Introduction to Universal Coalgebra

Ezra Schoen

June 29, 2021

Contents

Basic definitions and examples

Definition

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

Definition

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

$$\sigma : S \to BS$$

Definition

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

$$\sigma : S \to BS$$

Morphisms

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

Definition

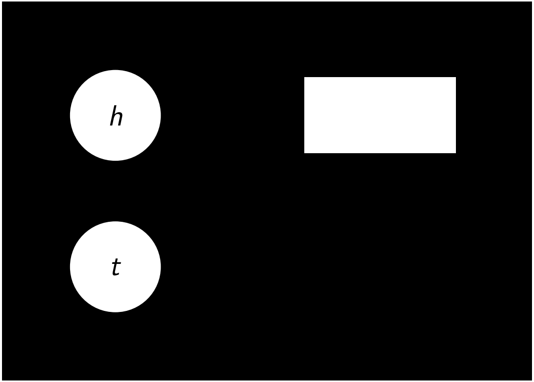
B : Set $\to$ Set a functor, S a set of states

$$\sigma : S \to BS$$

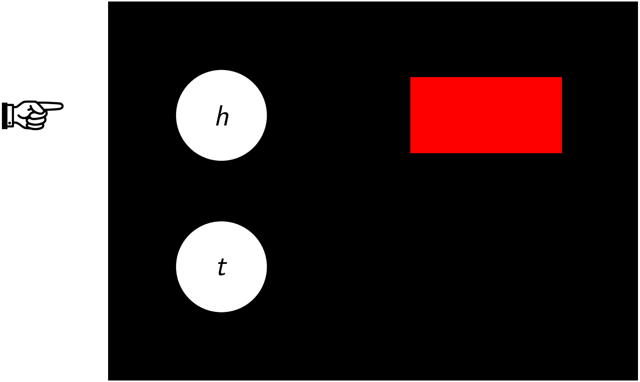
Morphisms

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

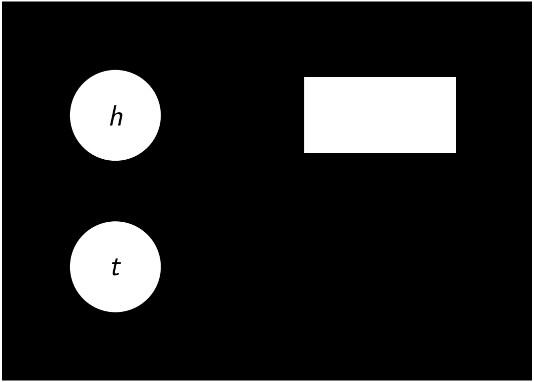
Black box machines



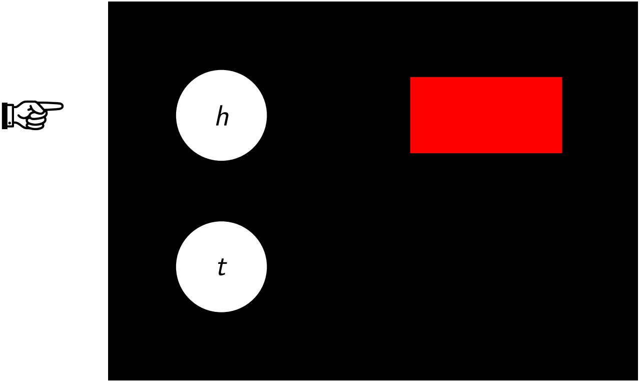
Black box machines



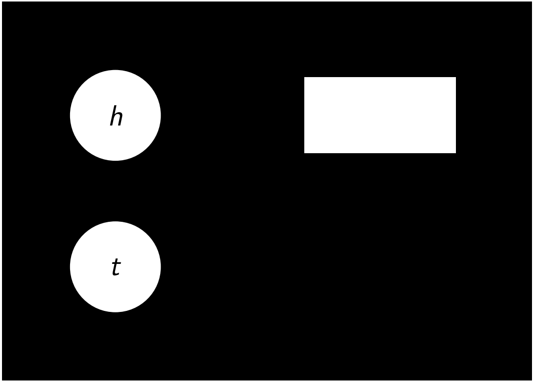
Black box machines



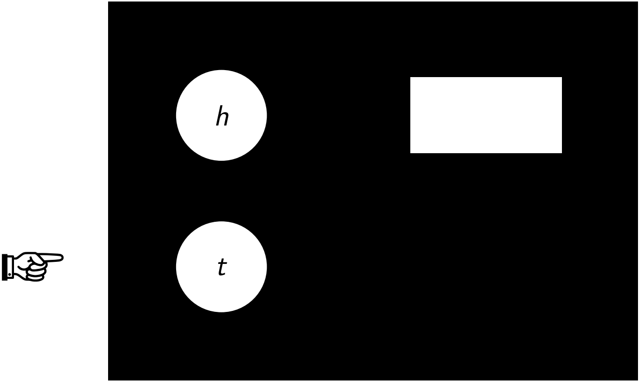
Black box machines



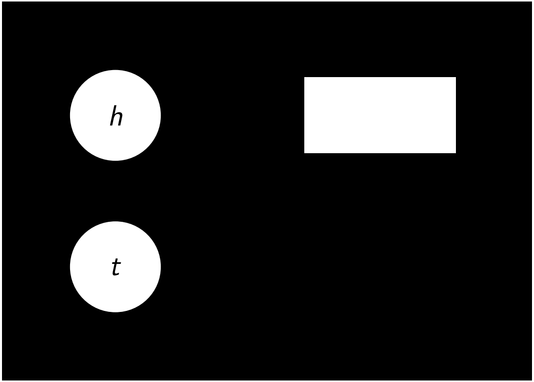
Black box machines



Black box machines

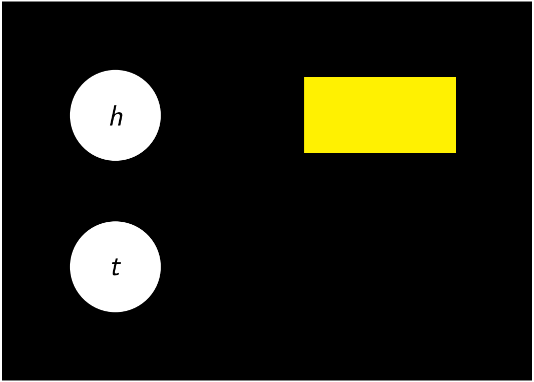


Black box machines



Black box machines

☞



Black box machines

C a set of colors

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

Black box machines

C a set of colors

$$h : S \to C, \quad t : S \to S \qquad \leftrightsquigarrow \qquad \langle h, t \rangle : S \to C \times S$$

Black box machines

C a set of colors

$$h : S \to C, \quad t : S \to S \qquad \rightsquigarrow \qquad \langle h, t \rangle : S \to C \times S$$

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

Deterministic automata

A an alphabet, S a set of states.

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

Deterministic automata

A an alphabet, S a set of states.

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

Deterministic automata

A an alphabet, S a set of states.

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

Deterministic automata

A an alphabet, S a set of states.

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

Deterministic automata

A an alphabet, S a set of states.

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

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

Deterministic automata

A an alphabet, S a set of states.

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

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

Final coalgebras and corecursion

Streams

Streams

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

Streams

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

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

Streams

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

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

- head : Streams $\to C$ given by $(c_0, c_1, c_2, \ldots) \mapsto c_0$;

Streams

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

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

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

Streams

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

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

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

So Streams is a BBM.

Streams

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

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

- head : Streams $\to C$ given by $(c_0, c_1, c_2, \dots) \mapsto c_0$;
- tail : Streams $\to$ 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 $\sigma = \langle h, t \rangle : S \to C \times S$, there is a *unique* coalgebra morphism $\mathrm{beh}_\sigma : S \to$ Streams.

Streams

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

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

- head : Streams $\to C$ given by $(c_0, c_1, c_2, \dots) \mapsto c_0$;
- tail : Streams $\to$ 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 $\sigma = \langle h, t \rangle : S \to C \times S$, there is a *unique* coalgebra morphism $\mathrm{beh}_\sigma : S \to$ Streams.

Proof.

The only possible map is $\mathrm{beh}_\sigma(s) = (h(s), ht(s), htt(s), httt(s), \dots)$. □

Streams

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

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

- head : Streams $\to C$ given by $(c_0, c_1, c_2, \dots) \mapsto c_0$;
- tail : Streams $\to$ 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 $C \times$ id-coalgebras

Proof.

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

Languages

Languages

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

Languages

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

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

Languages

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

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

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

Languages

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

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

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

Languages

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

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

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

So Langs is a deterministic automaton.

Languages

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

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

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

So Langs is a deterministic automaton.

Proposition

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

Corecursion

Corecursion

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

Corecursion

```
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?*

Corecursion

```
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 $\langle h, t\rangle$: Streams $\times$ Streams $\to C \times$ (Streams $\times$ Streams) as

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

Corecursion

```
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 $\langle h, t\rangle$: Streams $\times$ Streams $\to C \times$ (Streams $\times$ 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

$$\texttt{interleave} : \text{Streams} \times \text{Streams} \to \text{Streams}.$$

Behavioral equivalence and bisimulation

Behavioral equivalence

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

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

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

Behavioral equivalence

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

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

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

Behavioral equivalence is transitive via pushouts.

Spans

Spans

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

Spans

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

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

Spans

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

$(S, \sigma), s \leftrightarroweq (S', \sigma'), s'$ 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

Spans

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

$(S,\sigma), s \leftrightarroweq (S',\sigma'), s'$ 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

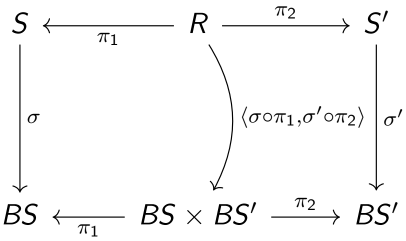
$$s \leftrightarroweq s' \implies s \simeq s'$$

Relations

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

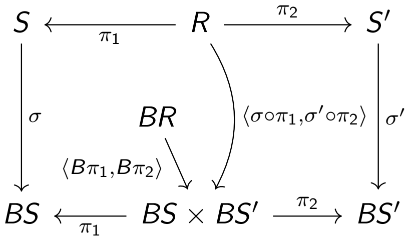
Relations

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



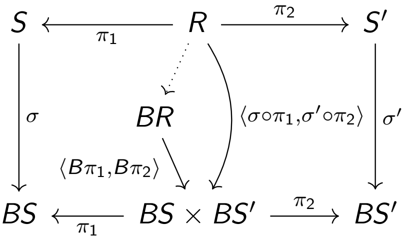
Relations

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



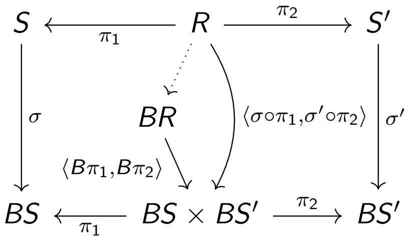
Relations

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



Relations

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

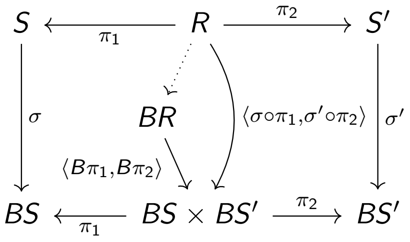


We get a span if

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

Relations

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



We get a span if

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

Bisimulation

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

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

Bisimulation

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

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

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

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

Bisimulation

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

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

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

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

Bisimilar states are behaviorally equivalent.

Bisimulation

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

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

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

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

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

Modal logic

Semantics of modalities

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.

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

Semantics of modalities

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

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

Semantics of modalities

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

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \sigma(w) \in \lambda_{\Diamond}([\![\phi]\!])$$

Semantics of modalities

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

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \sigma(w) \in \lambda_\Diamond([\![\phi]\!])$$

where $\lambda_\Diamond : P \to PB$ is given by

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

Semantics of modalities

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

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \sigma(w) \in \lambda_{\Diamond}([\![\phi]\!])$$

where $\lambda_{\Diamond} : \breve{P} \to \breve{P}B$ is given by

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

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 $\lambda_\Diamond : \breve{P} \to \breve{P}B$ is given by

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

Proposition letters are nullary modalities!

Semantics of modalities

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

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \sigma(w) \in \lambda_\Diamond([\![\phi]\!])$$

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

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

Proposition letters are nullary modalities! $\lambda_p : (\check{P})^0 \to \check{P}B$ 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.

$$\mathfrak{M}, w \Vdash \Diamond\phi \text{ iff } \sigma(w) \in \lambda_\Diamond([\![\phi]\!])$$

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

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

Proposition letters are nullary modalities! $\lambda_p : (\check{P})^0 \to \check{P}B$ 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 : Set $\to$ Set be a behavior functor. An *n-ary predicate lifting* is a natural transformation

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

Examples:

Predicate liftings

Let B : Set $\to$ Set be a behavior functor. An *n-ary predicate lifting* is a natural transformation

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

Examples:

- Modal logic: we have a $\Box$ given by

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

Predicate liftings

Let B : Set $\to$ Set be a behavior functor. An *n-ary predicate lifting* is a natural transformation

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

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

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

where λ is an n-ary predicate lifting.

Coalgebraic modal logic

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

where λ is an n-ary predicate lifting.

For modalities, the semantics is given by

$$[\![\langle\lambda\rangle(\phi_1, \ldots, \phi_n)]\!]_\sigma := \breve{P}\sigma \circ \lambda([\![\phi_1]\!]_\sigma, \ldots, [\![\phi_n]\!]_\sigma)$$

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

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

Proposition

If $f : (S, \sigma) \to (S', \sigma')$ 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

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

where λ is an n-ary predicate lifting.

For modalities, the semantics is given by

$$[\![\langle\lambda\rangle(\phi_1, \ldots, \phi_n)]\!]_\sigma := \check{P}\sigma \circ \lambda([\![\phi_1]\!]_\sigma, \ldots, [\![\phi_n]\!]_\sigma)$$

Proposition

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

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

Corrollary

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:

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

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 $m : A^* \to \mathcal{L}$ 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 $m : A^* \to \mathcal{L}$ by

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

Proposition

Let $\sigma : S \to BS$ 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!

References

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.

Data.Stream

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

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

Coinductive definitions

nLab

Home Page All Pages

coinductive type

Contents

Coinductive definitions

3. Related concepts

- coinduction, corecursion
- coinductive definition
- inductive type

Coinductive definitions

nLab

Home Page

coinductive definition

Contents

1. Idea

A *coinductive definition* is a definition by coinduction.

2. Definition

See at *coinductive type*.

Coinductive definitions

My conclusion

Everything is a coinductive definition