{Left}\, n \qquad \mathsf{F}\, f\ (\mathsf{Right}\ a) = \mathsf{Right}\ (f\ a)$
$\mathsf{Fset}\ (\mathsf{Left}\ n) = \varnothing \qquad \mathsf{Fset}\ (\mathsf{Right}\ a) = \{a\}$

$\mathsf{F}\ A = \mathsf{List}\ A$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$\mathsf{F}\ A = \mathbb{N} \times A$

$\mathsf{F} f\ (n, a) = (n, f\ a)$
$\mathsf{Fset}\ (n, a) = \{a\}$

$\mathsf{F}\ A = \mathbb{N} + A$

$\mathsf{F} f\ (\mathsf{Left}\ n) = \mathsf{Left}\ n \qquad \mathsf{F} f\ (\mathsf{Right}\ a) = \mathsf{Right}\ (f\ a)$
$\mathsf{Fset}\ (\mathsf{Left}\ n) = \varnothing \qquad \mathsf{Fset}\ (\mathsf{Right}\ a) = \{a\}$

$\mathsf{F}\ A = \mathsf{List}\ A$

$\mathsf{F} f\ (a_1 \cdot a_2 \cdot \ldots \cdot a_n) = f\ a_1 \cdot f\ a_2 \ldots \cdot f\ a_n$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$\mathsf{F}\ A = \mathbb{N} \times A$

$$\mathsf{F} f\ (n, a) = (n, f\ a)$$
$$\mathsf{Fset}\ (n, a) = \{a\}$$

$\mathsf{F}\ A = \mathbb{N} + A$

$$\mathsf{F} f\ (\mathsf{Left}\, n) = \mathsf{Left}\, n \qquad \mathsf{F} f\ (\mathsf{Right}\ a) = \mathsf{Right}\ (f\ a)$$
$$\mathsf{Fset}\ (\mathsf{Left}\, n) = \varnothing \qquad \mathsf{Fset}\ (\mathsf{Right}\ a) = \{a\}$$

$\mathsf{F}\ A = \mathsf{List}\ A$

$$\mathsf{F} f\ (a_1 \cdot a_2 \cdot \ldots \cdot a_n) = f\ a_1 \cdot f\ a_2 \ldots \cdot f\ a_n$$
$$\mathsf{Fset}\ (a_1 \cdot a_2 \cdot \ldots \cdot a_n) = \{a_1, a_2, \ldots, a_n\}$$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$\mathsf{F}\ A = \mathbb{N} \times A$

$\mathsf{F} f\ (n, a) = (n, f\ a)$
$\mathsf{Fset}\ (n, a) = \{a\}$

$\mathsf{F}\ A = \mathbb{N} + A$

$\mathsf{F} f\ (\mathsf{Left}\ n) = \mathsf{Left}\ n \qquad \mathsf{F} f\ (\mathsf{Right}\ a) = \mathsf{Right}\ (f\ a)$
$\mathsf{Fset}\ (\mathsf{Left}\ n) = \varnothing \qquad \mathsf{Fset}\ (\mathsf{Right}\ a) = \{a\}$

$\mathsf{F}\ A = \mathsf{List}\ A$

$\mathsf{F} f\ (a_1 \cdot a_2 \cdot \ldots \cdot a_n) = f\ a_1 \cdot f\ a_2 \ldots \cdot f\ a_n$
$\mathsf{Fset}\ (a_1 \cdot a_2 \cdot \ldots \cdot a_n) = \{a_1, a_2, \ldots, a_n\}$

$\mathsf{F}\ A = \mathsf{Stream}\ A$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$\mathsf{F}\ A = \mathbb{N} \times A$

$\mathsf{F} f\ (n, a) = (n, f\ a)$
$\mathsf{Fset}\ (n, a) = \{a\}$

$\mathsf{F}\ A = \mathbb{N} + A$

$\mathsf{F} f\ (\mathsf{Left}\ n) = \mathsf{Left}\ n \qquad \mathsf{F} f\ (\mathsf{Right}\ a) = \mathsf{Right}\ (f\ a)$
$\mathsf{Fset}\ (\mathsf{Left}\ n) = \varnothing \qquad \mathsf{Fset}\ (\mathsf{Right}\ a) = \{a\}$

$\mathsf{F}\ A = \mathsf{List}\ A$

$\mathsf{F} f\ (a_1 \cdot a_2 \cdot \ldots \cdot a_n) = f\ a_1 \cdot f\ a_2 \ldots \cdot f\ a_n$
$\mathsf{Fset}\ (a_1 \cdot a_2 \cdot \ldots \cdot a_n) = \{a_1, a_2, \ldots, a_n\}$

$\mathsf{F}\ A = \mathsf{Stream}\ A$

$\mathsf{F} f\ ((a_i)_{i \in \mathbb{N}}) = (f\ a_i)_{i \in \mathbb{N}}$

Examples of Natural Functors

$$A \xrightarrow{f} B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{F}\ f} \mathsf{F}\ B \qquad \mathsf{F}\ A \xrightarrow{\mathsf{Fset}} \mathcal{P}\ A$$

$\mathsf{F}\ A = \mathbb{N} \times A$

$\mathsf{F} f\ (n, a) = (n, f\ a)$
$\mathsf{Fset}\ (n, a) = \{a\}$

$\mathsf{F}\ A = \mathbb{N} + A$

$\mathsf{F} f\ (\mathsf{Left}\ n) = \mathsf{Left}\ n \qquad \mathsf{F} f\ (\mathsf{Right}\ a) = \mathsf{Right}\ (f\ a)$
$\mathsf{Fset}\ (\mathsf{Left}\ n) = \varnothing \qquad \mathsf{Fset}\ (\mathsf{Right}\ a) = \{a\}$

$\mathsf{F}\ A = \mathsf{List}\ A$

$\mathsf{F} f\ (a_1 \cdot a_2 \cdot \ldots \cdot a_n) = f\ a_1 \cdot f\ a_2 \ldots \cdot f\ a_n$
$\mathsf{Fset}\ (a_1 \cdot a_2 \cdot \ldots \cdot a_n) = \{a_1, a_2, \ldots, a_n\}$

$\mathsf{F}\ A = \mathsf{Stream}\ A$

$\mathsf{F} f\ ((a_i)_{i \in \mathbb{N}}) = (f\ a_i)_{i \in \mathbb{N}}$
$\mathsf{Fset}\ ((a_i)_{i \in \mathbb{N}}) = \{a_i \mid i \in \mathbb{N}\}$

Bounded Natural Functor (BNF)

"Bounded" means the existence of a cardinal k such that $|\mathsf{Fset}\ x| < k$ (for all A and $x \in \mathsf{F}\ A$).

Bounded Natural Functor (BNF)

“Bounded” means the existence of a cardinal k such that $|\mathsf{Fset}\, x| < k$ (for all A and $x \in \mathsf{F}\ A$).

There’s a fixed bound on the content storable in elements of F A (independently of the size of A).

This excludes, e.g., the powerset functor.

Datatypes = Initial Algebras of BNFs

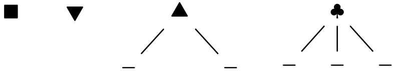
Iterating Shape Composition

Natural functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

Iterating Shape Composition

Natural functor F : Set → Set

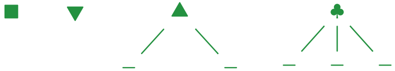
The shapes of F:



Iterating Shape Composition

Natural functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

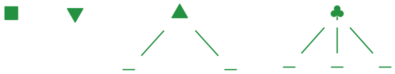
Copies of the shapes of F:



Iterating Shape Composition

Natural functor F : Set → Set

Copies of the shapes of F:

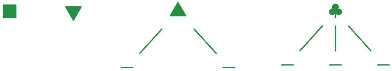


Put them together by plugging in shape for content slot

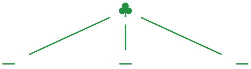
Iterating Shape Composition

Natural functor F : Set → Set

Copies of the shapes of F:



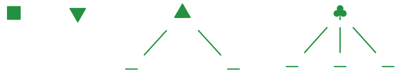
Put them together by plugging in shape for content slot



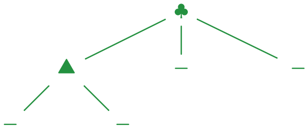
Iterating Shape Composition

Natural functor F : Set → Set

Copies of the shapes of F:



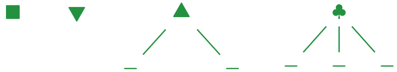
Put them together by plugging in shape for content slot



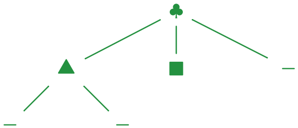
Iterating Shape Composition

Natural functor F : Set → Set

Copies of the shapes of F:



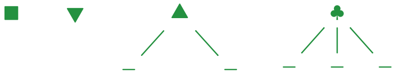
Put them together by plugging in shape for content slot



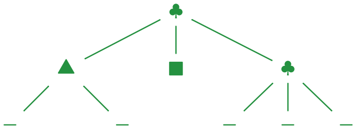
Iterating Shape Composition

Natural functor F : Set → Set

Copies of the shapes of F:



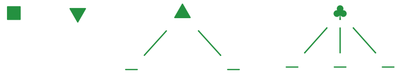
Put them together by plugging in shape for content slot



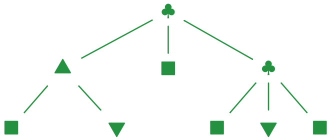
Iterating Shape Composition

Natural functor F : Set → Set

Copies of the shapes of F:



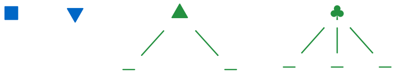
Put them together by plugging in shape for content slot until there are no lingering slots left!



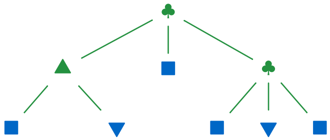
Iterating Shape Composition

Natural functor F : Set → Set

Copies of the shapes of F:



Put them together by plugging in shape for content slot until there are no lingering slots left!

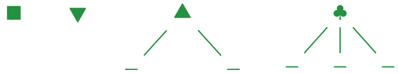


The leaves are always empty-content shapes

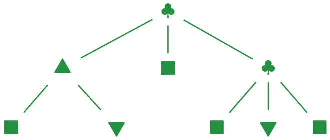
Iterating Shape Composition

Natural functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

Copies of the shapes of F:

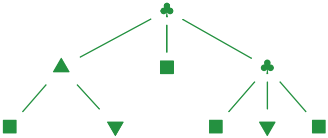


Put them together by plugging in shape for content slot until there are no lingering slots left!



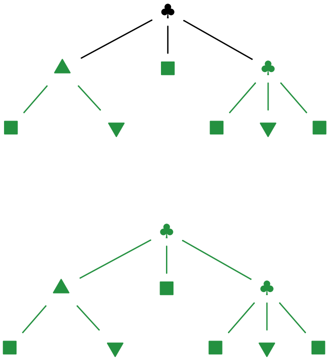
Define I_F = the set of all such finitary couplings

Properties of I_F: Bijectivity



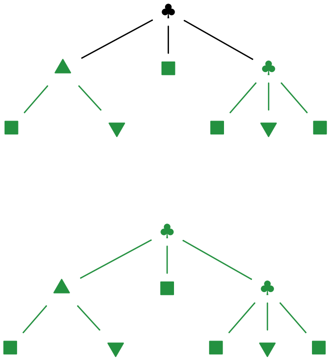
$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_\mathsf{F} \\
\mathsf{I}_\mathsf{F} \arrow[u, "\mathsf{dtor}"]
\end{tikzcd}$$

Properties of I_F: Bijectivity



$$\begin{tikzcd} F\ I_F \\ I_F \arrow[u, "\mathrm{dtor}"'] \end{tikzcd}$$

Properties of $\mathsf{I_F}$: Bijectivity



$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[d, shift left, "\text{ctor}"] \\
\mathsf{I_F} \arrow[u, shift left, "\text{dtor}"]
\end{tikzcd}$$

ctor and dtor are mutually inverse bijections

Properties of I_{F}: Iteration

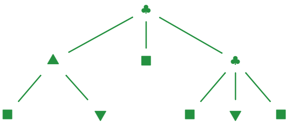
$$\begin{tikzcd}
 & \mathsf{F}\,A \arrow[d, "s"'] \\
\mathsf{I}_{\mathsf{F}} & A
\end{tikzcd}$$

Properties of I_F: Iteration

$$\begin{tikzcd}
 & \mathsf{F}\;A \arrow[d, "s"'] \\
\mathsf{I}_\mathsf{F} \arrow[r, dotted, "f"] & A
\end{tikzcd}$$

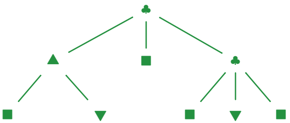
Properties of I_{F}: Iteration

$$\begin{tikzcd}
 & \mathsf{F}\,A \arrow[d, "s"'] \\
\mathsf{I}_{\mathsf{F}} \arrow[r, dotted, "f"] & A
\end{tikzcd}$$

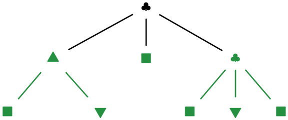


Properties of $\mathsf{I_F}$: Iteration

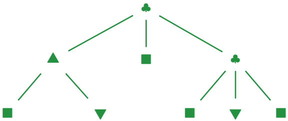
$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I_F} \arrow[u, "\mathsf{dtor}"'] \arrow[r, dotted, "f"] & A
\end{tikzcd}$$



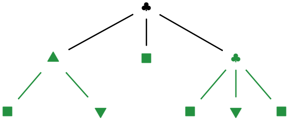
Properties of I_{F}: Iteration



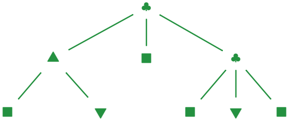
$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_{\mathsf{F}} & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I}_{\mathsf{F}} \arrow[u, "\mathsf{dtor}"'] \arrow[r, dotted, "f"] & A
\end{tikzcd}$$



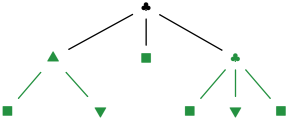
Properties of I_{F}: Iteration



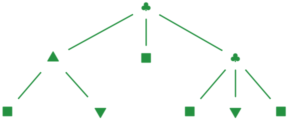
$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_{\mathsf{F}} \arrow[r, dotted, "\mathsf{F}\ f"'] & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I}_{\mathsf{F}} \arrow[u, "\mathsf{dtor}"'] \arrow[r, dotted, "f"] & A
\end{tikzcd}$$

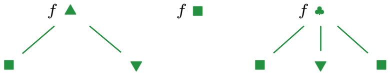


Properties of $\mathsf{I_F}$: Iteration

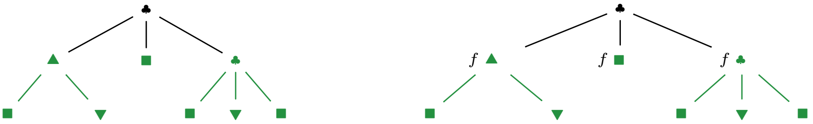


$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, dotted, "\mathsf{F}\ f"'] & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I_F} \arrow[u, "\mathsf{dtor}"'] \arrow[r, dotted, "f"] & A
\end{tikzcd}$$

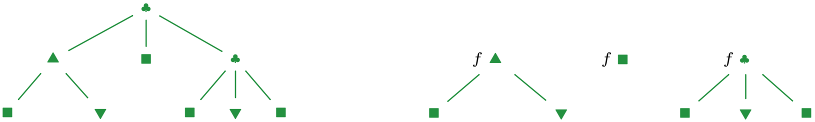




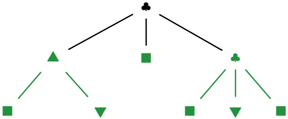
Properties of I_{F}: Iteration

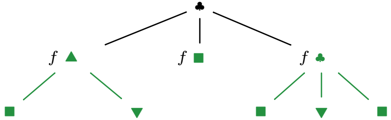


$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_{\mathsf{F}} \arrow[r, dotted, "\mathsf{F}\ f"'] & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I}_{\mathsf{F}} \arrow[u, "\mathsf{dtor}"'] \arrow[r, dotted, "f"] & A
\end{tikzcd}$$

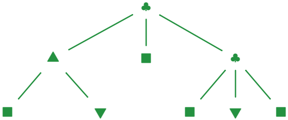


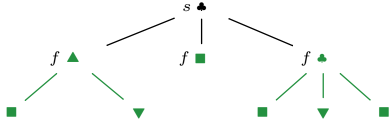
Properties of I_F: Iteration





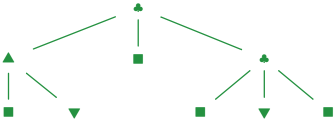
$$\begin{tikzcd}
\mathsf{F}\;\mathsf{I}_\mathsf{F} \arrow[r, dotted, "\mathsf{F}\;f"'] & \mathsf{F}\;A \arrow[d, "s"'] \\
\mathsf{I}_\mathsf{F} \arrow[u, "\mathsf{dtor}"'] \arrow[r, dotted, "f"] & A
\end{tikzcd}$$





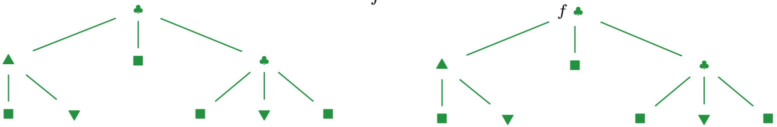
Properties of $\mathsf{I_F}$: Iteration

$$\begin{tikzcd}
 & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I_F} \arrow[r, dotted, "f"'] & A
\end{tikzcd}$$



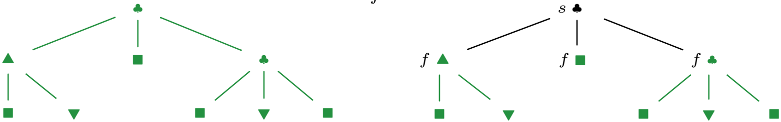
Properties of I_F: Iteration

$$\begin{tikzcd}
 & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I}_\mathsf{F} \arrow[r, dotted, "f"'] & A
\end{tikzcd}$$



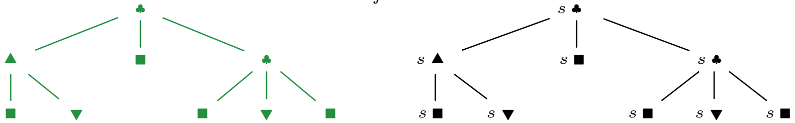
Properties of $\mathsf{I_F}$: Iteration

$$\begin{tikzcd}
& \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I_F} \arrow[r, dotted, "f"'] & A
\end{tikzcd}$$



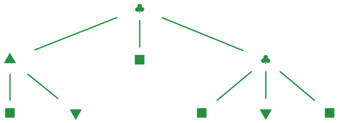
Properties of $\mathsf{I_F}$: Iteration

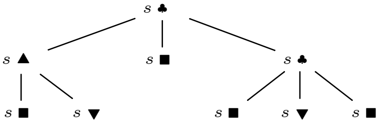
$$\begin{tikzcd}
 & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I_F} \arrow[r, dotted, "f"'] & A
\end{tikzcd}$$



Properties of I_{F}: Iteration

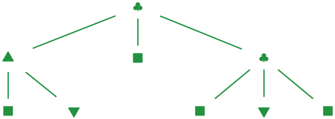
$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_{\mathsf{F}} \arrow[r, dotted, "\mathsf{F}\ f"] & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I}_{\mathsf{F}} \arrow[u, "\mathsf{dtor}"] \arrow[r, dotted, "f"'] & A
\end{tikzcd}$$

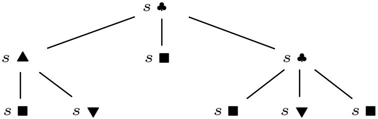




Properties of I_{F}: Iteration

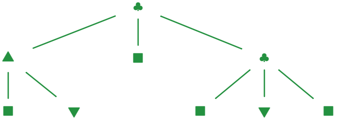
$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_{\mathsf{F}} \arrow[r, dotted, "\mathsf{F}\ f"] \arrow[d, "\mathsf{ctor}"'] & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I}_{\mathsf{F}} \arrow[r, dotted, "f"'] & A
\end{tikzcd}$$

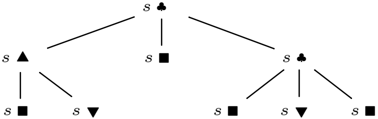




Properties of $\mathsf{I_F}$: Iteration

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, dotted, "\mathsf{F}\ f"] \arrow[d, "\mathsf{ctor}"'] & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I_F} \arrow[r, dotted, "f"'] & A
\end{tikzcd}$$

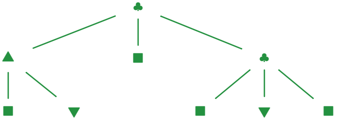


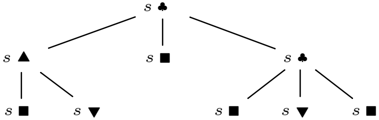


$\mathsf{I_F}$ is the initial F-algebra

Properties of I_{F}: Iteration

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_{\mathsf{F}} \arrow[r, dotted, "\mathsf{F}\ f"] \arrow[d, "\mathsf{ctor}"'] & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I}_{\mathsf{F}} \arrow[r, dotted, "f"'] & A
\end{tikzcd}$$





I_{F} is the initial F-algebra

f = iter_s

Properties of I_F: Induction

I_F

Properties of $\mathsf{I_F}$: Induction

$\mathsf{I_F}$

φ unary predicate on $\mathsf{I_F}$

Properties of I_F: Induction

I_F

φ unary predicate on I_F
Want: If
then $\forall i \in \mathsf{I}_\mathsf{F}.\ \varphi\, i$

Properties of $\mathsf{I_F}$: Induction

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \\
\mathsf{I_F} \arrow[u, "\mathsf{dtor}"]
\end{tikzcd}$$

φ unary predicate on $\mathsf{I_F}$

Want: If

then $\forall i \in \mathsf{I_F}.\ \varphi\ i$

Properties of $\mathsf{I_F}$: Induction

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{Fset}"] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} \arrow[u, "\mathsf{dtor}"] &
\end{tikzcd}$$

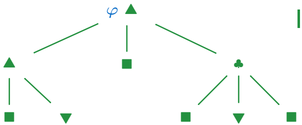
φ unary predicate on $\mathsf{I_F}$

Want: If

then $\forall i \in \mathsf{I_F}.\ \varphi\ i$

Properties of I_F: Induction

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_\mathsf{F} \arrow[r, "\mathsf{Fset}"] & \mathcal{P}\ \mathsf{I}_\mathsf{F} \\
\mathsf{I}_\mathsf{F} \arrow[u, "\mathsf{dtor}"] &
\end{tikzcd}$$

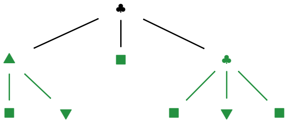


φ unary predicate on I_F

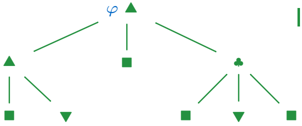
Want: If $\forall i \in \mathsf{I}_\mathsf{F}.$ $\qquad\qquad\qquad\qquad \Longrightarrow \varphi\ i$

then $\forall i \in \mathsf{I}_\mathsf{F}.\ \varphi\ i$

Properties of $\mathsf{I_F}$: Induction



$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{Fset}"] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} \arrow[u, "\mathsf{dtor}"] &
\end{tikzcd}$$

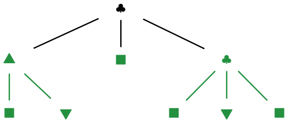


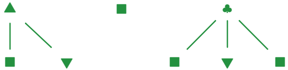
φ unary predicate on $\mathsf{I_F}$

Want: If $\forall i \in \mathsf{I_F}$. $\Longrightarrow \varphi\ i$

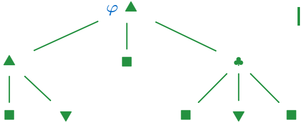
then $\forall i \in \mathsf{I_F}.\ \varphi\ i$

Properties of I_F: Induction





$$\begin{tikzcd} F\ I_F \arrow[r, "\mathsf{Fset}"] & \mathcal{P}\ I_F \\ I_F \arrow[u, "\mathsf{dtor}"] & \end{tikzcd}$$

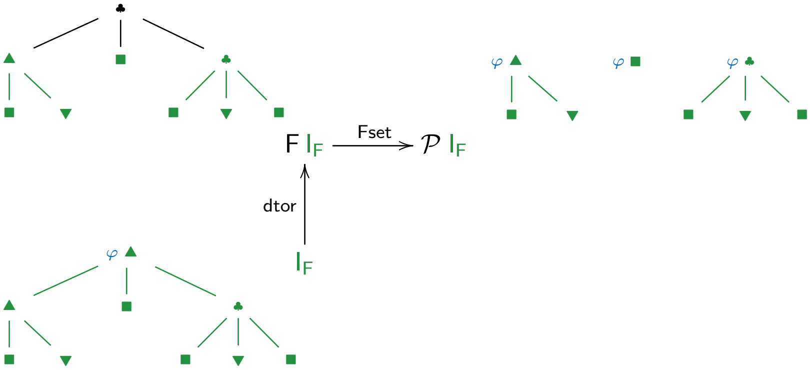


φ unary predicate on I_F

Want: If $\forall i \in I_F.$ $\Longrightarrow \varphi\ i$

then $\forall i \in I_F.\ \varphi\ i$

Properties of $\mathsf{I_F}$: Induction



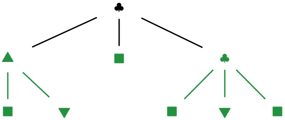
$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{Fset}"] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} \arrow[u, "\mathsf{dtor}"] &
\end{tikzcd}$$

φ unary predicate on $\mathsf{I_F}$

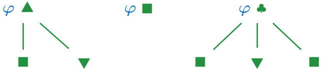
If $\forall i \in \mathsf{I_F}.\ (\forall i' \in \mathsf{Fset}\ (\mathsf{dtor}\ i).\ \varphi\ i') \Longrightarrow \varphi\ i$

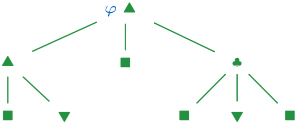
then $\forall i \in \mathsf{I_F}.\ \varphi\ i$

Properties of $\mathsf{I_F}$: Induction



$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{Fset}"] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} \arrow[u, "\mathsf{dtor}"] &
\end{tikzcd}$$



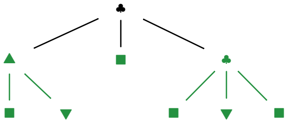


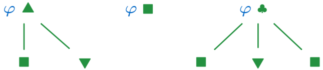
φ unary predicate on $\mathsf{I_F}$

If $\forall i \in \mathsf{I_F}.\ (\forall i' \in \mathsf{Fset}\ (\mathsf{dtor}\ i).\ \varphi\ i') \Longrightarrow \varphi\ i$

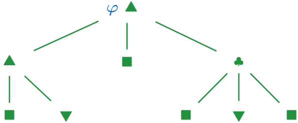
then $\forall i \in \mathsf{I_F}.\ \varphi\ i$

Properties of $\mathsf{I_F}$: Induction





$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{Fset}"] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} \arrow[u, "\mathsf{dtor}"] \arrow[ur, "\mathsf{components}"'] &
\end{tikzcd}$$

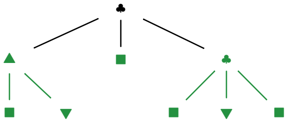


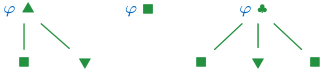
φ unary predicate on $\mathsf{I_F}$

If $\forall i \in \mathsf{I_F}.\ (\forall i' \in \mathsf{components}\ i.\ \varphi\ i') \Longrightarrow \varphi\ i$

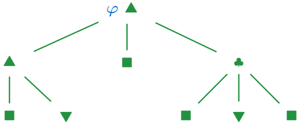
then $\forall i \in \mathsf{I_F}.\ \varphi\ i$

Properties of $\mathsf{I_F}$: Destructor-Style Induction





$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{Fset}"] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} \arrow[u, "\mathsf{dtor}"] \arrow[ur, "\mathsf{components}"'] &
\end{tikzcd}$$

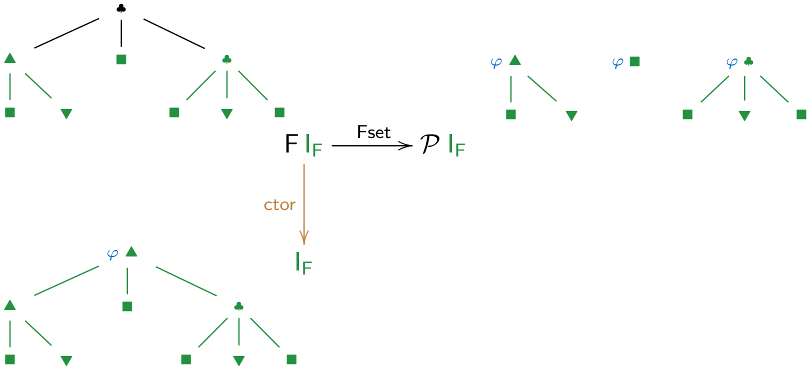


φ unary predicate on $\mathsf{I_F}$

If $\forall i \in \mathsf{I_F}.\ (\forall i' \in \mathsf{components}\ i.\ \varphi\ i') \Longrightarrow \varphi\ i$

then $\forall i \in \mathsf{I_F}.\ \varphi\ i$

Properties of $\mathsf{I_F}$: Constructor-Style Induction



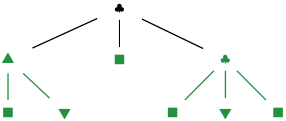
$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{Fset}"] \arrow[d, "\mathsf{ctor}"'] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} &
\end{tikzcd}$$

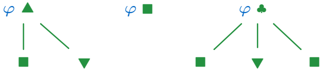
φ unary predicate on $\mathsf{I_F}$

If $\forall i \in \mathsf{I_F}.\ (\forall i' \in \mathsf{components}\ i.\ \varphi\ i') \Longrightarrow \varphi\ i$

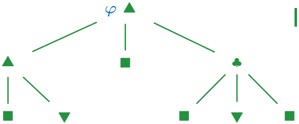
then $\forall i \in \mathsf{I_F}.\ \varphi\ i$

Properties of I_F: Constructor-Style Induction





$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_\mathsf{F} \arrow[r, "\mathsf{Fset}"] \arrow[d, "\mathsf{ctor}"'] & \mathcal{P}\ \mathsf{I}_\mathsf{F} \\
\mathsf{I}_\mathsf{F} &
\end{tikzcd}$$



φ unary predicate on I_F

If $\forall x \in \mathsf{F}\ \mathsf{I}_\mathsf{F}.\ (\forall i \in \mathsf{Fset}\ x.\ \varphi\ i) \Longrightarrow \varphi\ (\mathsf{ctor}\ x)$

then $\forall i \in \mathsf{I}_\mathsf{F}.\ \varphi\ i$

Bottom line for $\mathsf{I_F}$

Given a natural functor F, $\quad (\mathsf{I_F},\ \mathsf{ctor} : \mathsf{F}\,\mathsf{I_F} \to \mathsf{I_F})$ satisfies:

Bottom line for I_F

Given a natural functor F, $(\mathsf{I}_\mathsf{F},\ \mathsf{ctor} : \mathsf{F}\ \mathsf{I}_\mathsf{F} \to \mathsf{I}_\mathsf{F})$ satisfies:

ctor bijection

Bottom line for I_F

Given a natural functor F, $(\mathsf{I}_\mathsf{F},\ \mathsf{ctor} : \mathsf{F}\ \mathsf{I}_\mathsf{F} \to \mathsf{I}_\mathsf{F})$ satisfies:

ctor bijection

Iteration (Initial Algebra Property): For all $(A, s : \mathsf{F}\ A \to A)$, there exists a unique function iter_s such that

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_\mathsf{F} \arrow[r, "\mathsf{F}\ \mathsf{iter}_s"] \arrow[d, "\mathsf{ctor}"'] & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I}_\mathsf{F} \arrow[r, "\mathsf{iter}_s"'] & A
\end{tikzcd}$$

Bottom line for I_F

Given a natural functor F, $(\mathsf{I}_\mathsf{F},\ \mathsf{ctor} : \mathsf{F}\ \mathsf{I}_\mathsf{F} \to \mathsf{I}_\mathsf{F})$ satisfies:

ctor bijection

Iteration (Initial Algebra Property): For all $(A, s : \mathsf{F}\ A \to A)$, there exists a unique function iter_s such that

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_\mathsf{F} \arrow[r, "\mathsf{F}\ \mathsf{iter}_s"] \arrow[d, "\mathsf{ctor}"'] & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I}_\mathsf{F} \arrow[r, "\mathsf{iter}_s"'] & A
\end{tikzcd}$$

Induction: Given any predicate φ on I_F

$$\frac{\forall x \in \mathsf{F}\ \mathsf{I}_\mathsf{F}.\ (\forall i \in \mathsf{Fset}\ x.\ \varphi\ i) \Longrightarrow \varphi\ (\mathsf{ctor}\ x)}{\forall i \in \mathsf{I}_\mathsf{F}.\ \varphi\ i}$$

Bottom line for $\mathsf{I_F}$

Given a natural functor F, $\quad(\mathsf{I_F},\ \mathsf{ctor} : \mathsf{F}\ \mathsf{I_F} \to \mathsf{I_F})$ satisfies:

ctor bijection $\qquad \boxed{\mathsf{I_F} = \text{the datatype of } \mathsf{F}}$

Iteration (Initial Algebra Property): For all $(A, s : \mathsf{F}\ A \to A)$, there exists a unique function iter_s such that

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{F}\ \mathsf{iter}_s"] \arrow[d, "\mathsf{ctor}"'] & \mathsf{F}\ A \arrow[d, "s"'] \\
\mathsf{I_F} \arrow[r, "\mathsf{iter}_s"'] & A
\end{tikzcd}$$

Induction: Given any predicate φ on $\mathsf{I_F}$

$$\frac{\forall x \in \mathsf{F}\ \mathsf{I_F}.\ (\forall i \in \mathsf{Fset}\ x.\ \varphi\ i) \Longrightarrow \varphi\ (\mathsf{ctor}\ x)}{\forall i \in \mathsf{I_F}.\ \varphi\ i}$$

Example of Datatype

Let B be a fixed set. $\mathsf{F}\ A = \{*\} + B \times A$

Example of Datatype

Let B be a fixed set. $\mathsf{F}\ A = \{*\} + B \times A$

The shapes of F: Left $*$

Example of Datatype

Let B be a fixed set. F A = $\{*\} + B \times A$

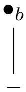
The shapes of F: Left $*$ Right $(b, \_)$ for each $b \in B$

Example of Datatype

Let B be a fixed set. $\quad \mathsf{F}\ A = \{*\} + B \times A$

The shapes of F: Left $*$ Right $(b, \_)$ for each $b \in B$

Or, graphically: $\blacksquare_*$

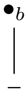


for each $b \in B$

Example of Datatype

Let B be a fixed set. $\mathsf{F}\,A = \{*\} + B \times A$

The shapes of F: Left $*$ Right $(b, \_)$ for each $b \in B$

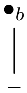
Or, graphically: ■$_*$  for each $b \in B$

Who is I_F?

Example of Datatype

Let B be a fixed set. $\mathsf{F}\ A = \{*\} + B \times A$

The shapes of F: Left $*$ Right $(b, \_)$ for each $b \in B$

Or, graphically: $\blacksquare_*$    for each $b \in B$

Who is I_F?
Its elements have the form $\mathsf{Right}(b_1, \ldots, \mathsf{Right}(b_n, \mathsf{Right}\ (\mathsf{Left}\ *)) \ldots)$

Example of Datatype

Let B be a fixed set. $\mathsf{F}\ A = \{*\} + B \times A$

The shapes of F: Left $*$ Right $(b, \_)$ for each $b \in B$

Or, graphically:

$\blacksquare_*$
$\bullet_b$
$\_$

for each $b \in B$

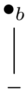
Who is I_{F}?
Its elements have the form $\mathsf{Right}(b_1, \ldots, \mathsf{Right}(b_n, \mathsf{Right}\ (\mathsf{Left}\ *)) \ldots)$
I.e., essentially lists $b_1 \cdot \ldots \cdot b_n$

Example of Datatype

Let B be a fixed set. $\quad \mathsf{F}\ A = \{*\} + B \times A$

The shapes of F: Left $*$ $\quad$ Right $(b, \_)$ for each $b \in B$

Or, graphically: $\blacksquare_*$



for each $b \in B$

Who is $\mathsf{I_F}$?
Its elements have the form $\mathsf{Right}(b_1, \ldots, \mathsf{Right}(b_n, \mathsf{Right}\ (\mathsf{Left}\ *)) \ldots)$
I.e., essentially lists $b_1 \cdot \ldots \cdot b_n$
So $\mathsf{I_F} = \mathsf{List}_B$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad f = \mathsf{iter}_s \quad \mathsf{I_F} = \mathsf{List}_B$

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{F}\ f"] \arrow[d, "\mathsf{ctor}"'] & \mathsf{F}\ A \arrow[d, "s"] \\
\mathsf{I_F} \arrow[r, "f"'] & A
\end{tikzcd}$$

$\forall x \in \mathsf{F}\ \mathsf{I_F}.\ f\ (\mathsf{ctor}\ x) = s\ ((\mathsf{F}\ f)\ x)$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad f = \mathsf{iter}_s \quad \mathsf{I_F} = \mathsf{List}_B$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I_F} \arrow[r, "\{*\}+B\times f"] \arrow[d, "\mathsf{ctor}"'] & \{*\} + B \times A \arrow[d, "s"] \\
\mathsf{I_F} \arrow[r, "f"'] & A
\end{tikzcd}$$

$\forall x \in \mathsf{F}\ \mathsf{I_F}.\ f\ (\mathsf{ctor}\ x) = s\ ((\mathsf{F}\ f)\ x)$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad f = \mathsf{iter}_s \quad \mathsf{I_F} = \mathsf{List}_B$

Define: $\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *) \quad \mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$
$\mathsf{Nil}^A = s\ (\mathsf{Left}\ *) \quad \mathsf{Cons}^A(b, a) = s\ (\mathsf{Right}\ (b, a))$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I_F} \arrow[r, "\{*\}+B\times f"] \arrow[d, "\mathsf{ctor}"'] & \{*\} + B \times A \arrow[d, "s"] \\
\mathsf{I_F} \arrow[r, "f"'] & A
\end{tikzcd}$$

$\forall x \in \mathsf{F}\ \mathsf{I_F}.\ f\ (\mathsf{ctor}\ x) = s\ ((\mathsf{F}\ f)\ x)$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad f = \mathsf{iter}_s \quad \mathsf{I_F} = \mathsf{List}_B$

Define: $\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *) \quad \mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$
$\mathsf{Nil}^A = s\ (\mathsf{Left}\ *) \quad \mathsf{Cons}^A(b, a) = s\ (\mathsf{Right}\ (b, a))$

$$\begin{tikzcd}
B \times \mathsf{I_F} \arrow[r, "B \times f"] \arrow[d, "\mathsf{Cons}"'] & B \times A \arrow[d, "\mathsf{Cons}^A"] \\
\mathsf{Nil} \in \mathsf{I_F} \arrow[r, "f"'] & A \ni \mathsf{Nil}^A
\end{tikzcd}$$

$\forall x \in \mathsf{F}\ \mathsf{I_F}.\ f\ (\mathsf{ctor}\ x) = s\ ((\mathsf{F}\ f)\ x)$

Example of Datatype: List

B fixed $\mathsf{F}\ A = \{*\} + B \times A$ $f = \mathsf{iter}_s$ $\mathsf{I}_\mathsf{F} = \mathsf{List}_B$

Define: $\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *)$ $\mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$
$\mathsf{Nil}^A = s\ (\mathsf{Left}\ *)$ $\mathsf{Cons}^A(b, a) = s\ (\mathsf{Right}\ (b, a))$

$$\begin{tikzcd}
B \times \mathsf{I}_\mathsf{F} \arrow[r, "B \times f"] \arrow[d, "\mathsf{Cons}"'] & B \times A \arrow[d, "\mathsf{Cons}^A"] \\
\mathsf{Nil} \in \mathsf{I}_\mathsf{F} \arrow[r, "f"'] & A \ni \mathsf{Nil}^A
\end{tikzcd}$$

$f\ \mathsf{Nil} = \mathsf{Nil}^A$
$\forall b \in B,\ i \in \mathsf{I}_\mathsf{F}.\ f\ (\mathsf{Cons}\ (b, i)) = \mathsf{Cons}^A\ (b, f\ i)$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad f = \mathsf{iter}_s \quad \mathsf{I_F} = \mathsf{List}_B$

Define: $\quad \mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *) \quad \mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$

$\qquad\qquad \mathsf{Nil}^A = s\ (\mathsf{Left}\ *) \quad \mathsf{Cons}^A(b, a) = s\ (\mathsf{Right}\ (b, a))$

$$\begin{tikzcd}
B \times \mathsf{I_F} \arrow[r, "B\times f"] \arrow[d, "\mathsf{Cons}"'] & B \times A \arrow[d, "\mathsf{Cons}^A"] \\
\mathsf{Nil} \in \mathsf{I_F} \arrow[r, "f"'] & A \ni \mathsf{Nil}^A
\end{tikzcd}$$

$f\ \mathsf{Nil} = \mathsf{Nil}^A$

We obtain standard list iteration!

$\forall b \in B,\ i \in \mathsf{I_F}.\ f\ (\mathsf{Cons}\ (b, i)) = \mathsf{Cons}^A\ (b, f\ i)$

Example of Datatype: List

B fixed $\quad \mathsf{F}\,A = \{*\} + B \times A \quad \mathsf{I_F} = \mathsf{List}_B$

$$\begin{tikzcd}
\mathsf{F}\,\mathsf{I_F} \arrow[r, "\mathsf{Fset}"] \arrow[d, "\mathsf{ctor}"'] & \mathcal{P}\,\mathsf{I_F} \\
\mathsf{I_F} &
\end{tikzcd}$$

$$\frac{\forall x \in \mathsf{F}\,\mathsf{I_F}.\ (\forall i \in \mathsf{Fset}\ x.\ \varphi\ i) \Longrightarrow \varphi\ (\mathsf{ctor}\ x)}{\forall i \in \mathsf{I_F}.\ \varphi\ i}$$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad \mathsf{I_F} = \mathsf{List}_B$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I_F} \arrow[r, "\mathsf{Left}\ * \mapsto \varnothing,\ \mathsf{Right}\ (b,i) \mapsto \{i\}"] \arrow[d, "\mathsf{ctor}"'] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} &
\end{tikzcd}$$

$$\frac{\forall x \in \mathsf{F}\ \mathsf{I_F}.\ (\forall i \in \mathsf{Fset}\ x.\ \varphi\ i) \Longrightarrow \varphi\ (\mathsf{ctor}\ x)}{\forall i \in \mathsf{I_F}.\ \varphi\ i}$$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad \mathsf{I_F} = \mathsf{List}_B$

$\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *) \quad \mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I_F} \arrow[r, "\mathsf{Left}\ * \mapsto \varnothing,\ \mathsf{Right}\ (b{,}i) \mapsto \{i\}"] \arrow[d, "\mathsf{ctor}"'] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} &
\end{tikzcd}$$

$$\frac{\forall x \in \mathsf{F}\ \mathsf{I_F}.\ (\forall i \in \mathsf{Fset}\ x.\ \varphi\ i) \Longrightarrow \varphi\ (\mathsf{ctor}\ x)}{\forall i \in \mathsf{I_F}.\ \varphi\ i}$$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad \mathsf{I_F} = \mathsf{List}_B$

$\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *) \quad \mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I_F} \arrow[rr, "\mathsf{Left}\ * \mapsto \varnothing,\ \mathsf{Right}\ (b{,}i) \mapsto \{i\}"] \arrow[d, "\mathsf{ctor}"'] & & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} & &
\end{tikzcd}$$

$$\frac{\begin{array}{l}(\forall i \in \mathsf{Fset}\ (\mathsf{Left}\ *).\ \varphi\ i) \Longrightarrow \varphi\ (\mathsf{ctor}\ (\mathsf{Left}\ *)) \\ \forall b \in B,\ i \in \mathsf{I_F}.\ (\forall i' \in \mathsf{Fset}\ (\mathsf{Right}\ (b, i)).\ \varphi\ i') \Longrightarrow \varphi\ (\mathsf{ctor}\ (\mathsf{Right}\ (b, i)))\end{array}}{\forall i \in \mathsf{I_F}.\ \varphi\ i}$$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad \mathsf{I_F} = \mathsf{List}_B$

$\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *) \quad \mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I_F} \arrow[r, "\mathsf{Left}\ * \mapsto \varnothing,\ \mathsf{Right}\ (b,i) \mapsto \{i\}"] \arrow[d, "\mathsf{ctor}"'] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} &
\end{tikzcd}$$

$$\frac{\begin{array}{l}(\forall i \in \varnothing.\ \varphi\ i) \Longrightarrow \varphi\ (\mathsf{ctor}\ (\mathsf{Left}\ *)) \\ \forall b \in B,\ i \in \mathsf{I_F}.\ (\forall i' \in \mathsf{Fset}\ (\mathsf{Right}\ (b, i)).\ \varphi\ i') \Longrightarrow \varphi\ (\mathsf{ctor}\ (\mathsf{Right}\ (b, i)))\end{array}}{\forall i \in \mathsf{I_F}.\ \varphi\ i}$$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad \mathsf{I_F} = \mathsf{List}_B$

$\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *) \quad \mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I_F} \arrow[rr, "\mathsf{Left}\ * \mapsto \varnothing,\ \mathsf{Right}\ (b,i) \mapsto \{i\}"] \arrow[d, "\mathsf{ctor}"'] & & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} & &
\end{tikzcd}$$

$$\frac{\begin{array}{l} \varphi\ (\mathsf{ctor}\ (\mathsf{Left}\ *)) \\ \forall b \in B,\ i \in \mathsf{I_F}.\ (\forall i' \in \mathsf{Fset}\ (\mathsf{Right}\ (b, i)).\ \varphi\ i') \Longrightarrow \varphi\ (\mathsf{ctor}\ (\mathsf{Right}\ (b, i))) \end{array}}{\forall i \in \mathsf{I_F}.\ \varphi\ i}$$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad \mathsf{I_F} = \mathsf{List}_B$

$\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *) \quad \mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I_F} \arrow[rr, "\mathsf{Left}\ * \mapsto \varnothing,\ \mathsf{Right}\ (b{,}i) \mapsto \{i\}"] \arrow[d, "\mathsf{ctor}"'] & & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} & &
\end{tikzcd}$$

$$\frac{\varphi\ \mathsf{Nil} \qquad \forall b \in B,\ i \in \mathsf{I_F}.\ (\forall i' \in \mathsf{Fset}\ (\mathsf{Right}\ (b, i)).\ \varphi\ i') \Longrightarrow \varphi\ (\mathsf{ctor}\ (\mathsf{Right}\ (b, i)))}{\forall i \in \mathsf{I_F}.\ \varphi\ i}$$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad \mathsf{I_F} = \mathsf{List}_B$

$\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *) \quad \mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I_F} \arrow[r, "\mathsf{Left}\ * \mapsto \varnothing,\ \mathsf{Right}\ (b,i) \mapsto \{i\}"] \arrow[d, "\mathsf{ctor}"'] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} &
\end{tikzcd}$$

$$\frac{\begin{array}{l}\varphi\ \mathsf{Nil} \\ \forall b \in B,\ i \in \mathsf{I_F}.\ (\forall i' \in \{i\}.\ \varphi\ i') \Longrightarrow \varphi\ (\mathsf{ctor}\ (\mathsf{Right}\ (b, i)))\end{array}}{\forall i \in \mathsf{I_F}.\ \varphi\ i}$$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad \mathsf{I_F} = \mathsf{List}_B$

$\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *) \quad \mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I_F} \arrow[rr, "\mathsf{Left}\ * \mapsto \varnothing,\ \mathsf{Right}\ (b,i) \mapsto \{i\}"] \arrow[d, "\mathsf{ctor}"'] & & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} & &
\end{tikzcd}$$

$$\frac{\begin{array}{l}\varphi\ \mathsf{Nil} \\ \forall b \in B,\ i \in \mathsf{I_F}.\ \varphi\ i \Longrightarrow \varphi\ (\mathsf{ctor}\ (\mathsf{Right}\ (b, i)))\end{array}}{\forall i \in \mathsf{I_F}.\ \varphi\ i}$$

Example of Datatype: List

B fixed $\mathsf{F}\ A = \{*\} + B \times A$ $\mathsf{I}_{\mathsf{F}} = \mathsf{List}_B$

$\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *)$ $\mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I}_{\mathsf{F}} \arrow[r, "\mathsf{Left}\ * \mapsto \varnothing,\ \ \mathsf{Right}\ (b{,}i) \mapsto \{i\}"] \arrow[d, "\mathsf{ctor}"'] & \mathcal{P}\ \mathsf{I}_{\mathsf{F}} \\
\mathsf{I}_{\mathsf{F}} &
\end{tikzcd}$$

$$\frac{\begin{array}{l}\varphi\ \mathsf{Nil} \\ \forall b \in B,\ i \in \mathsf{I}_{\mathsf{F}}.\ \varphi\ i \Longrightarrow \varphi\ (\mathsf{Cons}\ (b, i))\end{array}}{\forall i \in \mathsf{I}_{\mathsf{F}}.\ \varphi\ i}$$

Example of Datatype: List

B fixed $\quad \mathsf{F}\ A = \{*\} + B \times A \quad \mathsf{I_F} = \mathsf{List}_B$

$\mathsf{Nil} = \mathsf{ctor}\ (\mathsf{Left}\ *) \quad \mathsf{Cons}(b, i) = \mathsf{ctor}\ (\mathsf{Right}\ (b, i))$

$$\begin{tikzcd}
\{*\} + B \times \mathsf{I_F} \arrow[r, "\mathsf{Left}\ * \mapsto \varnothing,\ \mathsf{Right}\ (b,i) \mapsto \{i\}"] \arrow[d, "\mathsf{ctor}"'] & \mathcal{P}\ \mathsf{I_F} \\
\mathsf{I_F} &
\end{tikzcd}$$

Obtain standard list induction!

$$\frac{\varphi\ \mathsf{Nil} \qquad \forall b \in B,\ i \in \mathsf{I_F}.\ \varphi\ i \Longrightarrow \varphi\ (\mathsf{Cons}\ (b, i))}{\forall i \in \mathsf{I_F}.\ \varphi\ i}$$

Codatatypes $=$ Final Coalgebras of BNFs

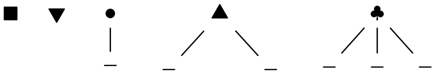
Iterating Shape Composition Revisited

Natural functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

Iterating Shape Composition Revisited

Natural functor $F : \mathsf{Set} \to \mathsf{Set}$

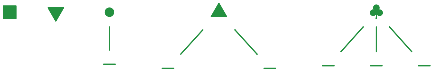
The shapes of F:



Iterating Shape Composition Revisited

Natural functor F : Set → Set

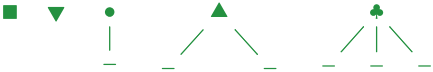
Copies of the shapes of F:



Iterating Shape Composition Revisited

Natural functor F : Set → Set

Copies of the shapes of F:

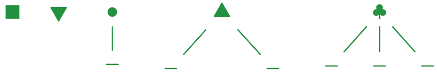


Put them together by plugging in shape for content slot

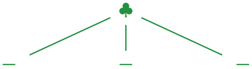
Iterating Shape Composition Revisited

Natural functor F : Set → Set

Copies of the shapes of F:



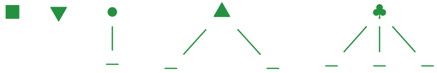
Put them together by plugging in shape for content slot



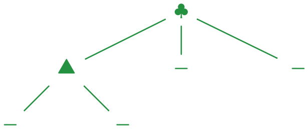
Iterating Shape Composition Revisited

Natural functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

Copies of the shapes of F:



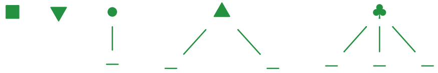
Put them together by plugging in shape for content slot



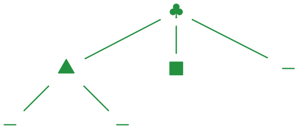
Iterating Shape Composition Revisited

Natural functor $F : \mathsf{Set} \to \mathsf{Set}$

Copies of the shapes of F:



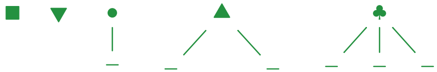
Put them together by plugging in shape for content slot



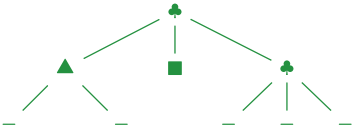
Iterating Shape Composition Revisited

Natural functor F : Set → Set

Copies of the shapes of F:



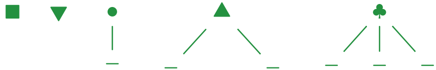
Put them together by plugging in shape for content slot



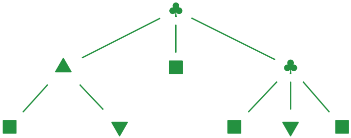
Iterating Shape Composition Revisited

Natural functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

Copies of the shapes of F:



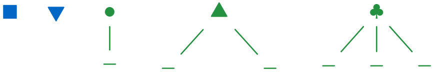
Put them together by plugging in shape for content slot
until there are no lingering slots left!



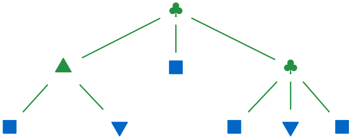
Iterating Shape Composition Revisited

Natural functor F : Set → Set

Copies of the shapes of F:



Put them together by plugging in shape for content slot until there are no lingering slots left!

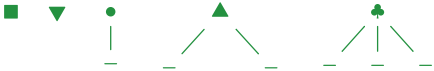


The leaves are always empty-content shapes

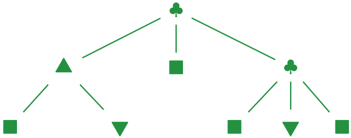
Iterating Shape Composition Revisited

Natural functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

Copies of the shapes of F:



Put them together by plugging in shape for content slot until there are no lingering slots left!

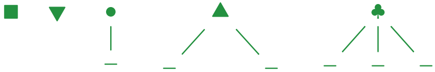


The leaves are always empty-content shapes

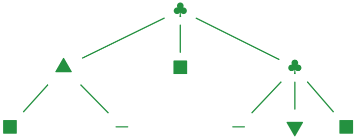
Iterating Shape Composition Revisited

Natural functor $F : \mathsf{Set} \to \mathsf{Set}$

Copies of the shapes of F:



Put them together by plugging in shape for content slot until there are no lingering slots left!

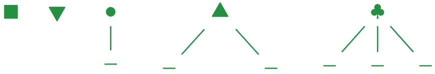


Allow infinite couplings

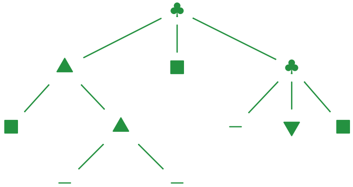
Iterating Shape Composition Revisited

Natural functor F : Set → Set

Copies of the shapes of F:



Put them together by plugging in shape for content slot until there are no lingering slots left!

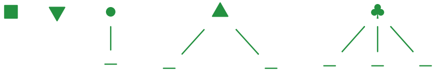


Allow infinite couplings

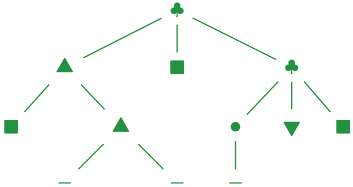
Iterating Shape Composition Revisited

Natural functor F : Set → Set

Copies of the shapes of F:



Put them together by plugging in shape for content slot
until there are no lingering slots left!

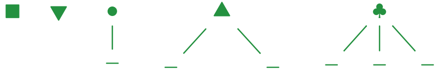


Allow infinite couplings

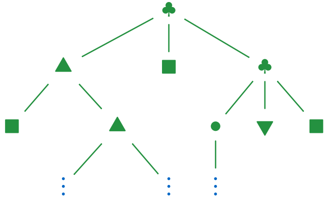
Iterating Shape Composition Revisited

Natural functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

Copies of the shapes of F:



Put them together by plugging in shape for content slot until there are no lingering slots left!

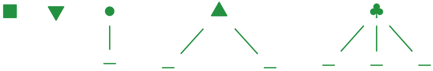


Allow infinite couplings

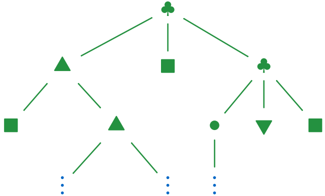
Iterating Shape Composition Revisited

Natural functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$

Copies of the shapes of F:

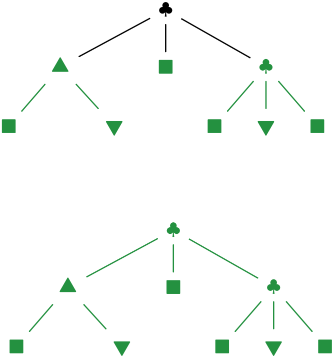


Put them together by plugging in shape for content slot
until there are no lingering slots left!



Define J_{F} = the set of all such (possibly) infinitary couplings

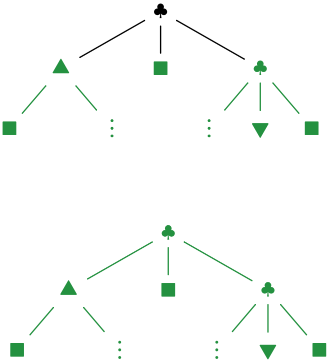
Recall: Properties of I_F: Bijectivity



$$\begin{tikzcd} F\ I_F \arrow[d, shift left, "\mathsf{ctor}"] \\ I_F \arrow[u, shift left, "\mathsf{dtor}"] \end{tikzcd}$$

ctor and dtor are mutually inverse bijections

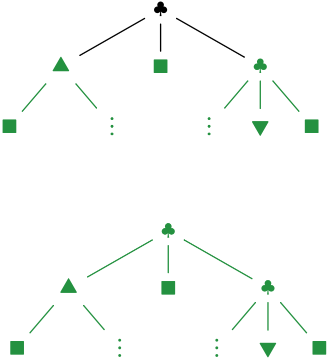
Properties of J_F: Bijectivity



$$\begin{tikzcd}
F\ J_F \arrow[d, shift left, "\mathsf{ctor}"] \\
J_F \arrow[u, shift left, "\mathsf{dtor}"]
\end{tikzcd}$$

ctor and dtor are mutually inverse bijections

Properties of J_F: Bijectivity



$$\begin{tikzcd}
F\,J_F \arrow[d, shift left, "\mathrm{ctor}"] \\
J_F \arrow[u, shift left, "\mathrm{dtor}"]
\end{tikzcd}$$

ctor and dtor are mutually inverse bijections

A similar property holds for J_F, where we use the same notations for constructor and destructor

$\mathsf{I_F}$ is embedded in $\mathsf{J_F}$

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{F}\ \iota"] \arrow[d, shift left, "\mathsf{ctor}"] & \mathsf{F}\ \mathsf{J_F} \arrow[d, shift left, "\mathsf{ctor}"] \\
\mathsf{I_F} \arrow[u, shift left, "\mathsf{dtor}"] \arrow[r, "\iota"] & \mathsf{J_F} \arrow[u, shift left, "\mathsf{dtor}"]
\end{tikzcd}$$

$\mathsf{I_F}$ is embedded in $\mathsf{J_F}$

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{F}\ \iota"] \arrow[d, shift left, "\mathsf{ctor}"] & \mathsf{F}\ \mathsf{J_F} \arrow[d, shift left, "\mathsf{ctor}"] \\
\mathsf{I_F} \arrow[r, "\iota"] \arrow[u, shift left, "\mathsf{dtor}"] & \mathsf{J_F} \arrow[u, shift left, "\mathsf{dtor}"]
\end{tikzcd}$$

$\iota = \mathsf{iter}_{\mathsf{ctor}:\mathsf{F}\ \mathsf{J_F}\to\mathsf{F}\ \mathsf{J_F}}$

Properties of J_{F}: Coiteration

$$\begin{tikzcd}
\mathsf{F}\ A & \\
A \arrow[u, "s"'] & \mathsf{J}_{\mathsf{F}}
\end{tikzcd}$$

Properties of J_{F}: Coiteration

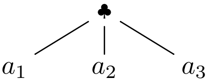
$$\begin{tikzcd}
\mathsf{F}\ A & \\
A \arrow[u, "s"'] \arrow[r, dotted, "f"] & \mathsf{J}_{\mathsf{F}}
\end{tikzcd}$$

Properties of J_{F}: Coiteration

$$\begin{tikzcd}
\mathsf{F}\ A & \\
A \arrow[u, "s"'] \arrow[r, dotted, "f"] & \mathsf{J}_{\mathsf{F}}
\end{tikzcd}$$

a

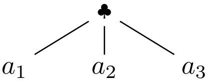
Properties of $\mathsf{J_F}$: Coiteration



$$\begin{tikzcd}
\mathsf{F}\ A & \\
A \arrow[u, "s"'] \arrow[r, dotted, "f"] & \mathsf{J_F}
\end{tikzcd}$$

a

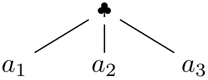
Properties of $\mathsf{J_F}$: Coiteration



$$\begin{tikzcd}
\mathsf{F}\ A & \mathsf{F}\ \mathsf{J_F} \arrow[d, "\mathsf{ctor}"'] \\
A \arrow[u, "s"'] \arrow[r, dotted, "f"] & \mathsf{J_F}
\end{tikzcd}$$

a

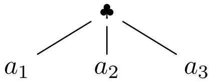
Properties of J_{F}: Coiteration

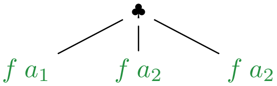


$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, dotted, "\mathsf{F}\ f"'] & \mathsf{F}\ \mathsf{J}_{\mathsf{F}} \arrow[d, "\mathsf{ctor}"'] \\
A \arrow[u, "s"'] \arrow[r, dotted, "f"] & \mathsf{J}_{\mathsf{F}}
\end{tikzcd}$$

a

Properties of $\mathsf{J_F}$: Coiteration

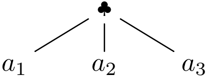


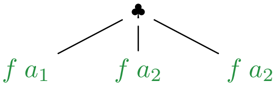


$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, dotted, "\mathsf{F}\ f"'] & \mathsf{F}\ \mathsf{J_F} \arrow[d, "\mathsf{ctor}"'] \\
A \arrow[u, "s"'] \arrow[r, dotted, "f"] & \mathsf{J_F}
\end{tikzcd}$$

a

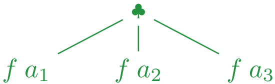
Properties of $\mathsf{J_F}$: Coiteration



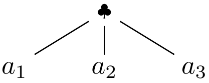


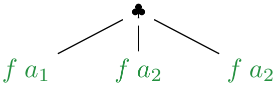
$$\begin{tikzcd} \mathsf{F}\ A \arrow[r, dotted, "\mathsf{F}\ f"'] & \mathsf{F}\ \mathsf{J_F} \arrow[d, "\mathsf{ctor}"'] \\ A \arrow[u, "s"'] \arrow[r, dotted, "f"] & \mathsf{J_F} \end{tikzcd}$$

a



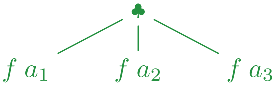
Properties of $\mathsf{J_F}$: Coiteration





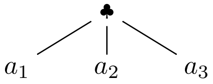
$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, dotted, "\mathsf{F}\ f"'] & \mathsf{F}\ \mathsf{J_F} \arrow[d, "\mathsf{ctor}"'] \\
A \arrow[u, "s"'] \arrow[r, dotted, "f"] & \mathsf{J_F}
\end{tikzcd}$$

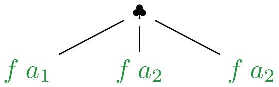
a



a_1, a_2, a_3 are not “smaller” than a in any sense

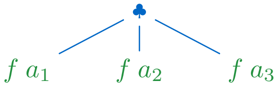
Properties of $\mathsf{J_F}$: Coiteration





$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, dotted, "\mathsf{F}\ f"'] & \mathsf{F}\ \mathsf{J_F} \arrow[d, "\mathsf{ctor}"'] \\
A \arrow[u, "s"'] \arrow[r, dotted, "f"] & \mathsf{J_F}
\end{tikzcd}$$

a



a_1, a_2, a_3 are not “smaller” than a in any sense

But computation has made progress

Properties of J_{F}: Coiteration

$$\begin{tikzcd}
 & \mathsf{F}\,A \\
A \arrow[ur, "s"] \arrow[r, dotted, "f"] & \mathsf{J}_{\mathsf{F}}
\end{tikzcd}$$

Properties of J_{F}: Coiteration

$$\begin{tikzcd}
 & \mathsf{F}\ A \\
A \arrow[ur, "s"] \arrow[r, dotted, "f"] & \mathsf{J}_{\mathsf{F}}
\end{tikzcd}$$

a

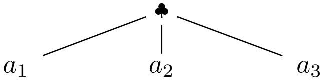
Properties of J_{F}: Coiteration

$s\ a$

$$\begin{tikzcd}
 & \mathsf{F}\ A \\
A \arrow[ur, "s"] \arrow[r, dotted, "f"] & \mathsf{J}_{\mathsf{F}}
\end{tikzcd}$$

a

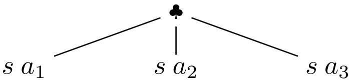
Properties of J_F: Coiteration



$$\begin{tikzcd}
 & \mathsf{F}\ A \\
A \arrow[ur, "s"] \arrow[r, dotted, "f"] & \mathsf{J}_\mathsf{F}
\end{tikzcd}$$

a

Properties of J_{F}: Coiteration



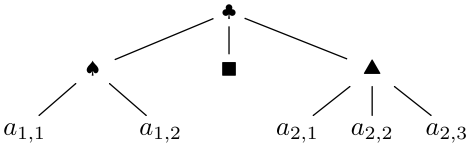
♣

$s\ a_1$ $s\ a_2$ $s\ a_3$

$$\begin{tikzcd}
 & \mathsf{F}\ A \\
A \arrow[ur, "s"] \arrow[r, dotted, "f"] & \mathsf{J}_{\mathsf{F}}
\end{tikzcd}$$

a

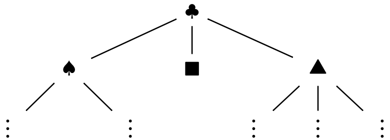
Properties of $\mathsf{J_F}$: Coiteration



$$\begin{tikzcd}
 & \mathsf{F}\ A \\
A \arrow[ur, "s"] \arrow[r, dotted, "f"] & \mathsf{J_F}
\end{tikzcd}$$

a

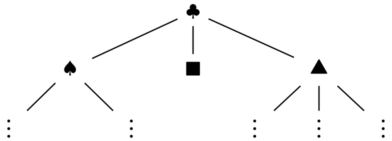
Properties of J_F: Coiteration



$$\begin{tikzcd}
 & \mathsf{F}\ A \\
A \arrow[ur, "s"] \arrow[r, dotted, "f"] & \mathsf{J}_\mathsf{F}
\end{tikzcd}$$

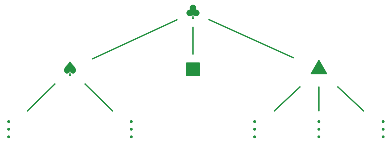
a

Properties of J_F: Coiteration

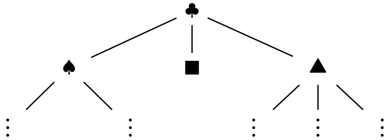


$$\begin{tikzcd}
 & \mathsf{F}\ A \\
A \arrow[ur, "s"] \arrow[r, dotted, "f"] & \mathsf{J}_\mathsf{F}
\end{tikzcd}$$

a



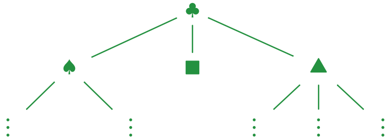
Properties of J_F: Coiteration



$$\begin{tikzcd} & \mathsf{F}\ A \\ A \arrow[ur, "s"] \arrow[r, dotted, "f"] & \mathsf{J}_\mathsf{F} \end{tikzcd}$$

$s\ a$ = the seed encoding the growth of the tree $f\ a$

a



Properties of $\mathsf{J_F}$: Coiteration

Given a natural functor F, $\quad(\mathsf{J_F},\ \mathsf{dtor} : \mathsf{J_F} \to \mathsf{F}\ \mathsf{J_F})$

Coiteration (Final Coalgebra Property): For all $(A, s : A \to \mathsf{F}\ A)$, there exists a unique function coiter_s with

$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, "\mathsf{F}\ \mathsf{coiter}_s"] & \mathsf{F}\ \mathsf{J_F} \arrow[d, "\mathsf{ctor}"] \\
A \arrow[u, "s"] \arrow[r, "\mathsf{coiter}_s"] & \mathsf{J_F}
\end{tikzcd}$$

Properties of $\mathsf{J_F}$: Coiteration

Given a natural functor F, $\quad(\mathsf{J_F},\ \mathsf{dtor} : \mathsf{J_F} \to \mathsf{F}\ \mathsf{J_F})$

Coiteration (Final Coalgebra Property): For all $(A, s : A \to \mathsf{F}\ A)$, there exists a unique function coiter_s with

$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, "\mathsf{F}\ \mathsf{coiter}_s"] & \mathsf{F}\ \mathsf{J_F} \\
A \arrow[u, "s"] \arrow[r, "\mathsf{coiter}_s"] & \mathsf{J_F} \arrow[u, "\mathsf{dtor}"']
\end{tikzcd}$$

Properties of J_F: Coiteration

Given a natural functor F, $(\mathsf{J}_\mathsf{F},\ \mathsf{dtor} : \mathsf{J}_\mathsf{F} \to \mathsf{F}\ \mathsf{J}_\mathsf{F})$

Coiteration (Final Coalgebra Property): For all $(A, s : A \to \mathsf{F}\ A)$, there exists a unique function coiter_s with

$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, "\mathsf{F}\ \mathsf{coiter}_s"] & \mathsf{F}\ \mathsf{J}_\mathsf{F} \\
A \arrow[u, "s"] \arrow[r, "\mathsf{coiter}_s"] & \mathsf{J}_\mathsf{F} \arrow[u, "\mathsf{dtor}"']
\end{tikzcd}$$

$\boxed{\mathsf{J}_\mathsf{F} = \text{the codatatype of F}}$

The $\mathsf{I_F}$ to $\mathsf{J_F}$ embedding revisited

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I_F} \arrow[r, "\mathsf{F}\ \iota"] \arrow[d, shift left, "\mathsf{ctor}"] & \mathsf{F}\ \mathsf{J_F} \arrow[d, shift left, "\mathsf{ctor}"] \\
\mathsf{I_F} \arrow[u, shift left, "\mathsf{dtor}"] \arrow[r, "\iota"] & \mathsf{J_F} \arrow[u, shift left, "\mathsf{dtor}"]
\end{tikzcd}$$

ι can be regarded as defined by iteration on $\mathsf{I_F}$

$\iota = \mathsf{iter}_{\mathsf{ctor}}$

The I_F to J_F embedding revisited

$$\begin{tikzcd}
\mathsf{F}\ \mathsf{I}_\mathsf{F} \arrow[r, "\mathsf{F}\ \iota"] \arrow[d, shift left, "\mathsf{ctor}"] & \mathsf{F}\ \mathsf{J}_\mathsf{F} \arrow[d, shift left, "\mathsf{ctor}"] \\
\mathsf{I}_\mathsf{F} \arrow[u, shift left, "\mathsf{dtor}"] \arrow[r, "\iota"] & \mathsf{J}_\mathsf{F} \arrow[u, shift left, "\mathsf{dtor}"]
\end{tikzcd}$$

ι can be regarded as defined by
iteration on I_F but also by coiteration on J_F!

$\iota = \mathsf{iter}_{\mathsf{ctor}} = \mathsf{coiter}_{\mathsf{dtor}}$

Properties of J_F: Coinduction

j

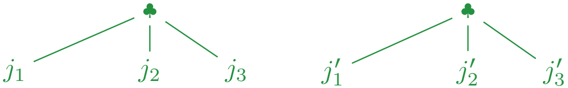
j'

Properties of $\mathsf{J_F}$: Coinduction

j $\qquad$ j'

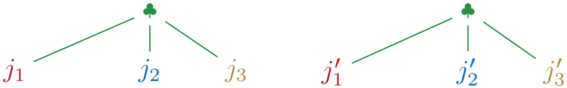
Want: $\quad j = j'$

Properties of $\mathsf{J_F}$: Coinduction



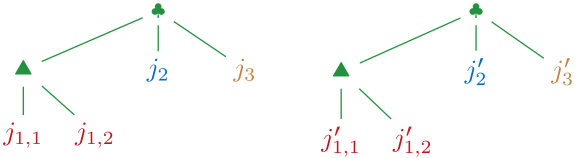
Want: $j = j'$

Properties of J_F: Coinduction



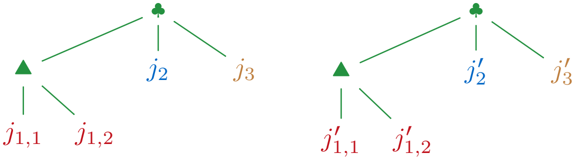
Suffices: $j_1 = j'_1$
$j_2 = j'_2$
$j_3 = j'_3$

Properties of $\mathsf{J_F}$: Coinduction



Suffices:
$$j_1 = j_1'$$
$$j_2 = j_2'$$
$$j_3 = j_3'$$

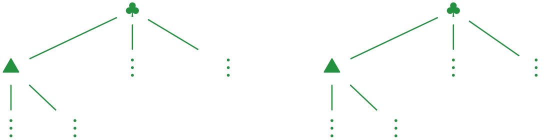
Properties of J_F: Coinduction



Suffices:

$$j_{1,1} = j'_{1,1},\ j_{1,2} = j'_{1,2}$$
$$j_2 = j'_2$$
$$j_3 = j'_3$$

Properties of J_F: Coinduction

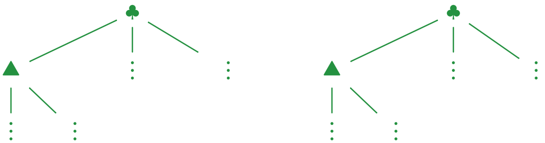


Suffices: $j_{1,1} = j'_{1,1},\ j_{1,2} = j'_{1,2}$

$$j_2 = j'_2$$

$$j_3 = j'_3$$

Properties of $\mathsf{J_F}$: Coinduction



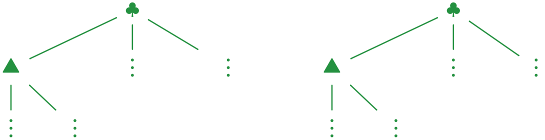
If we can stay in the game indefinitely, then equality holds!

Suffices: $j_{1,1} = j'_{1,1},\ j_{1,2} = j'_{1,2}$

$j_2 = j'_2$

$j_3 = j'_3$

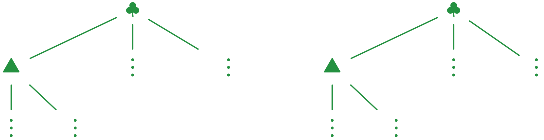
Properties of J_F: Coinduction



If we can stay in the game indefinitely, then equality holds!
But how to show we can "stay in the game"?

Suffices: $$j_{1,1} = j'_{1,1},\ j_{1,2} = j'_{1,2}$$
$$j_2 = j'_2$$
$$j_3 = j'_3$$

Properties of $\mathsf{J_F}$: Coinduction



If we can stay in the game indefinitely, then equality holds!
But how to show we can "stay in the game"?
By exhibiting a "strategy"

Suffices:

$$j_{1,1} = j'_{1,1},\ j_{1,2} = j'_{1,2}$$
$$j_2 = j'_2$$
$$j_3 = j'_3$$

But First: Relators

$$\begin{tikzcd}
A \arrow[d, dash, "R"] \\
B
\end{tikzcd}$$

But First: Relators

$$\begin{tikzcd}
A \arrow[d, dash, "R"] & \mathsf{F}\,A \arrow[d, dash, "\mathsf{Frel}\,R"] \\
B & \mathsf{F}\,B
\end{tikzcd}$$

But First: Relators

$$\begin{tikzcd}
A \arrow[d, dash, "R"] & \mathsf{F}\,A \arrow[d, dash, "\mathsf{Frel}\;R"] \\
B & \mathsf{F}\,B
\end{tikzcd}$$

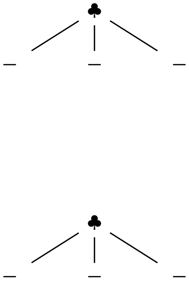
But First: Relators

$$\begin{tikzcd} A \arrow[d, dash, "R"] & \mathsf{F}\,A \arrow[d, dash, "\mathsf{Frel}\,R"] \\ B & \mathsf{F}\,B \end{tikzcd}$$

Two elements of F A and F B are related by Frel R iff

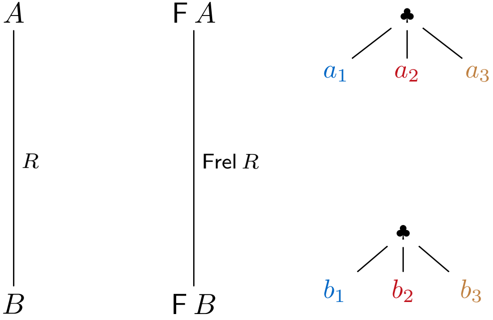
But First: Relators

$$\begin{tikzcd}
A \arrow[d, dash, "R"] & \mathsf{F}\,A \arrow[d, dash, "\mathsf{Frel}\,R"] \\
B & \mathsf{F}\,B
\end{tikzcd}$$



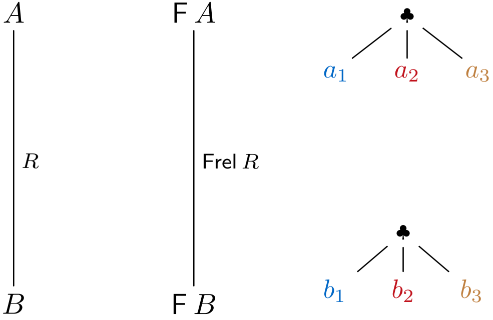
Two elements of F A and F B are related by Frel R iff they have the same shape

But First: Relators



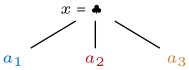
Two elements of $\mathsf{F}\ A$ and $\mathsf{F}\ B$ are related by $\mathsf{Frel}\ R$ iff
they have the same shape
and the contents from corresponding slots are related by R

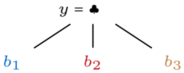
But First: Relators



Two elements of $\mathsf{F}\ A$ and $\mathsf{F}\ B$ are related by $\mathsf{Frel}\ R$ iff
they have the same shape
and the contents from corresponding slots are related by R
$R\ a_1\ b_1$, $R\ a_2\ b_2$, $R\ a_3\ b_3$

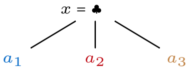
Relator Defined from Mapper

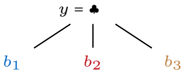




R relation between A and B, $x \in \mathsf{F}\ A$, $y \in \mathsf{F}\ B$

Relator Defined from Mapper

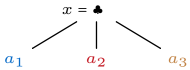


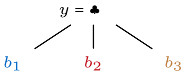


R relation between A and B, $x \in \mathsf{F}\ A$, $y \in \mathsf{F}\ B$

Frel $R\ x\ y$ defined as

Relator Defined from Mapper

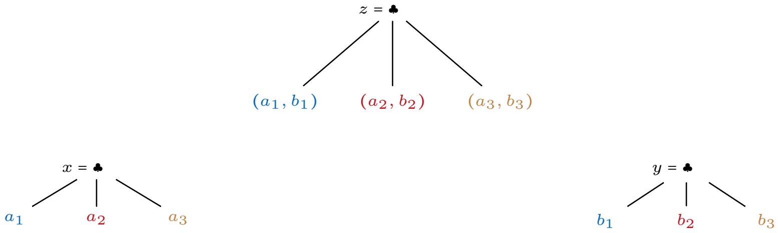




R relation between A and B, $x \in \mathsf{F}\ A$, $y \in \mathsf{F}\ B$

Frel $R\ x\ y$ defined as
$\exists z \in \mathsf{F}\ \{(a, b) \mid R\ a\ b\}.\ \mathsf{F}\ \pi_1\ z = x \wedge \mathsf{F}\ \pi_2\ z = y$

Relator Defined from Mapper



R relation between A and B, $x \in \mathsf{F}\ A$, $y \in \mathsf{F}\ B$

$\mathsf{Frel}\ R\ x\ y$ defined as
$\exists z \in \mathsf{F}\ \{(a, b) \mid R\ a\ b\}.\ \mathsf{F}\ \pi_1\ z = x \wedge \mathsf{F}\ \pi_2\ z = y$

Relators for the Running Examples

R relation between A and B

Relators for the Running Examples

R relation between A and B
Frel R relation between F A and F B

Relators for the Running Examples

R relation between A and B
$\mathsf{Frel}\ R$ relation between $\mathsf{F}\ A$ and $\mathsf{F}\ B$

$\mathsf{F}\ A = \mathbb{N} \times A$ $\quad$ $\mathsf{Frel}\ R\ (m, a)\ (n, b) \Longleftrightarrow$

Relators for the Running Examples

R relation between A and B

$\mathsf{Frel}\ R$ relation between $\mathsf{F}\ A$ and $\mathsf{F}\ B$

$\mathsf{F}\ A = \mathbb{N} \times A \qquad \mathsf{Frel}\ R\ (m, a)\ (n, b) \Leftrightarrow (m = n\ \wedge\ R\ a\ b)$

Relators for the Running Examples

R relation between A and B
$\mathsf{Frel}\ R$ relation between $\mathsf{F}\ A$ and $\mathsf{F}\ B$

$\mathsf{F}\ A = \mathbb{N} \times A \qquad \mathsf{Frel}\ R\ (m, a)\ (n, b) \Leftrightarrow (m = n\ \wedge\ R\ a\ b)$

$\mathsf{F}\ A = \mathbb{N} + A$

Relators for the Running Examples

R relation between A and B

Frel R relation between F A and F B

F $A = \mathbb{N} \times A$ Frel $R\ (m, a)\ (n, b) \Leftrightarrow (m = n\ \wedge\ R\ a\ b)$

Frel $R\ u\ v \Leftrightarrow$

F $A = \mathbb{N} + A$

Relators for the Running Examples

R relation between A and B
Frel R relation between F A and F B

F $A = \mathbb{N} \times A$ Frel $R\ (m, a)\ (n, b) \Leftrightarrow (m = n\ \wedge\ R\ a\ b)$

F $A = \mathbb{N} + A$ Frel $R\ u\ v \Leftrightarrow$
$(\exists n.\ u = v =$ Left $n) \vee$
$(\exists a, b.\ u =$ Right $a\ \wedge\ v =$ Right $b\ \wedge\ R\ a\ b)$

Relators for the Running Examples

R relation between A and B
$\mathsf{Frel}\ R$ relation between $\mathsf{F}\ A$ and $\mathsf{F}\ B$

$\mathsf{F}\ A = \mathbb{N} \times A$	$\mathsf{Frel}\ R\ (m, a)\ (n, b) \Leftrightarrow (m = n\ \wedge\ R\ a\ b)$
$\mathsf{F}\ A = \mathbb{N} + A$	$\mathsf{Frel}\ R\ u\ v \Leftrightarrow$ $(\exists n.\ u = v = \mathsf{Left}\ n) \vee$ $(\exists a, b.\ u = \mathsf{Right}\ a\ \wedge\ v = \mathsf{Right}\ b\ \wedge\ R\ a\ b)$
$\mathsf{F}\ A = \mathsf{List}\ A$	

Relators for the Running Examples

R relation between A and B

Frel R relation between F A and F B

F $A = \mathbb{N} \times A$ $\quad$ Frel $R\,(m, a)\,(n, b) \Leftrightarrow (m = n \,\wedge\, R\,a\,b)$

F $A = \mathbb{N} + A$ $\quad$ Frel $R\,u\,v \Leftrightarrow$

$\quad (\exists n.\ u = v = \mathsf{Left}\ n) \vee$

$\quad (\exists a, b.\ u = \mathsf{Right}\ a \,\wedge\, v = \mathsf{Right}\ b \,\wedge\, R\,a\,b)$

F A = List A $\quad$ Frel $R\,(a_1 \cdot a_2 \cdot \ldots \cdot a_m)\ (b_1 \cdot b_2 \cdot \ldots \cdot b_n) \Leftrightarrow$

Relators for the Running Examples

R relation between A and B

Frel R relation between F A and F B

F $A = \mathbb{N} \times A$ Frel $R\ (m, a)\ (n, b) \Leftrightarrow (m = n \wedge R\ a\ b)$

F $A = \mathbb{N} + A$

Frel $R\ u\ v \Leftrightarrow$
$(\exists n.\ u = v = \text{Left}\ n) \vee$
$(\exists a, b.\ u = \text{Right}\ a \wedge v = \text{Right}\ b \wedge R\ a\ b)$

F A = List A

Frel $R\ (a_1 \cdot a_2 \cdot \ldots \cdot a_m)\ (b_1 \cdot b_2 \cdot \ldots \cdot b_n) \Leftrightarrow$
$m = n \wedge (\forall i.\ R\ a_i\ b_i)$

Back to the “Strategy” for Proving Equality

$$\begin{tikzcd}
\mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
\mathsf{F\ J_F}
\end{tikzcd}$$

Back to the “Strategy” for Proving Equality

$$\begin{tikzcd}
\mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
\mathsf{F}\ \mathsf{J_F}
\end{tikzcd}$$

Given binary relation R on $\mathsf{J_F}$

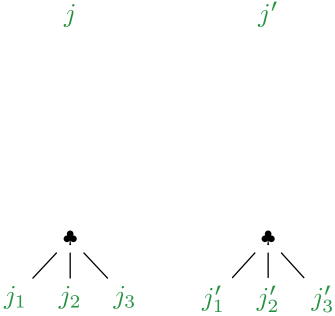
Back to the “Strategy” for Proving Equality

$$\begin{tikzcd}
\mathsf{J_F} \arrow[d, "\mathsf{dtor}"] & j & j' \\
\mathsf{F\ J_F} & &
\end{tikzcd}$$

Given binary relation R on $\mathsf{J_F}$
If $\forall j, j'.\, R\ j\ j'$

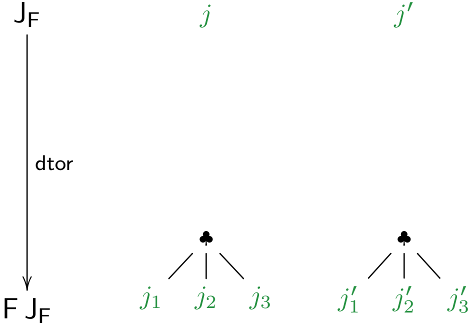
Back to the “Strategy” for Proving Equality

$$\begin{tikzcd}
\mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
\mathsf{F}\ \mathsf{J_F}
\end{tikzcd}$$



Given binary relation R on $\mathsf{J_F}$
If $\forall j, j'.\, R\ j\ j' \Longrightarrow \mathsf{Frel}\ R\ (\mathsf{dtor}\ j)\ (\mathsf{dtor}\ j')$

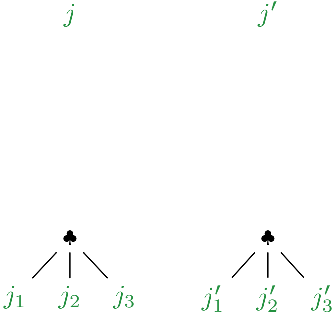
Back to the “Strategy” for Proving Equality



Given binary relation R on $\mathsf{J_F}$
If $\forall j, j'.\ R\ j\ j' \Longrightarrow \mathsf{Frel}\ R\ (\mathsf{dtor}\ j)\ (\mathsf{dtor}\ j')$
Then R is included in equality

Back to the “Strategy” for Proving Equality

$$\begin{tikzcd}
\mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
\mathsf{F}\ \mathsf{J_F}
\end{tikzcd}$$



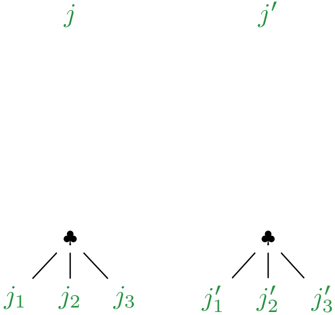
Given binary relation R on $\mathsf{J_F}$

If $\forall j, j'.\ R\ j\ j' \Longrightarrow \mathsf{Frel}\ R\ (\mathsf{dtor}\ j)\ (\mathsf{dtor}\ j')$

Then R is included in equality $\quad \forall j, j'.\ R\ j\ j' \Longrightarrow j = j'$

Back to the “Strategy” for Proving Equality

$$\begin{tikzcd}
\mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
\mathsf{F}\ \mathsf{J_F}
\end{tikzcd}$$



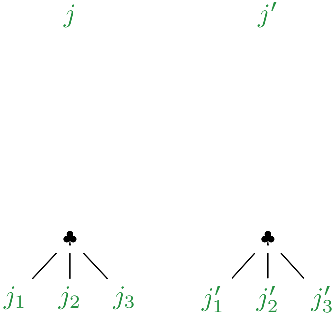
Given binary relation R on $\mathsf{J_F}$

If $\forall j, j'.\ R\ j\ j' \Longrightarrow \mathsf{Frel}\ R\ (\mathsf{dtor}\ j)\ (\mathsf{dtor}\ j')$ $\boxed{R\ \text{F-bisimulation}}$

Then R is included in equality $\quad \forall j, j'.\ R\ j\ j' \Longrightarrow j = j'$

Back to the “Strategy” for Proving Equality

$$\begin{tikzcd}
\mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
\mathsf{F}\ \mathsf{J_F}
\end{tikzcd}$$



Summary: to prove $j = j'$,

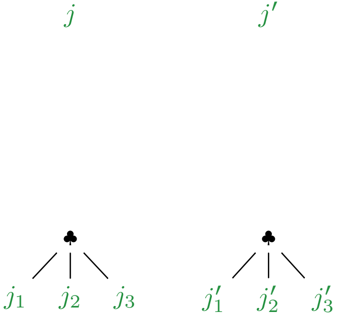
Given binary relation R on $\mathsf{J_F}$

If $\forall j, j'.\ R\ j\ j' \Longrightarrow \mathsf{Frel}\ R\ (\mathsf{dtor}\ j)\ (\mathsf{dtor}\ j')$ $\boxed{R\ \text{F-bisimulation}}$

Then R is included in equality $\quad \forall j, j'.\ R\ j\ j' \Longrightarrow j = j'$

Back to the “Strategy” for Proving Equality

$$\begin{tikzcd}
\mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
\mathsf{F}\ \mathsf{J_F}
\end{tikzcd}$$



Summary: to prove $j = j'$, find F-bisimulation R with $R\ j\ j'$

Given binary relation R on $\mathsf{J_F}$

If $\forall j, j'.\ R\ j\ j' \Longrightarrow$ Frel R (dtor j) (dtor j') $\boxed{R \text{ F-bisimulation}}$

Then R is included in equality $\quad \forall j, j'.\ R\ j\ j' \Longrightarrow j = j'$

Summary for J_F

Given a natural functor F, $(\mathsf{J}_\mathsf{F},\ \mathsf{dtor} : \mathsf{J}_\mathsf{F} \to \mathsf{F}\ \mathsf{J}_\mathsf{F})$ satisfies:

Summary for J_F

Given a natural functor F, $(J_F,\ \mathrm{dtor} : J_F \to F\ J_F)$ satisfies:

dtor bijection

Summary for J_F

Given a natural functor F, $(\mathsf{J}_\mathsf{F},\ \mathsf{dtor} : \mathsf{J}_\mathsf{F} \to \mathsf{F}\ \mathsf{J}_\mathsf{F})$ satisfies:

dtor bijection

Coiteration (Final Coalgebra Property): For all $(A, s : A \to \mathsf{F}\ A)$, there exists a unique function coiter_s with

$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, "\mathsf{F}\ \mathsf{coiter}_s"] & \mathsf{F}\ \mathsf{J}_\mathsf{F} \\
A \arrow[u, "s"] \arrow[r, "\mathsf{coiter}_s"] & \mathsf{J}_\mathsf{F} \arrow[u, "\mathsf{dtor}"']
\end{tikzcd}$$

Summary for $\mathsf{J_F}$

Given a natural functor F, $(\mathsf{J_F},\ \mathsf{dtor} : \mathsf{J_F} \to \mathsf{F}\ \mathsf{J_F})$ satisfies:

dtor bijection

Coiteration (Final Coalgebra Property): For all $(A, s : A \to \mathsf{F}\ A)$, there exists a unique function coiter_s with

$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, "\mathsf{F}\ \mathsf{coiter}_s"] & \mathsf{F}\ \mathsf{J_F} \\
A \arrow[u, "s"] \arrow[r, "\mathsf{coiter}_s"] & \mathsf{J_F} \arrow[u, "\mathsf{dtor}"']
\end{tikzcd}$$

Coinduction: Given any binary relation R on $\mathsf{J_F}$

$$\frac{R \text{ is an F-bisimulation}}{\forall j, j'.\ R\ j\ j' \Longrightarrow j = j'}$$

Summary for J_F

Given a natural functor F, $(\mathsf{J}_\mathsf{F},\ \mathsf{dtor} : \mathsf{J}_\mathsf{F} \to \mathsf{F}\ \mathsf{J}_\mathsf{F})$ satisfies:

dtor bijection

Coiteration (Final Coalgebra Property): For all $(A, s : A \to \mathsf{F}\ A)$, there exists a unique function coiter_s with

$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, "\mathsf{F}\ \mathsf{coiter}_s"] & \mathsf{F}\ \mathsf{J}_\mathsf{F} \\
A \arrow[u, "s"] \arrow[r, "\mathsf{coiter}_s"] & \mathsf{J}_\mathsf{F} \arrow[u, "\mathsf{dtor}"']
\end{tikzcd}$$

Coinduction: Given any binary relation R on J_F

$$\frac{\forall j, j'.\ R\ j\ j' \Longrightarrow \mathsf{Frel}\ R\ (\mathsf{dtor}\ j)\ (\mathsf{dtor}\ j')}{\forall j, j'.\ R\ j\ j' \Longrightarrow j = j'}$$

Summary for $\mathsf{J_F}$

Given a natural functor F, $(\mathsf{J_F},\ \mathsf{dtor} : \mathsf{J_F} \to \mathsf{F}\ \mathsf{J_F})$ satisfies:

dtor bijection

$$\boxed{\mathsf{J_F} = \text{the codatatype of F}}$$

Coiteration (Final Coalgebra Property): For all $(A, s : A \to \mathsf{F}\ A)$, there exists a unique function coiter_s with

$$\begin{tikzcd}
\mathsf{F}\ A \arrow[r, "\mathsf{F}\ \mathsf{coiter}_s"] & \mathsf{F}\ \mathsf{J_F} \\
A \arrow[u, "s"] \arrow[r, "\mathsf{coiter}_s"] & \mathsf{J_F} \arrow[u, "\mathsf{dtor}"']
\end{tikzcd}$$

Coinduction: Given any binary relation R on $\mathsf{J_F}$

$$\frac{\forall j, j'.\ R\ j\ j' \Longrightarrow \mathsf{Frel}\ R\ (\mathsf{dtor}\ j)\ (\mathsf{dtor}\ j')}{\forall j, j'.\ R\ j\ j' \Longrightarrow j = j'}$$

Example of Codatatype

Let B be a fixed set. $\mathsf{F}\ A = B \times A$

Example of Codatatype

Let B be a fixed set. $\mathsf{F}\ A = B \times A$

The shapes of F:

Example of Codatatype

Let B be a fixed set. $\mathsf{F}\,A = B \times A$

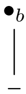
The shapes of F: $(b, \_)$ for each $b \in B$

Example of Codatatype

Let B be a fixed set. $\mathsf{F}\ A = B \times A$

The shapes of F: $(b, \_)$ for each $b \in B$

Or, graphically:



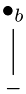
for each $b \in B$

Example of Codatatype

Let B be a fixed set. $\mathsf{F}\, A = B \times A$

The shapes of F: $(b, \_)$ for each $b \in B$

Or, graphically:



for each $b \in B$

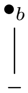
Who is J_{F}?

Example of Codatatype

Let B be a fixed set. $\mathsf{F}\ A = B \times A$

The shapes of F: $(b, \_)$ for each $b \in B$

Or, graphically:



for each $b \in B$

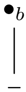
Who is J_F?
Its elements have the form $(b_1, (b_2, \ldots, (b_n, \ldots$

Example of Codatatype

Let B be a fixed set. F $A = B \times A$

The shapes of F: $(b, \_)$ for each $b \in B$

Or, graphically:



for each $b \in B$

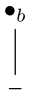
Who is J_{F}?
Its elements have the form $(b_1, (b_2, \ldots, (b_n, \ldots$
I.e., essentially streams $b_1 \cdot b_2 \cdot \ldots \cdot b_n \cdot \ldots$

Example of Codatatype

Let B be a fixed set. $\quad \mathsf{F}\ A = B \times A$

The shapes of F: $\quad (b, \_)$ for each $b \in B$

Or, graphically:



for each $b \in B$

Who is J_{F}?

Its elements have the form $(b_1, (b_2, \ldots, (b_n, \ldots$

I.e., essentially streams $b_1 \cdot b_2 \cdot \ldots \cdot b_n \cdot \ldots$

So $\mathsf{J}_{\mathsf{F}} = \mathsf{Stream}_B$

Example of Codatatype: Stream

B fixed $\quad \mathsf{F}\ A = B \times A \quad f = \mathsf{coiter}_s \quad \mathsf{J_F} = \mathsf{Stream}_B$

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "s"'] & \mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
\mathsf{F}\ A \arrow[r, "\mathsf{F}\ f"] & \mathsf{F}\ \mathsf{J_F}
\end{tikzcd}$$

$\mathsf{dtor}\ (f\ a) = (\mathsf{F}\ f)\ (s\ a)$

Example of Codatatype: Stream

B fixed $\mathsf{F}\ A = B \times A$ $f = \mathsf{coiter}_s$ $\mathsf{J}_\mathsf{F} = \mathsf{Stream}_B$

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "s"'] & \mathsf{J}_\mathsf{F} \arrow[d, "\mathsf{dtor}"] \\
B \times A \arrow[r, "B\times f"] & B \times \mathsf{J}_\mathsf{F}
\end{tikzcd}$$

$\mathsf{dtor}\ (f\ a) = (\mathsf{F}\ f)\ (s\ a)$

Example of Codatatype: Stream

B fixed $\mathsf{F}\ A = B \times A$ $f = \mathsf{coiter}_s$ $\mathsf{J_F} = \mathsf{Stream}_B$

Define: $\mathsf{hd} = \pi_1 \circ \mathsf{dtor}$ $\mathsf{tl} = \pi_2 \circ \mathsf{dtor}$
$\mathsf{hd}^A = \pi_1 \circ s$ $\mathsf{tl}^A = \pi_2 \circ s$

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "s"'] & \mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
B \times A \arrow[r, "B \times f"] & B \times \mathsf{J_F}
\end{tikzcd}$$

$$\mathsf{dtor}\ (f\ a) = (\mathsf{F}\ f)\ (s\ a)$$

Example of Codatatype: Stream

B fixed $\mathsf{F}\ A = B \times A$ $f = \mathsf{coiter}_s$ $\mathsf{J_F} = \mathsf{Stream}_B$

Define: $\mathsf{hd} = \pi_1 \circ \mathsf{dtor}$ $\mathsf{tl} = \pi_2 \circ \mathsf{dtor}$

$\mathsf{hd}^A = \pi_1 \circ s$ $\mathsf{tl}^A = \pi_2 \circ s$

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "\langle \mathsf{hd}^A{,}\mathsf{tl}^A \rangle"'] & \mathsf{J_F} \arrow[d, "\langle \mathsf{hd}{,}\mathsf{tl} \rangle"] \\
B \times A \arrow[r, "B \times f"] & B \times \mathsf{J_F}
\end{tikzcd}$$

$\mathsf{dtor}\ (f\ a) = (\mathsf{F}\ f)\ (s\ a)$

Example of Codatatype: Stream

B fixed $\quad \mathsf{F}\ A = B \times A \quad f = \mathsf{coiter}_s \quad \mathsf{J_F} = \mathsf{Stream}_B$

Define: $\mathsf{hd} = \pi_1 \circ \mathsf{dtor} \quad \mathsf{tl} = \pi_2 \circ \mathsf{dtor}$

$\mathsf{hd}^A = \pi_1 \circ s \quad \mathsf{tl}^A = \pi_2 \circ s$

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[dr, "\mathsf{hd}^A"'] & \mathsf{J_F} \arrow[d, "\mathsf{hd}"] \\
 & B
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "\mathsf{tl}^A"'] & \mathsf{J_F} \arrow[d, "\mathsf{hd}"] \\
A \arrow[r, "f"] & B
\end{tikzcd}$$

$\mathsf{dtor}\ (f\ a) = (\mathsf{F}\ f)\ (s\ a)$

Example of Codatatype: Stream

B fixed $\mathsf{F}\ A = B \times A$ $f = \mathsf{coiter}_s$ $\mathsf{J}_\mathsf{F} = \mathsf{Stream}_B$

Define: $\mathsf{hd} = \pi_1 \circ \mathsf{dtor}$ $\mathsf{tl} = \pi_2 \circ \mathsf{dtor}$
$\mathsf{hd}^A = \pi_1 \circ s$ $\mathsf{tl}^A = \pi_2 \circ s$

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[dr, "\mathsf{hd}^A"'] & \mathsf{J}_\mathsf{F} \arrow[d, "\mathsf{hd}"] \\
& B
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "\mathsf{tl}^A"'] & \mathsf{J}_\mathsf{F} \arrow[d, "\mathsf{hd}"] \\
A \arrow[r, "f"] & B
\end{tikzcd}$$

$\mathsf{hd}\ (f\ a) = \mathsf{hd}^A\ a$
$\mathsf{tl}\ (f\ a) = f\ (\mathsf{tl}^A\ a)$

Example of Codatatype: Stream

B fixed $\quad \mathsf{F}\ A = B \times A \quad f = \mathsf{coiter}_s \quad \mathsf{J}_\mathsf{F} = \mathsf{Stream}_B$

Define: $\begin{array}{ll} \mathsf{hd} = \pi_1 \circ \mathsf{dtor} & \mathsf{tl} = \pi_2 \circ \mathsf{dtor} \\ \mathsf{hd}^A = \pi_1 \circ s & \mathsf{tl}^A = \pi_2 \circ s \end{array}$

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[dr, "\mathsf{hd}^A"'] & \mathsf{J}_\mathsf{F} \arrow[d, "\mathsf{hd}"] \\
 & B
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "\mathsf{tl}^A"'] & \mathsf{J}_\mathsf{F} \arrow[d, "\mathsf{hd}"] \\
A \arrow[r, "f"] & B
\end{tikzcd}$$

$\mathsf{hd}\ (f\ a) = \mathsf{hd}^A\ a$

$\mathsf{tl}\ (f\ a) = f\ (\mathsf{tl}^A\ a)$

Standard stream coiteration

Example of Codatatype: Stream

B fixed $\quad \mathsf{F}\ {\color{blue}A} = B \times {\color{blue}A} \qquad \mathsf{J_F} = \mathsf{Stream}_B$

$$\begin{tikzcd}
\mathsf{J_F} \arrow[r, dash, "R"] \arrow[d, "\mathsf{dtor}"'] & \mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
\mathsf{F}\ \mathsf{J_F} \arrow[r, dash, "\mathsf{Frel}\ R"] & \mathsf{F}\ \mathsf{J_F}
\end{tikzcd}$$

$$\frac{R \text{ is an F-bisimulation}}{\forall j, j'.\ R\ j\ j' \Longrightarrow j = j'}$$

Example of Codatatype: Stream

B fixed $\quad \mathsf{F}\ A = B \times A \qquad \mathsf{J_F} = \mathsf{Stream}_B$

$$\begin{tikzcd}
\mathsf{J_F} \arrow[rr, dash, "R"] \arrow[d, "\mathsf{dtor}"'] & & \mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
B \times \mathsf{J_F} \arrow[rr, dash, "((b,j),(b',j')) \mapsto b=b' \wedge R\ j\ j'"] & & B \times \mathsf{J_F}
\end{tikzcd}$$

$$\frac{R \text{ is an F-bisimulation}}{\forall j, j'.\ R\ j\ j' \Longrightarrow j = j'}$$

Example of Codatatype: Stream

B fixed $\mathsf{F}\ A = B \times A$ $\mathsf{J_F} = \mathsf{Stream}_B$

$\mathsf{hd} = \pi_1 \circ \mathsf{dtor}$ $\mathsf{tl} = \pi_2 \circ \mathsf{dtor}$

$$\begin{tikzcd}
\mathsf{J_F} \arrow[r, dash, "R"] \arrow[d, "\mathsf{dtor}"'] & \mathsf{J_F} \arrow[d, "\mathsf{dtor}"] \\
B \times \mathsf{J_F} \arrow[r, dash, "((b,j),(b',j')) \mapsto b=b' \wedge R\ j\ j'"] & B \times \mathsf{J_F}
\end{tikzcd}$$

$$\frac{R \text{ is an F-bisimulation}}{\forall j, j'.\ R\ j\ j' \Longrightarrow j = j'}$$

Example of Codatatype: Stream

B fixed $\quad \mathsf{F}\ A = B \times A \qquad \mathsf{J_F} = \mathsf{Stream}_B$

$\mathsf{hd} = \pi_1 \circ \mathsf{dtor} \quad \mathsf{tl} = \pi_2 \circ \mathsf{dtor}$

$$\begin{tikzcd}
\mathsf{J_F} \arrow[r, dash, "R"] \arrow[d, "\langle \mathsf{hd}{,}\mathsf{tl} \rangle"'] & \mathsf{J_F} \arrow[d, "\langle \mathsf{hd}{,}\mathsf{tl} \rangle"] \\
B \times \mathsf{J_F} \arrow[r, dash, "((b{,}j){,}(b'{,}j')) \mapsto b{=}b' \wedge R\ j\ j'"] & B \times \mathsf{J_F}
\end{tikzcd}$$

$$\frac{R \text{ is an F-bisimulation}}{\forall j, j'.\ R\ j\ j' \Longrightarrow j = j'}$$

Example of Codatatype: Stream

B fixed $\quad \mathsf{F}\ A = B \times A \qquad \mathsf{J_F} = \mathsf{Stream}_B$

$\mathsf{hd} = \pi_1 \circ \mathsf{dtor} \quad \mathsf{tl} = \pi_2 \circ \mathsf{dtor}$

$$\begin{tikzcd}
\mathsf{J_F} \arrow[r, dash, "R"] \arrow[d, "\langle \mathsf{hd},\mathsf{tl} \rangle"'] & \mathsf{J_F} \arrow[d, "\langle \mathsf{hd},\mathsf{tl} \rangle"] \\
B \times \mathsf{J_F} \arrow[r, dash, "((b,j),(b',j')) \mapsto b = b' \wedge R\ j\ j'"] & B \times \mathsf{J_F}
\end{tikzcd}$$

$$\frac{\forall j, j'.\ R\ j\ j' \Longrightarrow \mathsf{Frel}\ R\ (\mathsf{dtor}\ j)\ (\mathsf{dtor}\ j')}{\forall j, j'.\ R\ j\ j' \Longrightarrow j = j'}$$

Example of Codatatype: Stream

B fixed $\quad \mathsf{F}\ A = B \times A \qquad \mathsf{J}_\mathsf{F} = \mathsf{Stream}_B$

$\mathsf{hd} = \pi_1 \circ \mathsf{dtor} \quad \mathsf{tl} = \pi_2 \circ \mathsf{dtor}$

$$\begin{tikzcd}
\mathsf{J}_\mathsf{F} \arrow[rrr, dash, "R"] \arrow[d, "\langle \mathsf{hd},\mathsf{tl} \rangle"'] & & & \mathsf{J}_\mathsf{F} \arrow[d, "\langle \mathsf{hd},\mathsf{tl} \rangle"] \\
B \times \mathsf{J}_\mathsf{F} \arrow[rrr, dash, "((b,j),(b',j')) \mapsto b=b' \wedge R\ j\ j'"] & & & B \times \mathsf{J}_\mathsf{F}
\end{tikzcd}$$

$$\frac{\forall j, j'.\ R\ j\ j' \Longrightarrow \mathsf{Frel}\ R\ (\mathsf{hd}\ j, \mathsf{tl}\ j)\ (\mathsf{hd}\ j', \mathsf{tl}\ j')}{\forall j, j'.\ R\ j\ j' \Longrightarrow j = j'}$$

Example of Codatatype: Stream

B fixed $\quad \mathsf{F}\ A = B \times A \qquad \mathsf{J_F} = \mathsf{Stream}_B$

$\mathsf{hd} = \pi_1 \circ \mathsf{dtor} \quad \mathsf{tl} = \pi_2 \circ \mathsf{dtor}$

$$\begin{tikzcd}
\mathsf{J_F} \arrow[r, dash, "R"] \arrow[d, "\langle \mathsf{hd},\mathsf{tl} \rangle"'] & \mathsf{J_F} \arrow[d, "\langle \mathsf{hd},\mathsf{tl} \rangle"] \\
B \times \mathsf{J_F} \arrow[r, dash, "((b,j),(b',j')) \mapsto b=b' \wedge R\ j\ j'"] & B \times \mathsf{J_F}
\end{tikzcd}$$

$$\frac{\forall j, j'.\ R\ j\ j' \Longrightarrow \mathsf{hd}\ j = \mathsf{hd}\ j' \ \wedge\ R\ (\mathsf{tl}\ j)\ (\mathsf{tl}\ j')}{\forall j, j'.\ R\ j\ j' \Longrightarrow j = j'}$$

Concrete Example of Coiteration

$$\begin{aligned}
&even : \mathsf{Stream}_B \to \mathsf{Stream}_B \\
&\quad \mathsf{hd}\;(even\ j) = \mathsf{hd}\ j \\
&\quad \mathsf{tl}\;(even\ j) = even\ (\mathsf{tl}\,(\mathsf{tl}\ j))
\end{aligned}$$

Concrete Example of Coiteration

$$\begin{aligned}
&\mathit{even} : \mathsf{Stream}_B \to \mathsf{Stream}_B\\
&\quad \mathsf{hd}\ (\mathit{even}\ j) = \mathsf{hd}\ j\\
&\quad \mathsf{tl}\ (\mathit{even}\ j) = \mathit{even}\ (\mathsf{tl}\ (\mathsf{tl}\ j))
\end{aligned}$$

$$\begin{aligned}
&\mathsf{odd} : \mathsf{Stream}_B \to \mathsf{Stream}_B\\
&\quad \mathsf{hd}\ (\mathsf{odd}\ j) = \mathsf{hd}\ (\mathsf{tl}\ j)\\
&\quad \mathsf{tl}\ (\mathsf{odd}\ j) = \mathsf{odd}\ (\mathsf{tl}\ (\mathsf{tl}\ j))
\end{aligned}$$

Concrete Example of Coiteration

$$\begin{aligned}
&\mathit{even} : \mathsf{Stream}_B \to \mathsf{Stream}_B \\
&\quad \mathsf{hd}\,(\mathit{even}\ j) = \mathsf{hd}\ j \\
&\quad \mathsf{tl}\,(\mathit{even}\ j) = \mathit{even}\ (\mathsf{tl}\,(\mathsf{tl}\ j))
\end{aligned}$$

$$\begin{aligned}
&\mathsf{odd} : \mathsf{Stream}_B \to \mathsf{Stream}_B \\
&\quad \mathsf{hd}\,(\mathsf{odd}\ j) = \mathsf{hd}\ (\mathsf{tl}\ j) \\
&\quad \mathsf{tl}\,(\mathsf{odd}\ j) = \mathsf{odd}\ (\mathsf{tl}\,(\mathsf{tl}\ j))
\end{aligned}$$

$$\begin{aligned}
&\mathsf{zip} : \mathsf{Stream}_B \times \mathsf{Stream}_B \to \mathsf{Stream}_B \\
&\quad \mathsf{hd}\,(\mathsf{zip}\,(j_1, j_2)) = \mathsf{hd}\ j_1 \\
&\quad \mathsf{tl}\,(\mathsf{zip}\,(j_1, j_2)) = \mathsf{zip}\ (j_2,\ \mathsf{tl}\ j_1)
\end{aligned}$$

Incremental Proof by Structural Coinduction

$\mathsf{zip}\ (even\ j, \mathsf{odd}\ j) = j$

Incremental Proof by Structural Coinduction

$\mathsf{zip}\,(\mathit{even}\;j, \mathsf{odd}\;j) = j$

$\mathsf{tl}\,(\mathsf{zip}\,(\mathit{even}\;j, \mathsf{odd}\;j)) = \mathsf{tl}\;j$

$\mathsf{hd}\,(\mathsf{zip}\,(\mathit{even}\;j, \mathsf{odd}\;j)) = \mathsf{hd}\;j$

Incremental Proof by Structural Coinduction

$\mathsf{zip}\,(\mathit{even}\ j, \mathsf{odd}\ j) = j$

$\mathsf{tl}\,(\mathsf{zip}\,(\mathit{even}\ j, \mathsf{odd}\ j)) = \mathsf{tl}\ j$

$\mathsf{hd}\,(\mathsf{zip}\,(\mathit{even}\ j, \mathsf{odd}\ j)) = \mathsf{hd}\ j$

Incremental Proof by Structural Coinduction

$\mathsf{zip}\,(even\ j, \mathsf{odd}\ j) = j$

$\mathsf{zip}\,(\mathsf{odd}\ j,\ \mathsf{tl}\,(even\ j)) = \mathsf{tl}\ j$

$\mathsf{hd}\,(\mathsf{zip}\,(even\ j, \mathsf{odd}\ j)) = \mathsf{hd}\ j$

Incremental Proof by Structural Coinduction

$\mathsf{zip}\,(\mathit{even}\ j, \mathsf{odd}\ j) = j$

$\mathsf{zip}\,(\mathsf{odd}\ j,\ \mathit{even}\,(\mathsf{tl}\,(\mathsf{tl}\ j))) = \mathsf{tl}\ j$

$\mathsf{hd}\,(\mathsf{zip}\,(\mathit{even}\ j, \mathsf{odd}\ j)) = \mathsf{hd}\ j$

Incremental Proof by Structural Coinduction

$\mathsf{zip}\,(\mathit{even}\ j, \mathsf{odd}\ j) = j$

$\mathsf{zip}\,(\mathsf{odd}\ j,\ \mathit{even}\,(\mathsf{tl}\,(\mathsf{tl}\ j))) = \mathsf{tl}\ j$

$\mathsf{tl}\,(\mathsf{zip}\,(\mathsf{odd}\ j,\ \mathit{even}\,(\mathsf{tl}\,(\mathsf{tl}\ j))) = \mathsf{tl}\,(\mathsf{tl}\ j)$

$\mathsf{hd}\,(\mathsf{zip}\,(\mathit{even}\ j, \mathsf{odd}\ j)) = \mathsf{hd}\ j$

$\mathsf{hd}\ \ldots = \mathsf{hd}\,(\mathsf{tl}\ j)$

Incremental Proof by Structural Coinduction

$\mathsf{zip}\,(\mathit{even}\ j, \mathsf{odd}\ j) = j$

$\mathsf{zip}\,(\mathsf{odd}\ j,\ \mathit{even}\,(\mathsf{tl}\,(\mathsf{tl}\ j))) = \mathsf{tl}\ j$ $\qquad$ $\mathsf{hd}\,(\mathsf{zip}\,(\mathit{even}\ j, \mathsf{odd}\ j)) = \mathsf{hd}\ j$

$\mathsf{zip}\,(\mathit{even}\,(\mathsf{tl}\,(\mathsf{tl}\ j)), \mathsf{odd}\,(\mathsf{tl}\,(\mathsf{tl}\ j))) = \mathsf{tl}\,(\mathsf{tl}\ j)$ $\qquad$ $\mathsf{hd}\ \ldots = \mathsf{hd}\,(\mathsf{tl}\ j)$

Incremental Proof by Structural Coinduction

$\mathsf{zip}\,(even\ j, \mathsf{odd}\ j) = j$

$\mathsf{zip}\,(\mathsf{odd}\ j,\ even\,(\mathsf{tl}\,(\mathsf{tl}\ j))) = \mathsf{tl}\ j$ $\quad$ $\mathsf{hd}\,(\mathsf{zip}\,(even\ j, \mathsf{odd}\ j)) = \mathsf{hd}\ j$

$\mathsf{zip}\,(even\,(\mathsf{tl}\,(\mathsf{tl}\ j)), \mathsf{odd}\,(\mathsf{tl}\,(\mathsf{tl}\ j))) = \mathsf{tl}\,(\mathsf{tl}\ j)$ $\quad$ $\mathsf{hd}\ \ \ldots = \mathsf{hd}\,(\mathsf{tl}\ j)$

Incremental Proof by Structural Coinduction

$\mathsf{zip}\,(\mathit{even}\ j, \mathsf{odd}\ j) = j$

$\mathsf{zip}\,(\mathsf{odd}\ j,\ \mathit{even}\,(\mathsf{tl}\,(\mathsf{tl}\ j))) = \mathsf{tl}\ j$

$\mathsf{hd}\,(\mathsf{zip}\,(\mathit{even}\ j, \mathsf{odd}\ j)) = \mathsf{hd}\ j$

$\mathsf{zip}\,(\mathit{even}\,(\mathsf{tl}\,(\mathsf{tl}\ j)), \mathsf{odd}\,(\mathsf{tl}\,(\mathsf{tl}\ j))) = \mathsf{tl}\,(\mathsf{tl}\ j)$

$\mathsf{hd}\ \ldots = \mathsf{hd}\,(\mathsf{tl}\ j)$

Bisimulation: $R\ j_1\ j_2 \equiv$
$j_1 = \mathsf{zip}\,(\mathit{even}\ j_2, \mathsf{odd}\ j_2) \lor$
$\exists j.\ j_1 = \mathsf{zip}\,(\mathsf{odd}\ j,\ \mathit{even}\,(\mathsf{tl}\,(\mathsf{tl}\ j)))\ \land\ j_2 = \mathsf{tl}\ j$

(Co)datatypes in Isabelle/HOL

(Co)datatypes in Isabelle/HOL

Natural functors are a class of functors

Natural functors are a class of functors
containing the standard basic functors: sum, product, etc.

(Co)datatypes in Isabelle/HOL

Natural functors are a class of functors

- containing the standard basic functors: sum, product, etc.
- closed under the datatype and codatatype constructor

Natural functors are a class of functors
  containing the standard basic functors: sum, product, etc.
  closed under the datatype and codatatype constructor

E.g.: fixing B, List$_B$ is the datatype of $A \mapsto \{*\} + B \times A$

(Co)datatypes in Isabelle/HOL

Natural functors are a class of functors
  containing the standard basic functors: sum, product, etc.
  closed under the datatype and codatatype constructor

E.g.: fixing B, List_B is the datatype of $A \mapsto \{*\} + B \times A$
but $B \mapsto \mathsf{List}_B$ is also a natural functor

(Co)datatypes in Isabelle/HOL

Natural functors are a class of functors
  containing the standard basic functors: sum, product, etc.
  closed under the datatype and codatatype constructor

E.g.: fixing B, List$_B$ is the datatype of $A \mapsto \{*\} + B \times A$
but $B \mapsto$ List$_B$ is also a natural functor
and similarly for $B \mapsto$ Stream$_B$

(Co)datatypes in Isabelle/HOL

Natural functors are a class of functors
containing the standard basic functors: sum, product, etc.
closed under the datatype and codatatype constructor

E.g.: fixing B, List_B is the datatype of $A \mapsto \{*\} + B \times A$
but $B \mapsto \mathsf{List}_B$ is also a natural functor
and similarly for $B \mapsto \mathsf{Stream}_B$

Nesting datatypes in codatatypes or vice versa
allows for modular specs of fancy data structures

Universe of (Co)Datatypes in Isabelle/HOL

The Isabelle system maintains a database of natural functors

Universe of (Co)Datatypes in Isabelle/HOL

The Isabelle system maintains a database of natural functors

User can write high-level specifications:

codatatype Stream A = Cons (hd : A) (tl : List A)

Universe of (Co)Datatypes in Isabelle/HOL

The Isabelle system maintains a database of natural functors

User can write high-level specifications:

codatatype Stream A = Cons (hd : A) (tl : List A)

In the background:

- Isabelle parses this into a natural functor: $B \mapsto B \times A$
- Then infers high-level principles for (co)recursion and (co)induction for Stream
- Finally, Stream is itself registered as a natural functor

Examples

datatype List A = Nil $\mid$ Cons A (List A)

Examples

datatype List A = Nil $\mid$ Cons A (List A)

codatatype LazyList A = Nil $\mid$ Cons A (List A)

Examples

datatype List A = Nil $|$ Cons A (List A)

codatatype LazyList A = Nil $|$ Cons A (List A)

datatype BTree A = Leaf A $|$ Node (BTree A) (BTree A)

Examples

datatype List A = Nil $\mid$ Cons A (List A)

codatatype LazyList A = Nil $\mid$ Cons A (List A)

datatype BTree A = Leaf A $\mid$ Node (BTree A) (BTree A)

datatype Tree A = Node A (List (Tree A))

Examples

datatype List A = Nil | Cons A (List A)

codatatype LazyList A = Nil | Cons A (List A)

datatype BTree A = Leaf A | Node (BTree A) (BTree A)

datatype Tree A = Node A (List (Tree A))

finite-depths, finitely branching
A-labeled trees

Examples

datatype List A = Nil | Cons A (List A)

codatatype LazyList A = Nil | Cons A (List A)

datatype BTree A = Leaf A | Node (BTree A) (BTree A)

datatype Tree A = Node A (Lazy_List (Tree A))

finite-depths, infinitely branching
A-labeled trees

Examples

datatype List A = Nil | Cons A (List A)

codatatype LazyList A = Nil | Cons A (List A)

datatype BTree A = Leaf A | Node (BTree A) (BTree A)

codatatype Tree A = Node A (Lazy_List (Tree A))

possibly infinite-depths, infinitely branching
A-labeled trees

Examples

datatype List A = Nil | Cons A (List A)

codatatype LazyList A = Nil | Cons A (List A)

datatype BTree A = Leaf A | Node (BTree A) (BTree A)

codatatype Tree A = Node A (Lazy_List (Tree A))

possibly infinite-depths, infinitely branching unordered
A-labeled trees

Examples

datatype List A = Nil | Cons A (List A)

codatatype LazyList A = Nil | Cons A (List A)

datatype BTree A = Leaf A | Node (BTree A) (BTree A)

codatatype Tree A = Node A (Countable_Set (Tree A))

possibly infinite-depths, infinitely branching unordered A-labeled trees

Examples

datatype List A = Nil $\mid$ Cons A (List A)

codatatype LazyList A = Nil $\mid$ Cons A (List A)

datatype BTree A = Leaf A $\mid$ Node (BTree A) (BTree A)

codatatype Tree A = Node A (Set$_k$ (Tree A))

possibly infinite-depths, infinitely branching unordered A-labeled trees

Examples

datatype List A = Nil | Cons A (List A)

codatatype LazyList A = Nil | Cons A (List A)

datatype BTree A = Leaf A | Node (BTree A) (BTree A)

codatatype Tree A = Node A (Multi_Set (Tree A))

possibly infinite-depths, infinitely branching unordered A-labeled trees

Examples

datatype List A = Nil | Cons A (List A)

codatatype LazyList A = Nil | Cons A (List A)

datatype BTree A = Leaf A | Node (BTree A) (BTree A)

codatatype Tree A = Node A (Fuzzy_Set (Tree A))

possibly infinite-depths, infinitely branching unordered A-labeled trees

Examples

datatype List A = Nil | Cons A (List A)

codatatype LazyList A = Nil | Cons A (List A)

datatype BTree A = Leaf A | Node (BTree A) (BTree A)

codatatype Tree A = Node A (PLUG_YOUR_OWN (Tree A))

possibly infinite-depths, infinitely branching unordered
A-labeled trees

Examples

datatype List A = Nil $|$ Cons A (List A)

codatatype LazyList A = Nil $|$ Cons A (List A)

datatype BTree A = Leaf A $|$ Node (BTree A) (BTree A)

codatatype Tree A = Node A (PLUG_YOUR_OWN (Tree A))

possibly infinite-depths, infinitely branching unordered A-labeled trees

Examples

datatype List A = Nil $\mid$ Cons A (List A)

codatatype LazyList A = Nil $\mid$ Cons A (List A)

datatype BTree A = Leaf A $\mid$ Node (BTree A) (BTree A)

codatatype Tree A = Node A (PLUG_YOUR_OWN (Tree A))

possibly infinite-depths, infinitely branching unordered A-labeled trees

Examples

datatype List A = Nil | Cons A (List A)

codatatype LazyList A = Nil | Cons A (List A)

datatype BTree A = Leaf A | Node (BTree A) (BTree A)

codatatype Tree A = Node A (PLUG_YOUR_OWN (Tree A))

possibly infinite-depths, infinitely branching unordered A-labeled trees

- Show a set operator to be a bounded natural functor (BNF)
- Register it
- Then Isabelle will allow nesting it in (co)datatype expressions

Summary

Datatypes and codatatypes have intuitive representations in terms of Shape and Content

They form a rich, extendable universe

The proof assistant Isabelle/HOL represents this universe and makes it available to the users

Summary

Datatypes and codatatypes have intuitive representations in terms of Shape and Content

They form a rich, extendable universe

The proof assistant Isabelle/HOL represents this universe and makes it available to the users with a lot of sugar to hide the category theory ☺

Moreover, the abstract constructions have very concrete intuitions

Summary

Datatypes and codatatypes have intuitive representations in terms of Shape and Content

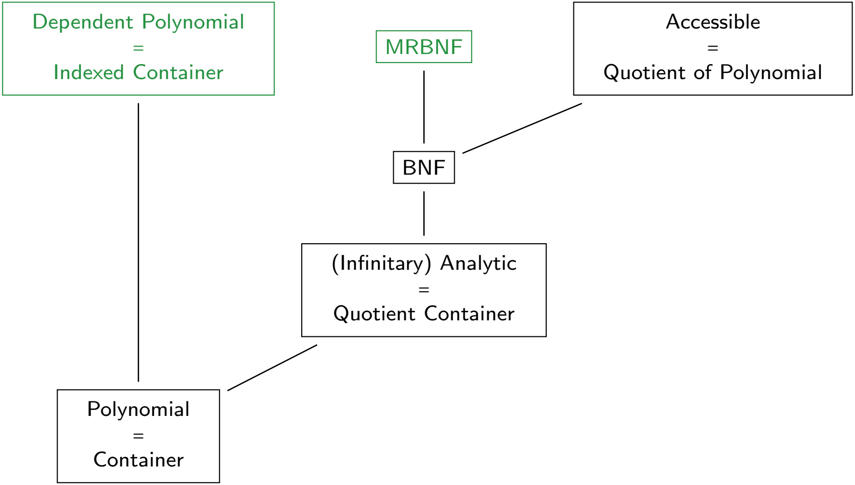
They form a rich, extendable universe

The proof assistant Isabelle/HOL represents this universe and makes it available to the users with a lot of sugar to hide the category theory ☺

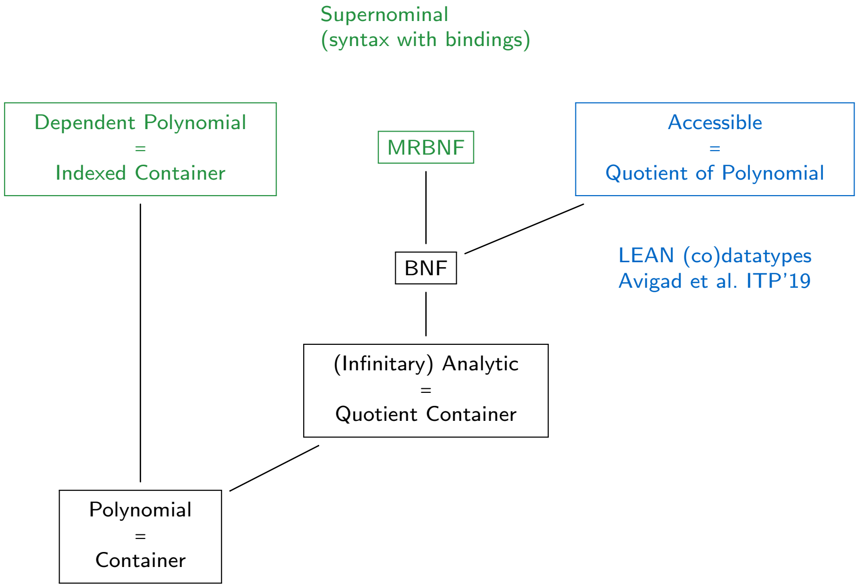
Moreover, the abstract constructions have very concrete intuitions

The abstract reality can be very concrete

Relevant Classes of Functors



Relevant Classes of Functors



Supernominal
(syntax with bindings)

Dependent Polynomial
=
Indexed Container

MRBNF

Accessible
=
Quotient of Polynomial

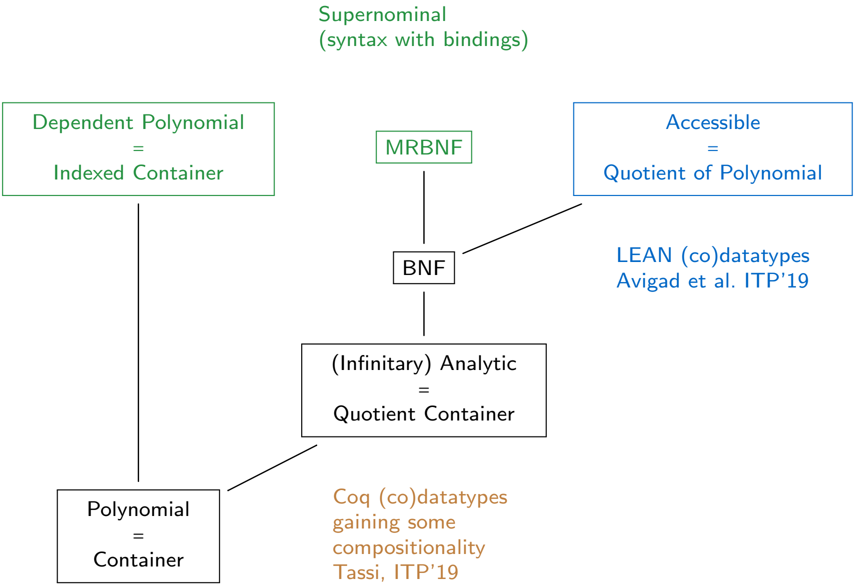
BNF

LEAN (co)datatypes
Avigad et al. ITP'19

(Infinitary) Analytic
=
Quotient Container

Polynomial
=
Container

Relevant Classes of Functors



Supernominal
(syntax with bindings)

Dependent Polynomial
=
Indexed Container

MRBNF

Accessible
=
Quotient of Polynomial

BNF

LEAN (co)datatypes
Avigad et al. ITP'19

(Infinitary) Analytic
=
Quotient Container

Polynomial
=
Container

Coq (co)datatypes
gaining some
compositionality
Tassi, ITP'19

Much more references to relevant literature
will be provided from the course website.