

Introduction to Universal Coalgebra

Ezra Schoen

June 29, 2021

- ① Basic definitions and examples
- ② Final coalgebras and corecursion
- ③ Behavioral equivalence and bisimulation
- ④ Modal logic

Basic definitions and examples

Definition

$B : \mathbb{C} \rightarrow \mathbb{C}$ a functor, S an object in \mathbb{C}

Definition

$B : \mathbb{C} \rightarrow \mathbb{C}$ a functor, S an object in \mathbb{C}

$$\sigma : S \rightarrow BS$$

Definition

$B : \mathbb{C} \rightarrow \mathbb{C}$ a functor, S an object in \mathbb{C}

$$\sigma : S \rightarrow BS$$

Morphisms

$$\begin{array}{ccc} S & \xrightarrow{f} & S' \\ \downarrow \sigma & & \downarrow \tau \\ BS & \xrightarrow{Bf} & BS' \end{array}$$

Definition

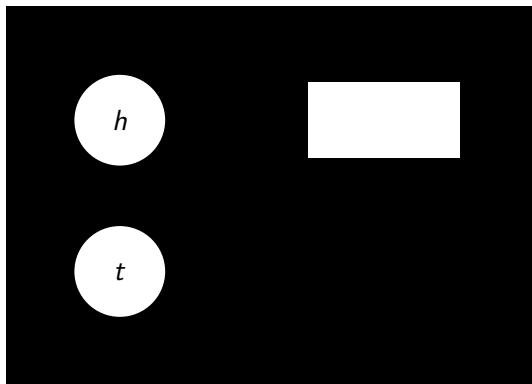
$B : \text{Set} \rightarrow \text{Set}$ a functor, S a set of states

$$\sigma : S \rightarrow BS$$

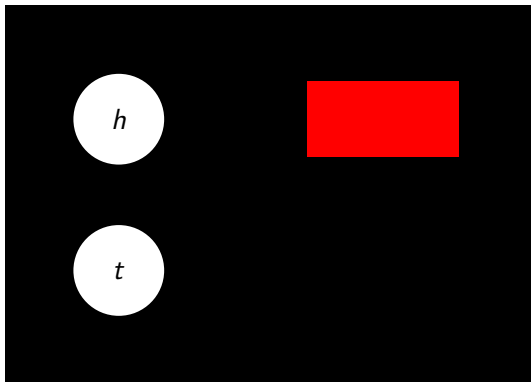
Morphisms

$$\begin{array}{ccc} S & \xrightarrow{f} & S' \\ \downarrow \sigma & & \downarrow \tau \\ BS & \xrightarrow{Bf} & BS' \end{array}$$

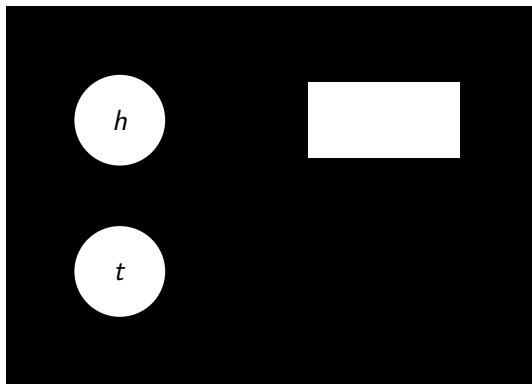
Black box machines



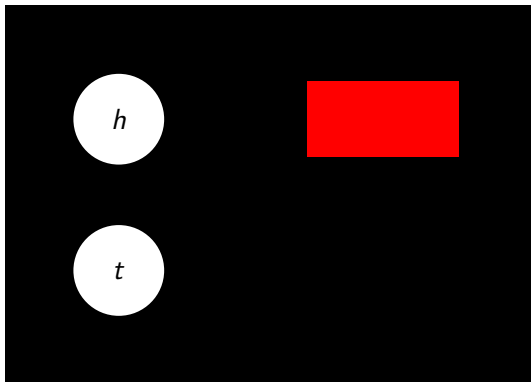
Black box machines



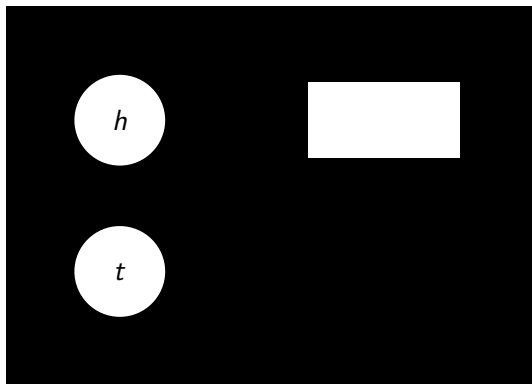
Black box machines



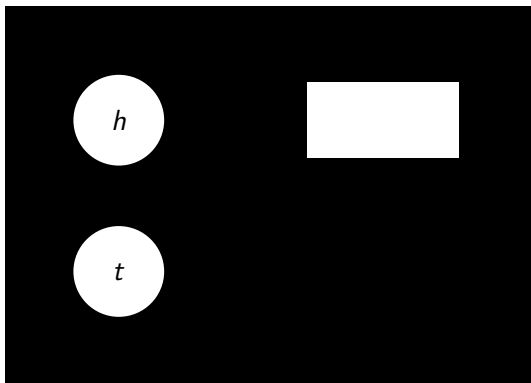
Black box machines



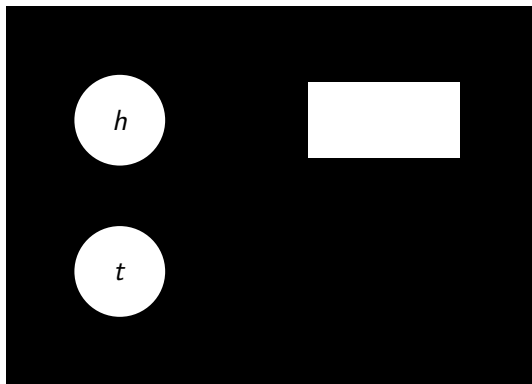
Black box machines



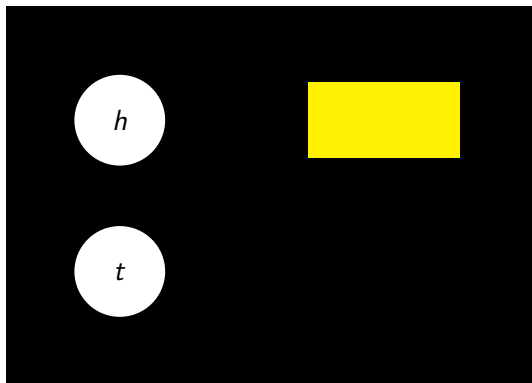
Black box machines



Black box machines



Black box machines



Black box machines

C a set of colors

$$h : S \rightarrow C, \quad t : S \rightarrow S$$

Black box machines

C a set of colors

$$h : S \rightarrow C, \quad t : S \rightarrow S \quad \Leftrightarrow \quad \langle h, t \rangle : S \rightarrow C \times S$$

Black box machines

C a set of colors

$$h : S \rightarrow C, \quad t : S \rightarrow S \quad \iff \quad \langle h, t \rangle : S \rightarrow C \times S$$

Black box machines are $C \times \text{id}$ -coalgebras

Deterministic automata

A an alphabet, S a set of states.

A subset $F \subseteq S$, a function $S \times A \rightarrow S$.

Deterministic automata

A an alphabet, S a set of states.

A **function** $\alpha : S \rightarrow \{0, 1\}$, a function $S \times A \rightarrow S$.

Deterministic automata

A an alphabet, S a set of states.

A function $\alpha : S \rightarrow \{0, 1\}$, a **function** $\sigma : S \rightarrow S^A$.

Deterministic automata

A an alphabet, S a set of states.

A function $\langle \alpha, \sigma \rangle : S \rightarrow \{0, 1\} \times S^A$.

Deterministic automata

A an alphabet, S a set of states.

A function $\langle \alpha, \sigma \rangle : S \rightarrow \{0, 1\} \times S^A$.

A deterministic automaton is a $2 \times \text{id}^A$ -coalgebra

Deterministic automata

A an alphabet, S a set of states.

A function $\langle \alpha, \sigma \rangle : S \rightarrow \{0, 1\} \times P(S)^A$

A nondeterministic automaton is a $2 \times P(\text{id})^A$ -coalgebra

Final coalgebras and corecursion

A C -stream is a function $s : \mathbb{N} \rightarrow C$.

A C -stream is a function $s : \mathbb{N} \rightarrow C$.

Write `Streams` for the set of C -streams. We get some natural functions:

A C -stream is a function $s : \mathbb{N} \rightarrow C$.

Write `Streams` for the set of C -streams. We get some natural functions:

- `head` : `Streams` $\rightarrow C$ given by $(c_0, c_1, c_2, \dots) \mapsto c_0$;

A C -stream is a function $s : \mathbb{N} \rightarrow C$.

Write `Streams` for the set of C -streams. We get some natural functions:

- `head` : `Streams` $\rightarrow C$ given by $(c_0, c_1, c_2, \dots) \mapsto c_0$;
- `tail` : `Streams` \rightarrow `Streams` given by $(c_0, c_1, c_2, \dots) \mapsto (c_1, c_2, c_3, \dots)$.

A C -stream is a function $s : \mathbb{N} \rightarrow C$.

Write `Streams` for the set of C -streams. We get some natural functions:

- `head` : `Streams` $\rightarrow C$ given by $(c_0, c_1, c_2, \dots) \mapsto c_0$;
- `tail` : `Streams` \rightarrow `Streams` given by $(c_0, c_1, c_2, \dots) \mapsto (c_1, c_2, c_3, \dots)$.

So `Streams` is a BBM.

A C -stream is a function $s : \mathbb{N} \rightarrow C$.

Write Streams for the set of C -streams. We get some natural functions:

- $\text{head} : \text{Streams} \rightarrow C$ given by $(c_0, c_1, c_2, \dots) \mapsto c_0$;
- $\text{tail} : \text{Streams} \rightarrow \text{Streams}$ given by $(c_0, c_1, c_2, \dots) \mapsto (c_1, c_2, c_3, \dots)$.

So Streams is a BBM.

Proposition

For any BBM $\sigma = \langle h, t \rangle : S \rightarrow C \times S$, there is a *unique* coalgebra morphism $\text{beh}_\sigma : S \rightarrow \text{Streams}$.

Streams

A C -stream is a function $s : \mathbb{N} \rightarrow C$.

Write Streams for the set of C -streams. We get some natural functions:

- $\text{head} : \text{Streams} \rightarrow C$ given by $(c_0, c_1, c_2, \dots) \mapsto c_0$;
- $\text{tail} : \text{Streams} \rightarrow \text{Streams}$ given by $(c_0, c_1, c_2, \dots) \mapsto (c_1, c_2, c_3, \dots)$.

So Streams is a BBM.

Proposition

For any BBM $\sigma = \langle h, t \rangle : S \rightarrow C \times S$, there is a *unique* coalgebra morphism $\text{beh}_\sigma : S \rightarrow \text{Streams}$.

Proof.

The only possible map is $\text{beh}_\sigma(s) = (h(s), ht(s), htt(s), htts(s), \dots)$. \square

A C -stream is a function $s : \mathbb{N} \rightarrow C$.

Write `Streams` for the set of C -streams. We get some natural functions:

- $\text{head} : \text{Streams} \rightarrow C$ given by $(c_0, c_1, c_2, \dots) \mapsto c_0$;
- $\text{tail} : \text{Streams} \rightarrow \text{Streams}$ given by $(c_0, c_1, c_2, \dots) \mapsto (c_1, c_2, c_3, \dots)$.

So `Streams` is a BBM.

Proposition

`Streams` is final in the category of $C \times \text{id}$ -coalgebras

Proof.

The only possible map is $\text{beh}_\sigma(s) = (h(s), ht(s), htt(s), htts(s), \dots)$. \square

Languages

An A -language is a set of words over A .

An A -language is a set of words over A .

Write Langs for the set of A -languages. We get some natural functions:

An A -language is a set of words over A .

Write Langs for the set of A -languages. We get some natural functions:

- $\text{acc} : \text{Langs} \rightarrow \{0, 1\}$ given by $\text{acc}(L) = 1$ iff $\epsilon \in L$;

An A -language is a set of words over A .

Write Langs for the set of A -languages. We get some natural functions:

- $\text{acc} : \text{Langs} \rightarrow \{0, 1\}$ given by $\text{acc}(L) = 1$ iff $\epsilon \in L$;
- $\delta : \text{Langs} \rightarrow \text{Langs}^A$ given by $\delta(L, a) := \{w \mid aw \in L\}$.

An A -language is a set of words over A .

Write Langs for the set of A -languages. We get some natural functions:

- $\text{acc} : \text{Langs} \rightarrow \{0, 1\}$ given by $\text{acc}(L) = 1$ iff $\epsilon \in L$;
- $\delta : \text{Langs} \rightarrow \text{Langs}^A$ given by $\delta(L, a) := \{w \mid aw \in L\}$.

So Langs is a deterministic automaton.

An A -language is a set of words over A .

Write Langs for the set of A -languages. We get some natural functions:

- $\text{acc} : \text{Langs} \rightarrow \{0, 1\}$ given by $\text{acc}(L) = 1$ iff $\epsilon \in L$;
- $\delta : \text{Langs} \rightarrow \text{Langs}^A$ given by $\delta(L, a) := \{w \mid aw \in L\}$.

So Langs is a deterministic automaton.

Proposition

Langs is the final $2 \times \text{id}^A$ -coalgebra.

```
interleave :: (Stream a, Stream a) -> Stream a
head $ interleave (s0, s1) = head s0
tail $ interleave (s0, s1) = interleave (s1, tail s0)
```

```
interleave :: (Stream a, Stream a) -> Stream a
head $ interleave (s0, s1) = head s0
tail $ interleave (s0, s1) = interleave (s1, tail s0)
```

Why does this work?*

```
interleave :: (Stream a, Stream a) -> Stream a
head $ interleave (s0, s1) = head s0
tail $ interleave (s0, s1) = interleave (s1, tail s0)
```

Why does this work?*

Define $\langle h, t \rangle : \text{Streams} \times \text{Streams} \rightarrow C \times (\text{Streams} \times \text{Streams})$ as

- $h(s_0, s_1) = \text{head}(s_0)$;
- $t(s_0, s_1) = (s_1, \text{tail}(s_0))$.

```
interleave :: (Stream a, Stream a) -> Stream a
head $ interleave (s0, s1) = head s0
tail $ interleave (s0, s1) = interleave (s1, tail s0)
```

Why does this work?*

Define $\langle h, t \rangle : \text{Streams} \times \text{Streams} \rightarrow C \times (\text{Streams} \times \text{Streams})$ as

- $h(s_0, s_1) = \text{head}(s_0)$;
- $t(s_0, s_1) = (s_1, \text{tail}(s_0))$.

Then there is a unique coalgebra morphism

$$\text{interleave} : \text{Streams} \times \text{Streams} \rightarrow \text{Streams}.$$

Behavioral equivalence and bisimulation

Behavioral equivalence

$(S, \sigma), s \simeq (S', \sigma'), s'$ iff there is a cospan

$$\begin{array}{ccc} & (S', \sigma') & \\ & \downarrow g & \\ (S, \sigma) & \xrightarrow{f} & (Z, \zeta) \end{array}$$

with $f(s) = g(s')$.

Behavioral equivalence

$(S, \sigma), s \simeq (S', \sigma'), s'$ iff there is a cospan

$$\begin{array}{ccc} & (S', \sigma') & \\ & \downarrow g & \\ (S, \sigma) & \xrightarrow{f} & (Z, \zeta) \end{array}$$

with $f(s) = g(s')$.

Behavioral equivalence is transitive via pushouts.

Spans

$$\begin{array}{ccc} (R, \rho) & \xrightarrow{\pi_2} & (S', \sigma') \\ \downarrow \pi_1 & & \\ (S, \sigma) & & \end{array}$$

$$\begin{array}{ccc} (R, \rho) & \xrightarrow{\pi_2} & (S', \sigma') \\ \downarrow \pi_1 & & \\ (S, \sigma) & & \end{array}$$

$(S, \sigma), s \leftrightarrow (S', \sigma'), s'$ if and only if there is a span (as above) and a $p \in R$ with $\pi_1(p) = s$ and $\pi_2(p) = s'$.

$$\begin{array}{ccc} (R, \rho) & \xrightarrow{\pi_2} & (S', \sigma') \\ \downarrow \pi_1 & & \\ (S, \sigma) & & \end{array}$$

$(S, \sigma), s \leftrightarrow (S', \sigma'), s'$ if and only if there is a span (as above) and a $p \in R$ with $\pi_1(p) = s$ and $\pi_2(p) = s'$.

By taking pushouts, we have

$$\begin{array}{ccc} (R, \rho) & \xrightarrow{\pi_2} & (S', \sigma') \\ \downarrow \pi_1 & & \\ (S, \sigma) & & \end{array}$$

$(S, \sigma), s \leftrightarrow (S', \sigma'), s'$ if and only if there is a span (as above) and a $p \in R$ with $\pi_1(p) = s$ and $\pi_2(p) = s'$.

By taking pushouts, we have

$$s \leftrightarrow s' \implies s \simeq s'$$

Take $R \subseteq S \times S'$.

Take $R \subseteq S \times S'$.

$$\begin{array}{ccccc} S & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & S' \\ \downarrow \sigma & & \searrow \langle \sigma \circ \pi_1, \sigma' \circ \pi_2 \rangle & & \downarrow \sigma' \\ BS & \xleftarrow{\pi_1} & BS \times BS' & \xrightarrow{\pi_2} & BS' \end{array}$$

Take $R \subseteq S \times S'$.

$$\begin{array}{ccccc}
 S & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & S' \\
 \downarrow \sigma & & \downarrow \langle \sigma \circ \pi_1, \sigma' \circ \pi_2 \rangle & & \downarrow \sigma' \\
 BS & \xleftarrow{\pi_1} & BS \times BS' & \xrightarrow{\pi_2} & BS'
 \end{array}$$

BR is a relation between R and $BS \times BS'$.
 $\langle B\pi_1, B\pi_2 \rangle$ is a relation between R and $BS \times BS'$.

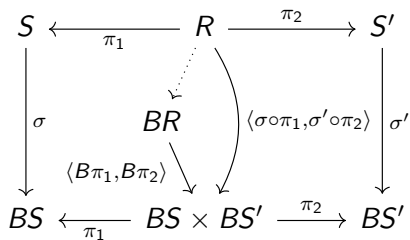
Take $R \subseteq S \times S'$.

$$\begin{array}{ccccc}
 S & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & S' \\
 \downarrow \sigma & & \downarrow \langle B\pi_1, B\pi_2 \rangle & \searrow \langle \sigma \circ \pi_1, \sigma' \circ \pi_2 \rangle & \downarrow \sigma' \\
 BS & \xleftarrow{\pi_1} & BS \times BS' & \xrightarrow{\pi_2} & BS'
 \end{array}$$

The diagram illustrates the relationship between a relation R and its image BR under a mapping B . The top row shows the relation R as a subset of the Cartesian product $S \times S'$, with projection maps π_1 and π_2 . The bottom row shows the image BR as a subset of $BS \times BS'$, with projection maps π_1 and π_2 . The mapping B is represented by the arrows $\langle B\pi_1, B\pi_2 \rangle$ and $\langle \sigma \circ \pi_1, \sigma' \circ \pi_2 \rangle$. The vertical arrows σ and σ' represent the mappings from S to BS and from S' to BS' respectively. A dashed arrow points from R to BR , indicating the image of R under B .

Relations

Take $R \subseteq S \times S'$.



We get a span if

$$\text{im}\langle \sigma \circ \pi_1, \sigma' \circ \pi_2 \rangle \subseteq \text{im}\langle B\pi_1, B\pi_2 \rangle$$

Relations

Take $R \subseteq S \times S'$.

$$\begin{array}{ccccc} S & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & S' \\ \downarrow \sigma & & \downarrow \langle B\pi_1, B\pi_2 \rangle & \searrow \langle \sigma \circ \pi_1, \sigma' \circ \pi_2 \rangle & \downarrow \sigma' \\ BS & \xleftarrow{\pi_1} & BS \times BS' & \xrightarrow{\pi_2} & BS' \end{array}$$

The diagram illustrates a commutative square of relations. The top row consists of sets S , R , and S' with projection maps $\pi_1: R \rightarrow S$ and $\pi_2: R \rightarrow S'$. The bottom row consists of sets BS , $BS \times BS'$, and BS' with projection maps $\pi_1: BS \times BS' \rightarrow BS$ and $\pi_2: BS \times BS' \rightarrow BS'$. A relation $\sigma: S \rightarrow BS$ is shown as a downward arrow from S to BS . A relation $\sigma': S' \rightarrow BS'$ is shown as a downward arrow from S' to BS' . A relation BR is shown as a downward arrow from R to $BS \times BS'$. A curved arrow labeled $\langle \sigma \circ \pi_1, \sigma' \circ \pi_2 \rangle$ points from R to $BS \times BS'$. A straight arrow labeled $\langle B\pi_1, B\pi_2 \rangle$ also points from R to $BS \times BS'$. A dashed arrow points from R to BR .

We get a span if

$$(s, s') \in R \implies \exists p \in BR : B\pi_1(p) = \sigma(s), B\pi_2(p) = \sigma'(s')$$

Let B be a functor. For a relation $R : X \multimap Y$, define

$$\overline{BR} = \{(\alpha, \beta) \in BX \times BY \mid \exists p \in BR : \alpha = B\pi_1(p), \beta = B\pi_2(p)\}$$

Let B be a functor. For a relation $R : X \multimap Y$, define

$$\overline{BR} = \{(\alpha, \beta) \in BX \times BY \mid \exists p \in BR : \alpha = B\pi_1(p), \beta = B\pi_2(p)\}$$

A relation $R \subseteq S \times S'$ is a bisimulation if for all s, s' , we have

$$(s, s') \in R \implies (\sigma(s), \sigma'(s')) \in \overline{BR}$$

Let B be a functor. For a relation $R : X \multimap Y$, define

$$\overline{BR} = \{(\alpha, \beta) \in BX \times BY \mid \exists p \in BR : \alpha = B\pi_1(p), \beta = B\pi_2(p)\}$$

A relation $R \subseteq S \times S'$ is a bisimulation if for all s, s' , we have

$$(s, s') \in R \implies (\sigma(s), \sigma'(s')) \in \overline{BR}$$

Bisimilar states are behaviorally equivalent.

Let B be a functor. For a relation $R : X \multimap Y$, define

$$\overline{BR} = \{(\alpha, \beta) \in BX \times BY \mid \exists p \in BR : \alpha = B\pi_1(p), \beta = B\pi_2(p)\}$$

A relation $R \subseteq S \times S'$ is a bisimulation if for all s, s' , we have

$$(s, s') \in R \implies (\sigma(s), \sigma'(s')) \in \overline{BR}$$

Bisimilar states are behaviorally equivalent. But not always the other way around!

Modal logic

A Kripke model is a $P(\text{Prop}) \times P(-)$ -coalgebra.

Semantics of modalities

A Kripke model is a $P(\text{Prop}) \times P(-)$ -coalgebra.

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \exists v \in \sigma(w) : \mathfrak{M}, v \Vdash \phi$$

Semantics of modalities

A Kripke model is a $P(\text{Prop}) \times P(-)$ -coalgebra.

$\mathfrak{M}, w \Vdash \diamond\phi$ iff $\sigma(w)$ has property $\lambda_{\diamond}[\phi]$

Semantics of modalities

A Kripke model is a $P(\text{Prop}) \times P(-)$ -coalgebra.

$\mathfrak{M}, w \Vdash \Diamond\phi$ iff $\sigma(w)$ has property $\lambda_{\Diamond}(\llbracket\phi\rrbracket)$

Semantics of modalities

A Kripke model is a $P(\text{Prop}) \times P(-)$ -coalgebra.

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \sigma(w) \in \lambda_{\Diamond}(\llbracket\phi\rrbracket)$$

Semantics of modalities

A Kripke model is a $P(\text{Prop}) \times P(-)$ -coalgebra.

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \sigma(w) \in \lambda_{\Diamond}(\llbracket\phi\rrbracket)$$

where $\lambda_{\Diamond} : P \rightarrow PB$ is given by

$$\lambda_{\Diamond}(U) := \{(A, V) \mid U \cap V \neq \emptyset\}$$

Semantics of modalities

A Kripke model is a $P(\text{Prop}) \times P(-)$ -coalgebra.

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \sigma(w) \in \lambda_{\Diamond}(\llbracket\phi\rrbracket)$$

where $\lambda_{\Diamond} : \check{P} \rightarrow \check{P}B$ is given by

$$\lambda_{\Diamond}(U) := \{(A, V) \mid U \cap V \neq \emptyset\}$$

Semantics of modalities

A Kripke model is a $P(\text{Prop}) \times P(-)$ -coalgebra.

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \sigma(w) \in \lambda_{\Diamond}(\llbracket\phi\rrbracket)$$

where $\lambda_{\Diamond} : \check{P} \rightarrow \check{P}B$ is given by

$$\lambda_{\Diamond}(U) := \{(A, V) \mid U \cap V \neq \emptyset\}$$

Proposition letters are nullary modalities!

Semantics of modalities

A Kripke model is a $P(\text{Prop}) \times P(-)$ -coalgebra.

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \sigma(w) \in \lambda_{\Diamond}(\llbracket\phi\rrbracket)$$

where $\lambda_{\Diamond} : \check{P} \rightarrow \check{P}B$ is given by

$$\lambda_{\Diamond}(U) := \{(A, V) \mid U \cap V \neq \emptyset\}$$

Proposition letters are nullary modalities! $\lambda_p : (\check{P})^0 \rightarrow \check{P}B$ is given by

$$\lambda_p(*) := \{(A, V) \mid p \in A\}$$

Semantics of modalities

A Kripke model is a $P(\text{Prop}) \times P(-)$ -coalgebra.

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \sigma(w) \in \lambda_{\Diamond}(\llbracket\phi\rrbracket)$$

where $\lambda_{\Diamond} : \check{P} \rightarrow \check{P}B$ is given by

$$\lambda_{\Diamond}(U) := \{(A, V) \mid U \cap V \neq \emptyset\}$$

Proposition letters are nullary modalities! $\lambda_p : (\check{P})^0 \rightarrow \check{P}B$ is given by

$$\lambda_p(*) := \{(A, V) \mid p \in A\}$$

Then

$$\mathfrak{M}, w \Vdash p \text{ iff } \sigma(w) \in \lambda_p(*)$$

Predicate liftings

Let $B : \text{Set} \rightarrow \text{Set}$ be a behavior functor. An n -ary predicate lifting is a natural transformation

$$\lambda : (\check{P})^n \rightarrow \check{P}B$$

Examples:

Predicate liftings

Let $B : \text{Set} \rightarrow \text{Set}$ be a behavior functor. An n -ary predicate lifting is a natural transformation

$$\lambda : (\check{P})^n \rightarrow \check{P}B$$

Examples:

- Modal logic: we have a \square given by

$$\lambda_{\square} : U \mapsto \{(A, V) \mid V \subseteq U\}$$

Predicate liftings

Let $B : \text{Set} \rightarrow \text{Set}$ be a behavior functor. An n -ary predicate lifting is a natural transformation

$$\lambda : (\check{P})^n \rightarrow \check{P}B$$

Examples:

- Modal logic: we have a \Box given by

$$\lambda_{\Box} : U \mapsto \{(A, V) \mid V \subseteq U\}$$

- (Labeled) binary trees: functor is $X \mapsto P(\text{Prop}) \times X \times X$. We get a binary modality $[\leftrightarrow]$ given by

$$\lambda_{\leftrightarrow}(U, V) = \{(A, x, y) \mid x \in U \text{ iff } y \in V\}$$

Coalgebraic modal logic

$$\mathcal{L} ::= \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \langle \lambda \rangle (\phi_1, \dots, \phi_n)$$

where λ is an n -ary predicate lifting.

Coalgebraic modal logic

$$\mathcal{L} ::= \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \langle\lambda\rangle(\phi_1, \dots, \phi_n)$$

where λ is an n -ary predicate lifting.

For modalities, the semantics is given by

$$\llbracket \langle\lambda\rangle(\phi_1, \dots, \phi_n) \rrbracket_\sigma := \check{P}\sigma \circ \lambda(\llbracket \phi_1 \rrbracket_\sigma, \dots, \llbracket \phi_n \rrbracket_\sigma)$$

Coalgebraic modal logic

$$\mathcal{L} ::= \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \langle\lambda\rangle(\phi_1, \dots, \phi_n)$$

where λ is an n -ary predicate lifting.

For modalities, the semantics is given by

$$\llbracket \langle\lambda\rangle(\phi_1, \dots, \phi_n) \rrbracket_\sigma := \check{P}\sigma \circ \lambda(\llbracket \phi_1 \rrbracket_\sigma, \dots, \llbracket \phi_n \rrbracket_\sigma)$$

Proposition

If $f : (S, \sigma) \rightarrow (S', \sigma')$ is a morphism, then for all $s \in S$ and all formulas ϕ , we have

$$s \Vdash \phi \text{ iff } f(s) \Vdash \phi$$

Coalgebraic modal logic

$$\mathcal{L} ::= \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \langle\lambda\rangle(\phi_1, \dots, \phi_n)$$

where λ is an n -ary predicate lifting.

For modalities, the semantics is given by

$$\llbracket \langle\lambda\rangle(\phi_1, \dots, \phi_n) \rrbracket_\sigma := \check{P}_\sigma \circ \lambda(\llbracket \phi_1 \rrbracket_\sigma, \dots, \llbracket \phi_n \rrbracket_\sigma)$$

Proposition

If $f : (S, \sigma) \rightarrow (S', \sigma')$ is a morphism, then for all $s \in S$ and all formulas ϕ , we have

$$s \Vdash \phi \text{ iff } f(s) \Vdash \phi$$

Corollary

If $s \simeq s'$, then s and s' are logically equivalent.

Regular languages

Recall: $BX = 2 \times X^A$.

Regular languages

Recall: $BX = 2 \times X^A$.

- Nullary lifting:

$$\lambda_{\vee}(*):= \{(i, u) \mid i = 1\}$$

Regular languages

Recall: $BX = 2 \times X^A$.

- Nullary lifting:

$$\lambda_{\checkmark} := \{(i, u) \mid i = 1\}$$

- For $a \in A$, a unary lifting

$$\lambda_a(U) := \{(i, u) \mid u(a) \in U\}$$

We get a translation $m : A^* \rightarrow \mathcal{L}$ by

$$\epsilon \mapsto \langle \checkmark \rangle, \quad aw \mapsto \langle \lambda_a \rangle(m(w))$$

Regular languages

Recall: $BX = 2 \times X^A$.

- Nullary lifting:

$$\lambda_{\checkmark}(*):= \{(i, u) \mid i = 1\}$$

- For $a \in A$, a unary lifting

$$\lambda_a(U) := \{(i, u) \mid u(a) \in U\}$$


We get a translation $m : A^* \rightarrow \mathcal{L}$ by


$$\epsilon \mapsto \langle \checkmark \rangle, \quad aw \mapsto \langle \lambda_a \rangle(m(w))$$


Proposition

Let $\sigma : S \rightarrow BS$ be a DFA. For $s \in S$, we have that s accepts w if and only if $s \Vdash m(w)$.

Thank you for listening!

 Bart Jacobs and Jan Rutten.
A tutorial on (co)algebras and (co)induction.
EATCS Bulletin, 62:62–222, 1997.

 D. Pattinson.
An introduction to the theory of coalgebras.
2003.

 Jan Rutten.
Universal coalgebra: A theory of systems.
Theoretical Computer Science, 249:3–80, 10 2000.

```
head :: Stream a -> a
```

```
tail :: Stream a -> Stream a
```

```
unfold :: (c -> (a,c)) -> c -> Stream a
```

coinductive type

Contents

[1. Idea](#)

[2. Properties](#)

[Categorical semantics](#)

[Coinductive type formation in homotopy type theory](#)

[3. Related concepts](#)

[4. References](#)

3. Related concepts

- coinduction, corecursion
- coinductive definition
- inductive type

coinductive definition

Contents

[1. Idea](#)

[2. Definition](#)

[3. Related concepts](#)

1. Idea

A *coinductive definition* is a [definition](#) by [coinduction](#).

2. Definition

See at [coinductive type](#).



My conclusion

Everything is a coinductive definition