Examples of functors (on Set):

- Identity functor: \( \text{Id}(X) = X \), \( \text{Id}(f) = f \)

- \( F_X = 1 + X \)
  
  If \( X \) is a set, \( F_X \) contains copies of all elements of \( X \) (\( \text{inr}(x) \) for \( x:X \)) plus a new element \( \text{inl}(\ast) \).

- Given a fixed set \( A \).
  
  \( F_X = A \times X \) is the functor we used for streams
  
  \( F_X = A \times X^2 \) is the functor we used for binary trees.

- \( \mathcal{P} \) the powerset functor
  
  \( \mathcal{P}X = \text{subsets of } X \)

  if \( f: X \rightarrow Y \), \( \mathcal{P}f: \mathcal{P}X \rightarrow \mathcal{P}Y \)

  \( U \overset{f}{\rightarrow} \{ fx \mid x \in U \} \)
Containers

Some functors have a nice intuitive explanation and good formal properties.

An element of \( FX \) is thought of as a structure with a shape into which elements of \( X \) are inserted at some positions.

For example, we could have three shapes

![Shapes](image)

The container is characterized by the set of shapes:

\[ S = \{ \circ, \triangle, \square \} \]

And, for every shape, the set of its positions

\[ PO = \{ 0 \} \]
\[ P\triangle = \{ 0, 1, 2 \} \]
\[ P\square = \{ 0, 1 \} \]

A container is a pair \((S, P)\) where

\[ S : \text{Set} \quad \text{pair set of shapes} \]
\[ P : S \rightarrow \text{Set} \quad \text{set of positions for every shape} \]

The functor associated with a container is

\[ F_{(S,P)} X = \sum_{s : S} P_s \rightarrow X \]

\[ \Sigma \text{type: dependent pairs} \]

An element of \( F_{(S,P)} X \) is a pair

\[ \langle s, f \rangle \text{ where } s : S \]
\[ f : P_s \rightarrow X \]

\( s \) is the shape of the element
\( f \) assigns an element of \( X \)

to every position in the shape \( s \).
Many of the functors we have seen are containers

- \( FX = 1 + X \)
  shapes: \( S = \{ L, R \} \)
  positions: \( PL = \emptyset \), \( PR = \{ \text{•} \} \)
  no positions: the element \( \text{inr}(\text{•}) \) doesn't contain any value from \( X \)
  one position: the element \( \text{inr}(x) \) contains one value \( x \in X \)

- \( FX = A \times X \)
  shapes: \( S = A \)
  positions: \( Pa = \{ \text{•} \} \)

- \( FX = A \times X^2 \)
  shapes: \( S = A \)
  positions: \( Pa = \{ L, R \} \)

The powerset functor \( \mathcal{P} \) is not a container

Other examples of containers

- The List functor is a container
  shapes: \( S = \mathbb{N} \) (length of the list)
  positions: \( Pn = \{ 0, 1, \ldots, n-1 \} \) (indices of the elements)

So \( \text{List} \ A = \sum n : S. Pn \rightarrow A \)

The list \( [a_0, a_1, a_2] \) is represented by \( \langle 3, f \rangle \) where \( f : \{ 0, 1, 2 \} \rightarrow A \)
  \( f_i = a_i \)

- The stream functor \( S \) is a container
  shapes: \( S = \{ \text{•} \} \)
  positions: \( P_\text{•} = \mathbb{N} \)

Exercise: Find a container representation for the functor \( T \) of infinite binary trees.
Coalgebra
A coalgebra for a functor $F$ is a pair $(X, \alpha)$ where $X$ is an object and $\alpha : X \to FX$ is a morphism.

Final coalgebra
A coalgebra $(A, \alpha)$ is final if for every coalgebra $(X, \gamma)$ there is a unique morphism $\hat{\gamma} : X \to A$ making the following diagram commute:

If $\hat{\gamma}$ always exists but is not unique, we call $(A, \alpha)$ a weakly-final coalgebra.

Lambek's Lemma:
Every final coalgebra is an isomorphism.

Proof: Let $(A, \alpha)$ be a final coalgebra. $F\alpha : FA \to F^2A$ is also a coalgebra. So there is a unique anamorphism $\bar{\alpha} : FA \to A$.

We can show that $\bar{\alpha}$ is the inverse of $\alpha$.

There is a unique anamorphism from $(A, \alpha)$ to itself: $\bar{\alpha} \circ \alpha = \text{id}_A$.

Then the commutativity of the upper rectangle gives:
$\alpha \circ \bar{\alpha} = F\bar{\alpha} \circ F\alpha = F(\bar{\alpha} \circ \alpha) = F \text{id}_A = \text{id}_{FA}$.
A final coalgebra $(A, d)$ would be an isomorphism $A \cong PA$ by Lambek. But this is impossible by Cantor.

Equality between elements of a final coalgebra.

Final coalgebras model coinductive types whose elements have infinite structure. How can we prove that two infinite things are equal?

Bisimulation

A bisimulation of streams is a relation $R$ on $S_A$ such that:

$$s_1 R s_2 \Rightarrow h s_1 = h s_2 \land t s_1 R t s_2$$

Coinduction Principle:

If $R$ is a bisimulation

$$s_1 R s_2 \Rightarrow s_1 = s_2$$
Generalization: Bisimulation between coalgebras

Let \((X, \alpha : X \to A \times X), (Y, \beta : Y \to A \times Y)\) be two coalgebras.

A bisimulation between \((X, \alpha)\) and \((Y, \beta)\) is a relation \(R\) between \(X\) an \(Y\) such that:

\[ x R y \Rightarrow \pi_1(\alpha x) = \pi_1(\beta y) \land \pi_2(\alpha x) R \pi_2(\beta y) \]

or, with different notation:

\[ x R y \Rightarrow h_{\alpha x} = h_{\alpha y} \land t_{\alpha x} R t_{\beta y} \]

Conduction Principle
If \(R\) is a bisimulation between \((X, \alpha)\) and \((Y, \beta)\),

\[ x R y \Rightarrow \hat{\alpha}(x) = \hat{\beta}(y) \]

In intensional type theory (Coq), coinductive types are weak final coalgebras.

So the Coinduction Principle doesn't hold.

But we can define the appropriate equivalence:

Bisimilarity:
Two streams \(s_1\) and \(s_2\) are bisimilar if there exists a bisimulation \(R\) such that \(s_1 R s_2\).

In that case we write \(s_1 \sim s_2\).

A weakly final (all-encompassing) coalgebra is (strongly) final \(\iff \forall x, y. x \sim y \Rightarrow x = y\).
Bisimulation for trees

A bisimulation on $T_A$ is a relation $R$ on $T_A$ such that

$$t_1 R t_2 \Rightarrow \text{label}(t_1) = \text{label}(t_2) \land \text{left}(t_1) R \text{left}(t_2) \land \text{right}(t_1) R \text{right}(t_2).$$

As for streams, we can generalize the definition to any coalgebras.

$$t_1 \sim t_2 \text{ if there exists a bisimulation } R \text{ such that } t_1 R t_2$$

The coinduction principle holds for trees.

Bisimulations for a generic functor.

In a category $C$, a relation between objects $A$ and $B$ is represented by a span:

$$\begin{array}{c}
R \\
\Rightarrow \ \\
P_1 \Rightarrow A \\
P_2 \Rightarrow B
\end{array}$$

idea: $R$ is the set of pairs that are related, $p_1$ and $p_2$ give the two related elements.

more general idea:

$R$ is the set of proofs of the relation, $p_1$ and $p_2$ give the elements related by the proof.
A relation/span can be lifted to a functor:

\[
\begin{array}{c}
\text{A} \\
\alpha \quad \text{FA} \\
\text{B} \\
\beta \\
\end{array}
\]

Intuition for containers:

Two elements \( \alpha : FA \) and \( \gamma : FB \) are related by \( FR \) if

- they have the same shape
- components in the same position are related by \( R \).

A bisimulation between coalgebras \((A,\alpha), (B,\beta)\) is a span \((R,\rho,\rho)\) that "implies" its lifting through the coalgebras.

This means:

There is a coalgebra structure on \( R \)

\( \rho : R \rightarrow FR \) that makes this diagram commute.

So a bisimulation is just a coalgebra \((R,\rho)\) with coalgebra morphisms to \((A,\alpha)\) and \((B,\beta)\).

Exercise:

Verify that when you instantiate this definition for the functors \( FX = A \times X \) and \( FX = A \times X^2 \) you get the notion of bisimulation for streams and binary trees.
Bisimilarity can itself be defined by coinductive means.

For streams:

Bisimilarity ($\sim$) is the relation on $\mathbb{S}_A$ coinductively defined by the rule

$$a : A \quad s_1, s_2 : \mathbb{S}_A \quad s_1 \sim s_2$$

$$\frac{}{(a \preceq s_1) \sim (a \preceq s_2)}$$

A proof of $s_1 \sim s_2$ then requires that $\nu s_1 = \nu s_2$ and that we have a proof that $\tau s_1 \sim \tau s_2$ which is in turn constructed by the rule in an infinite regression.

We can define such a proof by a guarded fixpoints, in the same way as we define elements of a coinductive type.

Coq allows us to define coinductive predicates and relations and to construct infinite proofs.

Coq definition of bisimilarity and coinductive predicates.