Teoria Kategorii

Weronika Jakimowicz

Lato 2024/25

Spis treści

03.03.2025 Funktory dołączone

1. Motywacja abstrakcyjnego nonsensu

Niech V będzie przestrzenią wektorową nad ciałem k, a B wybraną jej bazą. Dowolne odwzorowanie $B \to V$ możemy rozszerzyć na odwzorowanie liniowe $k[B] = V \to V$. To znaczy, mamy izomorfizm zbiorów

$$Hom(B, V) \cong Hom(V, V).$$

W języku abstrakcyjnego nonsensu możemy zdefiniować dwa funktory,

$$Set(-, U(-)) : Set^{op} \times Vect_k^{fin} \to Set$$

$$Vect_k(k[-], -) : Set^{op} \times Vect_k^{fin} \to Set,$$

gdzie $U : Vect_k^{fin} \to Set$ to funktor zapominający strukturę przestrzeni wektorowej, między którymi istnieją naturalne izomorfizmy.

$$Set(-, U(-)) \cong Vect_k(k[-], -)$$

Definicja 1.12: funktory dołączone

Niech $L : \mathcal{C} \to \mathcal{D}$ oraz $R : \mathcal{D} \to \mathcal{C}$ będą funktorami. Powiemy, że L jest **lewo dołączony** do funktora R, a R **prawo dołączony** do L, jeśli funktory

$$\mathcal{C}(-, R-), \mathcal{D}(L-, -) : \mathcal{C}^{op} \times \mathcal{D} \to Set$$

są naturalnie izomorficzne. Taką parę funktorów dołączonych oznaczamy $L \dashv R$.

2. Dużo przykładów funktorów dołączonych

1. Niech $R : Set_* \to Set$ będzie funktorem z kategorii zbiorów zbazowanych w kategorię zbiorów, który zapomina o punkcie bazowym. Chcemy teraz znaleźć funktor $L : Set \to Set_*$, który będzie do niego lewo dołączony. Niech $L(X) = X \cup \{X\}$ (lub bardziej obrazowo: $X \sqcup \{*\}$), gdzie y_0 poślemy na $\{X\}$, to znaczy doklejamy do X singleton i staje się on punktem wyróżnionym.

   Oba funktory są różnowartościowe na obiektach, więc wystarczy przekonać się, że

   $$Set_*(LX, (Y, y_0)) \cong Set(X, R(Y, y_0))$$

   jest izomorfizmem. Dowolna funkcja $X \to Y$ rozszerza się przez posłanie $\{X\} \mapsto y_0$ na funkcję $(X, \{X\}) \to (Y, y_0)$.

2. Podobna sytuacja ma miejsce, kiedy szukamy lewo dołączony funktor do $R : Ring \to Rng$ między kategorią pierścieni z jedynką, a wszystkimi pierścieniami. Definiujemy funktor

$$L : Rng \to Ring$$

jako doklejenie $\mathbb{Z}$, $L(S) = \mathbb{Z} \oplus S$ z działaniem $(n, s)(n', s') = (nn', ns' + ss' + n's)$, wtedy $(1, 0_S)$ jest jedynka w nowym pierścieniu. Pozostaje przyjrzeć się co się dzieje z morfizmami, skoro

$$Rng(S, RT) \cong Ring(LS, T).$$

Dowolny morfizm $\varphi : S \to RT$ wystarczy, że trzyma element neutralny ze względu na dodawanie i jest addytywny. Możemy go rozszerzyć na morfizm, który całą pierwszą współrzędną $LS = \mathbb{Z} \oplus S$ posyła w $1_T \in T$, a drugą zgodnie z φ. W drugą stronę wystarczy obciąć morfizm do drugiej współrzędnej.

3. Niech $\Delta : Set \to Set \times Set$ będzie funktorem takim, że $\Delta(C) = (C, C)$. Zaczniemy od szukania funktora dołączonego do niego z prawej strony, czyli $R : Set \times Set \to Set$ takiego, że

$$\mathrm{Hom}(X, R(Y, Z)) \cong \mathrm{Hom}(\Delta(X), (Y, Z)).$$

Od razu narzuca się $R(Y, Z) = Y \times Z$, czyli zlepiamy współrzędne $\Delta(X)$ w jedną. Przypomnijmy, że iloczyn kartezjański w kategorii zbiorów jest produktem.

Funktor lewo dołączony musi zatem spełniać

$$\mathrm{Hom}(L(X, Y), Z) \cong \mathrm{Hom}((X, Y), \Delta(Z)),$$

czyli dowolną funkcję $(X, Y) \to (Z, Z)$ musimy umieć zapisać jako funkcję z pojedynczego zbioru, którym będzie suma rozłączna $L(X, Y) = X \sqcup Y$, czyli koprodukt w kategorii zbiorów.

Historia funktora Δ uogalnia się na dowolną kategorię, w której są produkty i koprodukty:

$$\text{koprodukt} \dashv \Delta \dashv \text{produkt}$$

4. Ustalmy zbiór $Y \in Set_0$ i niech $R : Set \to Set$ będzie funktorem, który zbiorowi X przypisuje wszystkie funkcje z Y w ten zbiór, $R(X) = Set(Y, X)$. Chcemy znaleźć funktor lewo dołączony $L : Set \to Set$ do R. Patrzymy na morfizmy i mamy

$$Set(L(X), Z) \cong Set(X, \underbrace{Set(Y, Z)}_{R(Z)})$$

zbiór po prawej to funkcje z X w funkcje z Y w Z. Można to przedstawić jako funkcje $X \times Y \to Z$, czyli $LX = X \times Y$.

Technika tłumaczenia funkcji o więcej niż jednym argumencie na sekwencję funkcji nazywamy *currying*.

5. Analogicznie jak w poprzednim przykładzie, niech R będzie pierścieniem (przemiennym z jedynką), W R-modułem i R funktorem $R : RMod \to RMod$ takim, że $R(U) = \mathrm{Hom}_R(W, U)$ będzie zbiorem homomorfizmów R-modułów. Funktorem lewo-dołączonym do R będzie wtedy $L(V) = V \otimes W$:

$$RMod(V, \mathrm{Hom}_R(W, U)) \cong RMod(V \otimes W, U).$$

Uwaga: tensor produkt zwykle nie ma funktora lewo do siebie dołączonego.

6. Załóżmy, że kategoria $\mathcal{C}$ ma produkty i ustalmy $X \in \mathcal{C}$. Rozważmy funktor $L : \mathcal{C} \to \mathcal{C}$, $L(Y) = Y \times X$. Jeśli kategoria $\mathcal{C}$ posiada obiekty eksponencjalne, czyli wiemy jak uogólnić na nią przestrzeń funkcji $X \to Y$ (oznaczane Y^X), to funktorem prawo dołączonym do L jest właśnie funktor przypisujący obiektowi Y jego eksponens Y^X,

$$\mathcal{C}(Y, Z^X) \cong \mathcal{C}(Y \times X, Z).$$

Przykładem takiej kategorii są przestrzenie "core-compact".

W ramach kontrprzykładu rozważmy funktor zapominania $U : FinGrp \to FinSet$, i załóżmy, że $L : FinSet \to FinGrp$ jest jego funktorem lewo dołączonym. Niech p będzie taką liczbą pierwszą, że $p > |L(1)|$ (wystarczy, że są względnie pierwsze). Wtedy

$$FinSet(1, U(\mathbb{Z}_p)) \cong FinGrp(L(1), \mathbb{Z}_p)$$

gdzie po lewej zbiór ma $|\mathbb{Z}_p| = p$ różnych funkcji z singletona w zbiór elementów grupy $\mathbb{Z}_p$, a po prawej mamy jedynie trywialny morfizm, bo żaden element $L(1)$ nie ma rzędu podzielnego przez p, czyli nie może przejść w żaden nietrywialny element $\mathbb{Z}_p$.

3. Druga definicja

Definicja 1.13: funktory dołączone (naturalne transformacje)

Rozważmy parę funktorów

$$\mathcal{C} \underset{R}{\overset{L}{\rightleftarrows}} \mathcal{D}.$$

Powiemy, że L jest lewo dołączony do R i na odwrót, jeśli istnieją dwie natrualne transformacje

$$\varepsilon : LR \implies 1_{\mathcal{D}} \quad \eta : 1_{\mathcal{C}} \implies RL$$

takie, że komutują diagramy

η nazywamy *unit*, a ε to *counit*.

17.03.2025 Kategoria Kleislego

A monad is just a monoid in the category of endofunctors, what's the problem?

1. Po co właściwie te monady?

W programowaniu monady są używane do modelowania "robienia czegoś więcej" jako efektu działania funkcji. W OCamlu (autorka notatek dostaje oczopląsu na widok Haskella) jest definiowana jako

```
module type Monad = sig
  type 'a t
  val return : 'a -> 'a t
  val bind : 'a t -> ('a -> 'b t) -> 'b t
end
```

Przykładem namacalnej monady jest tzw. monada Maybe, która opakowuje dane w pudełko, tym samym pozwalając zwracać pudełka puste.

Powiedzmy, że potrzebujemy znaleźć element maksymalny listy, czyli `maxElem : int list -> int`. Co, jeśli nasza lista jest pusta? Możemy opakować zwracaną wartość i zmienić ją w `int option`. Wtedy w wypadku pustej listy zwracamy `None`.

```
let maxElem (x : int list) : int option =
  match x with
  | [] -> None
  | x::xs ->
    match maxElem(xs) with
    | None -> Some x
    | Some y -> Some max(x, y)
```

Pojawia się kolejny problem: zmiana zwracanego typu z `int` na `int option` nie pozwala nam dodawać elementów maksymalnych z różnych list, ani (po napisaniu `minElem`) odjąć od elementu maksymalnego elementu minimalnego. Potrzebujemy więc w elegancki sposób zmienić również operacje arytmetyczne. Zacznijmy od zdefiniowania funkcji potrzebnych w monadzie.

```
type 'a t = a' option

let return (x : int) : int option =
  Some x
```

```
let bind (x : int option) (op : int -> int option) : int option =
  match x with
  | None -> None
  | Some a -> op a
```

Funkcja `return` nie robi nic poza opakowaniem `int` w `int option`, natomiast funkcja `bind` wyjmuje `int` z pudełka i dopiero wtedy nakłada funkcję i pakuje z powrotem do pudła. Dla przykładu napiszemy tylko nową implementację dodawania, która będzie teraz pobierać dwa argumenty typu `int option` i zwracać `int option`.

```
let ( + ) : (x : int option) (y : int option) : int option =
  bind ( x, fun a -> bind(y, fun b -> Some(a+b)) )
```

Możemy teraz odpalić

```
maxElem([1; 4; 45]) + maxElem([44; -10; 9])
```

i na konsoli zobaczymy `Some 69`.

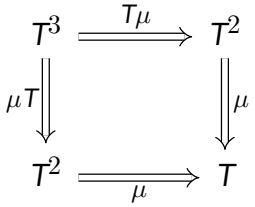
2. Definicja i przykłady monad

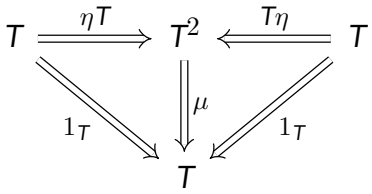
Definicja 1.20: monada

Monada na kategorii $\mathcal{C}$ składa się z

- endofunktora $T : \mathcal{C} \to \mathcal{C}$,
- naturalnej transformacji $\eta : 1_{\mathcal{C}} \to T$ (unit z funktorów dołączonych),
- naturalnej transformacji $\mu : T^2 \to T$, która definiuje mnożenie na funktorze T

takich, że poniższe diagramy komutują w kategorii $\mathcal{C}^{\mathcal{C}}$





Diagramy te są bardzo podobne do tych, które pojawiły się przy definiowaniu obiektu monoidalnego [1.11]. Nie jest to przypadkiem: monady są obiektem monoidalnym w kategorii endofunktorów $\mathcal{C}^{\mathcal{C}}$ z binarnym działaniem $\mathcal{C}^{\mathcal{C}} \times \mathcal{C}^{\mathcal{C}} \to \mathcal{C}^{\mathcal{C}}$ będącym składaniem funktorów.

Przykłady

1. Rozważmy parę funktorów sprzężonych znaną z poprzednich wykładów
$$Set \underset{U}{\overset{F}{\leftrightarrows}} Ab$$
gdzie F to funktor rozpinający wolną grupę abelową o generatorach równych zbiorowi, a U zapomina strukturę grupy. Niech $\eta : 1_{Set} \implies UF$ oraz $\varepsilon : FY \implies 1_{Ab}$ będą unitem oraz counitem z definicji gunktorów sprzężonych.
Widzimy tutaj endofunktor UF oraz naturalną transformację η jak z definicji monady. Potrzebujemy jeszcze mnożenia na UF.
Naturalne przekształcenie $\varepsilon : FU \implies 1_{Ab}$ na dowolnej grupie A jest homomorfizmem ewaluującym formalną sumę jej elementów (obiekt z FUA) jako właściwy element grupy A. Możemy ten homomorfizm wyrazić jako funkcję, podkładając funktory U i F z odpowiednich stron, tzn. rozważając złożenie
$$U\varepsilon F : UFUF \to UF.$$
Jest to występujący w definicji monady sposób mnożenia funktorów.
2. W przykładzie z funkcją `maxElem`, endofunktorem T jest zmiana typów `int -> int option`. Naturalnym przekształceniem $\eta : 1_{\mathcal{C}} \to T$ jest funkcja `return`, a funkcja `bind` mówi nam jak nałożyć funkcję `int -> int option` na element typu `int option`, czyli element poddany już działaniu endofunktora T.
3. Rozważmy kategorię Set i funktor $T : Set \to Set$, $T(X) = X \cup \{X\}$. Przypomnijmy, że jest to funktor będący złożeniem zapominającego funktora z kategorii zbiorów z wyróżnionym punktem z funktorem do niego dołączonym. $\eta : 1_{Set} \to T$ posyła elementy X w elementy X, tj. singleton $\{X\}$ nie jest w obrazie. $\mu_X : T^2X \to TX$ pośle elementy X w X, a zbiory $\{X\}$ oraz $\{X \cup \{X\}$ w singleton $\{X\}$. Czy widzisz podobieństwo z przykładem wyżej?

Lemat 1.21

Każda para $L \vdash R$ funktorów sprzężonych zadaje monadę, gdzie

- RL jest endofunktorem T,
- unit z definicji pary funktorów sprzężonych $\eta : 1_{\mathcal{C}} \to RL$ jest unitem z definicji monady,
- counit z nałożonymi funktorami, $R\varepsilon L : RLRL \implies RL$ jest mnożeniem $\mu : T^2 \to T$.

3. Konstruowanie funktorów sprzężonych z monad

Definicja 1.22

Niech $\mathcal{C}$ będzie kategorią z monadą (T, η, μ). Wtedy **kategorią Kleislego**, oznaczane $\mathcal{C}_T$, na $\mathcal{C}$ nazwiemy kategorię której

- obiekty są obiektami z $\mathcal{C}$
- morfizmy z A do B w $\mathcal{C}_T$, oznaczane (niekoniecznie konsekwentnie) $A \rightsquigarrow B$, jest morfizmem $A \to TB$ w kategorii $\mathcal{C}$.

Identyczność $id_A : A \rightsquigarrow A$ definiujemy, posiłkując się monadą, jako $\eta_A : A \to TA$. Złożenie morfizmów $f : A \rightsquigarrow B$ oraz $g : B \rightsquigarrow C$ to z kolei

$$A \xrightarrow{f} TB \xrightarrow{Tg} T^2C \xrightarrow{\mu_C} TC$$

Lemat 1.23

Składanie morfizmów w kategorii $\mathcal{C}_T$ jest łączne.

Dowód

Niech $f : A \rightsquigarrow B$, $g : B \rightsquigarrow C$ oraz $h : C \rightsquigarrow D$ będą morfizmami w kategorii $\mathcal{C}_T$. Chcemy pokazać, że $h \circ (g \circ f) = (h \circ g) \circ f$. Z definicji wiemy, że $h \circ g = \mu_D \circ Th \circ g$, ale ponieważ podkładamy pod to f, to musimy nałożyć na niego funktor T. Mamy diagram

$$\begin{array}{ccccccccccc}
A & \xrightarrow{f} & TB & \xrightarrow{Tg} & T^2C & \xrightarrow{\mu_C} & TC & \xrightarrow{Th} & T^2D & \xrightarrow{\mu_D} & TD \\
\| & & \| & & \| & & ? & & \| & & \| \\
A & \xrightarrow[f]{} & TB & \xrightarrow[Tg]{} & T^2C & \xrightarrow[T^2h]{} & T^3D & \xrightarrow[T\mu_D=\mu_{TD}]{} & T^2D & \xrightarrow[\mu_D]{} & TD
\end{array}$$

$h \circ (g \circ f)$

$(h \circ g) \circ f$

Punktem zapalnym jest $?$ w diagramie. Jeśli ten prostokąt komutuje, to koniec.

Z naturalności $\mu : T^2 \to T$ dostajemy komutujący diagram

$$\begin{array}{ccc}
T^2C & \xrightarrow{T^2h} & T^3D = T^2(TD) \\
{\scriptstyle \mu_C}\downarrow & & \downarrow{\scriptstyle \mu_{TD}} \\
TC & \xrightarrow[Th]{} & T(TD) = T^2D
\end{array}$$

czyli

$$\mu_{TD} T^2 h = TH\mu_C,$$

co daje nam równość przejścia po pomarańczowych strzałkach na górze (prawa strona równości) i na dole (lewa strona równości).

Przykład

Dla monady $T : Set \to Set$, $T(X) = X \cup \{X\}$ z przykładów wyżej, kategoria Kleisliego zawiera jako obiekty wszystkie zbiory. Morfizmy $A \rightsquigarrow B$ posyłają część elementów A w "kosmos", czyli singleton $\{B\}$. Są to funkcje częściowe! Czyli $Set_T = Set^{\delta}$ jest kategorią zbiorów z funkcjami częściowymi.

24.03.2025 Kategoria algebr i Eilenberga-Moore'a

tutaj kiedyś będzie wzmianka o modułach

Definicja 1.24: kategoria algebr

Niech (T, η, μ) będzie monadą na kategorii $\mathcal{C}$. Definiujemy wtedy **kateogrię algebr** na T, oznaczane $\mathfrak{alg}_T$, jako kategorię której

obiekty to pary (θ, c), gdzie $\theta : Tc \to c$

morfizmy $\mathfrak{alg}_T((\theta, c), (\theta', c'))$ to odwzorowania $f \in \mathcal{C}(c, c')$ takie, że komutuje diagram

$$\begin{array}{ccc} Tc & \xrightarrow{Tf} & Tc' \\ \downarrow{\scriptstyle\theta} & & \downarrow{\scriptstyle\theta'} \\ c & \xrightarrow{f} & c' \end{array}$$

Naturalnie, pytamy o istnienie obiektów początkowych i końcowych w tej kategorii.

Przykład

Niech $T \equiv c$ będzie funktorem stałym. Wtedy (θ, x) jest obiektem początkowym, jeśli dla każdego (ψ, y) jest dokładnie jeden komutujący diagram

$$\begin{array}{ccc} Tx = c & \xrightarrow{Tf = id_c} & c = Ty \\ \downarrow{\scriptstyle\theta} & & \downarrow{\scriptstyle\psi} \\ x & \xrightarrow[f]{} & y \end{array}$$

czyli $\psi = f \circ \theta$. Możemy wywnioskować, że $(\theta, x) = (id_c, c)$ i wtedy dla każdego innego (ψ, y) będzie jedyny morfizm $f = \psi$ spełniający diagram.

1. Kategoria Eilenberga-Moore'a

Definicja 1.25: Eilenberg-Moore

Kategoria **Eilenberga-Moore'a** dla T (kategorię T-algebra), oznaczaną jako $\mathcal{C}^T \subseteq \mathfrak{alg}_T$, jest podkategorią $\mathfrak{alg}_T$ w której

obiekty to pary (θ, a), $a \in \mathcal{C}$, $\theta : Ta \to a$ dla których komutują diagramy w $\mathcal{C}$

$$\begin{array}{ccc} a & \xrightarrow{\eta_a} & Ta \\ & \searrow^{1_a} & \downarrow{\scriptstyle\theta} \\ & & a \end{array} \qquad \begin{array}{ccc} T^2a & \xrightarrow{\mu_a} & Ta \\ \downarrow{\scriptstyle Ta} & & \downarrow{\scriptstyle\theta} \\ Ta & \xrightarrow[\theta]{} & a \end{array}$$

morfizmy $f : (\theta, a) \to (\varphi, b)$ są mapami $f : a \to b$ w $\mathcal{C}$ takie, że komutuje diagram

$$\begin{array}{ccc} Ta & \xrightarrow{Tf} & Tb \\ {\scriptstyle\theta}\downarrow & & \downarrow{\scriptstyle\varphi} \\ a & \xrightarrow[f]{} & b \end{array}$$

Chcemy teraz pokazać, że dla dowolnej monady możemy stworzyć parę funktorów sprzężonych.

Lemat 1.26

Istnieją funktory

$$\mathcal{C} \xrightarrow{L} \mathcal{C}_T \xrightarrow{J} \mathcal{C}^T \xrightarrow{R} \mathcal{C}_T$$

takie, że $RJL = T$.

Dowód

Zaczynamy od zdefiniowania wszystkich funktorów.

$$L : \mathcal{C} \to \mathcal{C}_T$$
$$L(c) = c, \quad L(f) = \eta \circ f$$

Wypada sprawdzić, czy $\eta \circ (gf) = L(gf) = L(g)L(f) = (\eta \circ g) \circ (\eta \circ f)$, czyli czy komutuje największy prostokąt w diagramie

$$\begin{array}{ccccccc} c & \xrightarrow{f} & c' & \xrightarrow{g} & c'' & \xrightarrow{\eta} & Tc'' \\ \| & & \downarrow{\scriptstyle\eta} & & \downarrow{\scriptstyle T(\eta)\circ\eta?} & & \| \\ c & \xrightarrow[\eta\circ f]{} & Tc' & \xrightarrow[\substack{T(\eta\circ g)= \\ =T(\eta)\circ T(g)}]{} & T^2c'' & \xrightarrow[\mu_{c''}]{} & Tc'' \end{array}$$

$$J : \mathcal{C}_T \to \mathcal{C}^T$$
$$J(c) = (Tc, \mu \circ \eta), \quad J(f : c \to Tc') = \mu \circ Tf : Tc \to Tc'$$

$$R : \mathcal{C}^T \to \mathcal{C}$$
$$R(c, \theta) = c, \quad R(f : (c, \theta) \to (c', \theta')) = f : c \to c'$$

Tutaj składanie działa bez problemów, bo f było morfizmem w $\mathcal{C}$.

Teraz pokażemy, że $RJL = T$. Dla obiektów mamy:

$$RJL(c) = RJ(c) = R(Tc) = Tc,$$

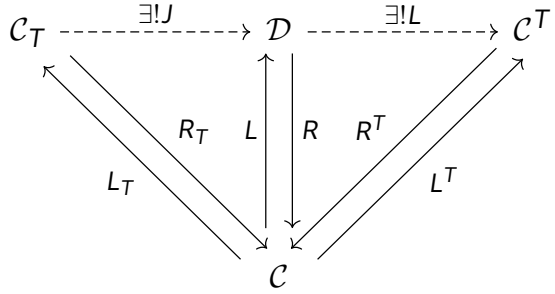
a dla dowolnego morfizmu $f : c \to c'$

$$RJL(f) = RJ(\eta \circ f) = R(\mu \circ T(\eta \circ f)) = R(\mu \circ T(\eta) \circ Tf) = R(1_T \circ Tf) = R(Tf) = Tf.$$

Pozostawiam ten dowód jako pomnik dla oryginalnego stwierdzenia, że RJ i L oraz R i JL to dwie pary funktorów dołączonych.

Twierdzenie 1.27

Dla dowolnej pary funktorów sprzężonych $\mathcal{C} \underset{R}{\overset{L}{\rightleftarrows}} \mathcal{D}$ indukujących monadę (T, η, μ) istnieją jedyne funktory J i K



takie, że prawy i lewy trójkąt komutuje.

Dowód

Emily Proposition 5.2.12.

31.03.2025 we back in business

Przykłady

1. $F : Set \to Monoid$, $U : Monoid \to Set$ -> wolny monoid $\iff$ słowa z konkatenacją
   $Set \xrightarrow{T} Set$ zbiór idzie w listę, $\eta : x \mapsto [x]$ idzie w jednoelementową listę, μ to spłaszczanie list
2. $F : Set \to AbMonoid$ przedłużamy
   tutaj $X \mapsto \{f : X \to \mathbb{N}, f = 0$ skończenie wiele razy $\}$, $\eta : x \to \delta_x$ (delta diraca), $\mu(\sum_n m_n \sum_x n_x x) = \sum(\sum m_n n_x)x$
3. $Vect \xrightarrow{F} AbAlg_k$, $V \mapsto \oplus S^n V$ podprzestrzeń $V^{\otimes n}$ niezmiennicza na S_n. η jest włożeniem
4. $F : Vect \to Alg_k$, $V \mapsto \oplus V^{\otimes n}$

1. Diagramy strunowe [string diagrams]

Do tej pory rysowaliśmy kropki jako kategorie, a strzałki jako funktory. Zmieniamy teraz konwencję i piszemy funktory jako kropki oraz kategorie jako kreski.

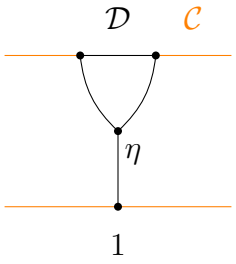
$$\mathcal{E} \qquad \mathcal{D} \qquad \mathcal{C}$$

dokończyć rysunek wyżej

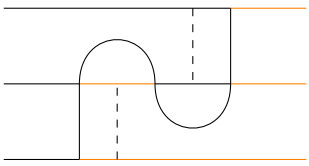
Niech teraz $L \vdash R$ będzie parą funktorów pochodnych i $\eta : 1_{\mathcal{C}} \implies RL$.

Diagramy czytamy od dołu do góry i od lewej do prawej.

Tutaj mamy narysowany unit



$(\varepsilon 1_L)(1_L \eta)$ to z kolei



zdjęcia + obrazki dla monady

maybe, reader monad