(CR15) CATEGORY THEORY FOR COMPUTER SCIENTISTS: LECTURE 9

10 OCTOBER 2024 — L. T. D. NGUYỄN

Last time: adjoint functors; preservation of (co)products.

Proof that right adjoints preserve products. Let $L \dashv R$ with $R\colon \mathcal{C} \to \mathcal{D}$. Let $(B, (\pi_i)_{i\in I})$ be a product of $(A_i)_{i\in I}$ in $\mathcal{C}$. Recall (from Lecture 7 on representable functors) that this is equivalent to having a natural bijection

$$\mathcal{C}(-, B) \cong \prod_{i\in I} \mathcal{C}(-, A_i)$$

that maps id_B to $(\pi_i)_{i\in I}$, which implies that it maps any f to $(\pi_i \circ f)_{i\in I}$. We have:

$$\mathcal{D}(-, R(B)) \cong \mathcal{C}(L(-), B) \cong \prod_{i\in I} \mathcal{C}(L(-), A_i) \cong \prod_{i\in I} \mathcal{D}(-, R(A_i))$$

This shows that $R(B)$ is a product of $(R(A_i))_{i\in I}$, for a family of projections which is the image of $\mathrm{id}_{R(B)}$ by this natural bijection (obtained by composition), that is:

$$\big((\theta_{X,A_i} \circ (\pi_i \circ -) \circ \theta^{-1}_{R(B),B})(\mathrm{id}_{R(B)})\big)_{i\in I}$$

To simplify this expression and conclude, it suffices to observe that, by naturality, we have $\theta_{X,A_i} \circ \underbrace{(\pi_i \circ -)}_{\mathcal{C}(L(X),\pi_i)} = \underbrace{(R(\pi_i) \circ -)}_{\mathcal{D}(X,R(\pi_i))} \circ \theta_{X,A_i}$. ☐

Monads

Definition. Consider an adjunction $\begin{tikzcd} \mathcal{C} \arrow[r, bend left, "L"] \arrow[r, phantom, "\perp"] & \mathcal{D} \arrow[l, bend left, "R"] \end{tikzcd}$ with unit η and counit ε.

The *monad induced by this adjunction* is (M, μ, η) where

$$M = R \circ L \qquad \mu = R(\varepsilon_L) = (R(\varepsilon))_L \text{ (cf. Lecture 6)}$$

Thus $M\colon \mathcal{C} \to \mathcal{C}$ is an endofunctor, $\eta\colon \mathrm{Id}_{\mathcal{C}} \Rightarrow M$ and

$$\mu\colon R \circ (L \circ R) \circ L \Rightarrow R \circ L \qquad \text{i.e.} \qquad \mu\colon M \circ M \Rightarrow M$$

We call η the *unit* of the monad, and μ its *multiplication*.

Note that the types of η and μ only refer to M, not to L and R separately. This will allow us to give later a direct definition of monads of the form "these diagrams involving η and μ must commute".

First, we look at monads on **Set**.

List monad: From the free/forgetful adjunction $(-)^* \dashv U$ we get a monad $(\mathsf{List}, \eta, \mu)$ where $\eta_X(x) = [x]$ and μ "flattens" lists of lists:

$$\begin{aligned} \mu_X\colon \mathsf{List}(\mathsf{List}(X)) &\to \mathsf{List}(X) \\ [\ell_1, \dots, \ell_n] &\mapsto \ell_1 \cdot \ldots \cdot \ell_n \end{aligned}$$

State monad: From the adjunction $(-) \times S \dashv \mathbf{Set}(S, -)$ we get a monad $(\mathsf{State}_S, \eta, \mu)$ where $\mathsf{State}_S(X) = \mathbf{Set}(S,\, X \times A)$ and

$$\begin{aligned} \eta_X \colon X &\to \mathbf{Set}(S,\, X \times S) \\ x &\mapsto (s \mapsto (x, s)) \quad \text{(cf. Lecture 8)} \\ \mu_X \colon \mathbf{Set}(S,\, \mathbf{Set}(S,\, X \times S) \times S) &\to \mathbf{Set}(S,\, X \times S) \\ m &\mapsto (s \mapsto \varepsilon_{X \times S}(m(s))) \end{aligned}$$

where $\varepsilon_Y \colon (f, s') \mapsto f(s')$ for $g \colon S \to Y$ and $s' \in S$.

The surprising thing is that monads on **Set** can represent *computational effects*.[1]

Definition ("bind" operator)**.** Let (M, μ, η) be a monad with $M \colon \mathbf{Set} \to \mathbf{Set}$. Let $m \in M(X)$ and $f \colon X \to M(Y)$. We set $m \mathbin{\texttt{>>=}} f = \mu_Y(M(f)(m))$.

For the list monad, this is the `concat_map` function in OCaml:

$$[x_1, \ldots, x_n] \mathbin{\texttt{>>=}} f = f(x_1) \cdot \ldots \cdot f(x_n)$$

For example $[1, 2] \mathbin{\texttt{>>=}} (x \mapsto [0, 3x]) = [0, 3, 0, 6]$. We can also chain >>=:

$$[1, 2] \mathbin{\texttt{>>=}} (x \mapsto [0, 3x] \mathbin{\texttt{>>=}} (y \mapsto [x + y])) = [1, 4, 2, 8]$$

If we consider a list as an (ordered[2]) *nondeterministic superposition* then the above can be seen, informally, as the semantics of a nondeterministic program:

$$x \leftarrow \texttt{choose}\ [1, 2];\ y \leftarrow \texttt{choose}\ [0, 3x];\ \texttt{return}\ (x + y)$$

With this point of view, $\eta(z) = [z]$ is the deterministic choice of the "pure value" z.

The state monad corresponds to having a mutable global variable, let's call it v:

$$m \in \mathsf{State}_S(X) \qquad \iff \qquad m \colon \overbrace{S}^{\text{old value of } v} \to \underbrace{X}_{\text{return value}} \times \overbrace{S}^{\text{new value of } v}$$

$\eta_X(x) : s \mapsto (x, s)$ is a computation that returns x without changing the state, and

$$m \mathbin{\texttt{>>=}} f = \big[s \mapsto \texttt{let}\ (x, s') = m(s)\ \texttt{in}\ f(x)(s')\big]$$

So we can represent the semantics of $x \leftarrow v;\ v := x + 1;\ \texttt{return}\ x$ (for $S = \mathbb{N}$) as

$$\texttt{get} \mathbin{\texttt{>>=}} (x \mapsto \texttt{put}(x + 1) \mathbin{\texttt{>>=}} (y \mapsto \eta(x))) \quad = \quad (s \mapsto (s, s + 1))$$

$$\text{where} \qquad \begin{aligned} \texttt{get} \colon S &\to S \times S \\ s &\mapsto (s, s) \end{aligned} \qquad \begin{aligned} \texttt{put} \colon S &\to \mathbf{Set}(S,\, \{*\} \times S) \\ s &\mapsto \big[s' \mapsto (*, s)\big] \end{aligned}$$

(these translations between imperative programs and chains of >>= correspond roughly to Haskell's "do-notation").

Let us mention yet another example of cultural interest: the *continuation monad* below models effects that act on the control flow, such as `call/cc`.

$$\mathbf{Set}(\mathbf{Set}(-, A), A) \quad \text{coming from} \quad \begin{tikzcd} \mathbf{Set}^{\mathrm{op}} \arrow[r, bend left, "\mathbf{Set}(-,A)"] \arrow[r, phantom, "\top"] & \mathbf{Set} \arrow[l, bend left, "{[\mathbf{Set}(-,A)]^{\mathrm{op}}}"] \end{tikzcd}$$

Given two functions $f \colon X \to M(Y)$ and $g \colon Y \to M(Z)$, there is now an obvious way to "plug them together": define

$$f \mathbin{\texttt{>=>}} g = (x \mapsto f(x) \mathbin{\texttt{>>=}} g)$$

[1] A few bibliographic references about monads in programming were given in the notes for Lecture 1.

[2] To represent nondeterminism without order and multiplicity one can use the covariant powerset monad instead; see the coalgebra course (CR17).

For example, $(y \mapsto [1+y,\ 2]) \mathbin{\texttt{>=>}} (x \mapsto [0, 3x]) = (y \mapsto [0, 3+3y, 0, 6])$. Intuitively, this is a kind of "composition of functions with effects" – here, the effect is a sort of non-determinism. This new operator generalizes to arbitrary categories:

Definition. Let $\mathcal{C}$ be a category, $M\colon \mathcal{C} \to \mathcal{C}$ and $\mu\colon M \circ M \Rightarrow M$. We define the *Kleisli composition* of $g \in \mathcal{C}(B, M(C))$ and $f \in \mathcal{C}(A, M(B))$ as

$$g \mathbin{\texttt{<=<}} f = \mu_C \circ M(g) \circ f$$

Proposition. *When $\mathcal{C} = \mathbf{Set}$, this general definition of Kleisli composition agrees with the set-specific one (using $\texttt{>>=}$): $g \mathbin{\texttt{<=<}} f = f \mathbin{\texttt{>=>}} g$.*

Proof idea. Just unfold the definitions. □

Since we have a composition operation, we'd like to use to build a *category*. But to do so, we'd need to check that, for example, Kleisli composition is associative, that is, $(h \mathbin{\texttt{<=<}} g) \mathbin{\texttt{<=<}} f = h \mathbin{\texttt{<=<}} (g \mathbin{\texttt{<=<}} f)$. The conditions that make this work are summed up in the following definition.

Definition. An *internal monoid in* $[\mathcal{C}, \mathcal{C}]$ is a 3-tuple (M, μ, η) where:

- M is an endofunctor of $\mathcal{C}$
- $\mu\colon M \circ M \Rightarrow M$ and $\eta\colon \mathrm{Id}_{\mathcal{C}} \Rightarrow M$ are natural transformations
- μ satisfies the *associativity law*: the diagram below commutes

$$\begin{tikzcd} M \circ M \circ M \arrow[r, Rightarrow, "\mu_M"] \arrow[d, Rightarrow, "M(\mu)"'] & M \circ M \arrow[d, Rightarrow, "\mu"] \\ M \circ M \arrow[r, Rightarrow, "\mu"'] & M \end{tikzcd}$$

- μ and η satisfy the *unit laws*: the diagram below commutes

$$\begin{tikzcd} M \arrow[r, Rightarrow, "M(\eta)"] \arrow[dr, Rightarrow, "\mathrm{id}_M"'] & M \circ M \arrow[d, Rightarrow, "\mu"] & M \arrow[l, Rightarrow, "\eta_M"'] \arrow[dl, Rightarrow, "\mathrm{id}_M"] \\ & M & \end{tikzcd}$$

  (this is equivalent to asking the left triangle and the right triangle to commute separately – each triangle corresponds to one of the two unit laws).

Remark. Recall that the notation $[\mathcal{C}, \mathcal{C}]$ stands for the category of functors $\mathcal{C} \to \mathcal{C}$ (with natural transformations as morphisms). Let us justify the name "internal monoid" by analogy. By formally replacing $([\mathcal{C}, \mathcal{C}], \circ, \mathrm{Id}_{\mathcal{C}})$ with $(\mathbf{Set}, \times, \{*\})$, we may define an *internal monoid in* $\mathbf{Set}$ as a set M with $\mu\colon M \times M \to M$ (so μ is a binary operation on M) and $\hat{e}\colon \{*\} \to M$, where the "associativity law" becomes

$$\begin{tikzcd} M \times M \times M \arrow[r, "\mu \times \mathrm{id}_M"] \arrow[d, "\mathrm{id}_M \times \mu"'] & M \times M \arrow[d, "\mu"] \\ M \times M \arrow[r, "\mu"'] & M \end{tikzcd}$$

stating that μ is associative, while the unit laws become the fact that $\hat{e}(*)$ is a unit for μ. So an internal monoid in $\mathbf{Set}$ is a monoid in the usual sense!

This analogy will be made technically rigorous in Olivier Laurent's part of the course using the notion of internal monoid in a *monoidal category*.

Proposition. *Every monad on $\mathcal{C}$ induced by an adjunction is an internal monoid in $[\mathcal{C}, \mathcal{C}]$.*

Proof. Let $L \dashv R$ an adjunction inducing a monad (M, μ, η).

If we expand the definitions $M = R \circ L$ and $\mu = R(\varepsilon_L)$ in the associativity diagram, we see that it is the image by $R(-)_L$ of

$$\begin{tikzcd}
L \circ R \circ L \circ R \arrow[r, Rightarrow, "\varepsilon_{L \circ R}"] \arrow[d, Rightarrow, "L \circ R(\varepsilon)"'] & L \circ R \arrow[d, Rightarrow, "\varepsilon"] \\
L \circ R \arrow[r, Rightarrow, "\varepsilon"'] & \mathrm{Id}_{\mathcal{D}}
\end{tikzcd}$$

This commutes by naturality of ε. In fact, it corresponds to the equality of the two ways of computing the *horizontal composition* $\varepsilon \circledast \varepsilon$ (cf. Lecture 6):

$$\begin{tikzcd}
\mathcal{D} \arrow[r, bend left, "L \circ R"] \arrow[r, bend right, "\mathrm{Id}_{\mathcal{D}}"'] \arrow[r, phantom, "\Downarrow \varepsilon"] & \mathcal{D} \arrow[r, bend left, "L \circ R"] \arrow[r, bend right, "\mathrm{Id}_{\mathcal{D}}"'] \arrow[r, phantom, "\Downarrow \varepsilon"] & \mathcal{D}
\end{tikzcd}$$

Thus, $\mu \circ \mu_M = R(\varepsilon \circledast \varepsilon)_L = \mu \circ M(\mu)$.

As for the unit laws, they are consequences of the *triangle identities* relating the unit and the counit of the adjunction that induces our monad. For instance the unit law involving $M(\eta)$ is the image by R of

$$\begin{tikzcd}
L \circ R \circ L \arrow[r, Rightarrow, "\varepsilon_L"] & L \\
L \arrow[u, Rightarrow, "L(\eta)"] \arrow[ur, Rightarrow, "\mathrm{id}_L"'] &
\end{tikzcd}$$

(this diagram has been copy-pasted from the previous lecture; to match it with the left triangle in the unit laws above, rotate it 90° clockwise...). $\square$

Theorem. *Given an internal monoid (M, μ, η) in $[\mathcal{C}, \mathcal{C}]$, the following data indeed defines a category, called the* Kleisli category $\mathcal{C}_M$ *of M:*

- *the objects of $\mathcal{C}_M$ are the same as of $\mathcal{C}$*
- *$\mathcal{C}_M(A, B) = \mathcal{C}(A, M(B))$ for any two objects A and B*
- *composition is the Kleisli composition* `<=<`
- *the identity for A in $\mathcal{C}_M(A, A) = \mathcal{C}(A, M(A))$ is η_A*

Proof. For $f \in \mathcal{C}_M(A, B) = \mathcal{C}(A, M(B))$, $g \in \mathcal{C}(B, M(C))$ and $h \in \mathcal{C}(C, M(D))$,

$$\begin{aligned}
(h \mathbin{\texttt{<=<}} g) \mathbin{\texttt{<=<}} f &= \mu_D \circ M(\mu_D \circ M(h) \circ g) \circ f \\
&= \mu_D \circ M(\mu_D) \circ M(M(h)) \circ M(g) \circ f \\
\text{(associativity law)} \quad &= \mu_D \circ \mu_{M(D)} \circ M(M(h)) \circ M(g) \circ f \\
\text{(naturality of } \mu) \quad &= \mu_D \circ M(h) \circ \mu_C \circ M(g) \circ f \\
&= h \mathbin{\texttt{<=<}} (g \mathbin{\texttt{<=<}} f)
\end{aligned}$$

One can also show that $f \mathbin{\texttt{<=<}} \eta_A = \eta_B \mathbin{\texttt{<=<}} f = f$ for $f \in \mathcal{C}(A, M(B))$ thanks to the unit laws concerning μ and η. $\square$

If we see a morphism $f \in \mathcal{C}_M(A, B) = \mathcal{C}(A, M(B))$ as an "effectful morphism" for the effect represented by the monad M – as in our previous examples over **Set** – this means we have a general well-behaved notion of "effectful composition". Now, let us see another remarkable property of Kleisli categories.

Theorem. *Let (M, μ, η) be an internal monoid in $[\mathcal{C}, \mathcal{C}]$. Then we have an adjunction $L \dashv R$ inducing the monad (M, μ, η), where*

$$\begin{aligned} L\colon \mathcal{C} &\to \mathcal{C}_M & R\colon \mathcal{C}_M &\to \mathcal{C} \\ A &\mapsto A & A &\mapsto M(A) \\ f \in \mathcal{C}(A,B) &\mapsto \eta_B \circ f & h \in \mathcal{C}_M(A,B) &\mapsto \mu_B \circ M(f) \end{aligned}$$

An intuition is that L includes "pure" morphisms into "effectful" morphisms, by postcomposing with η which sends pure values to computations that return them.

Corollary. *The internal monoids in $[\mathcal{C}, \mathcal{C}]$ are exactly the monads induced by adjunctions.*

Thus, we just call **monad** this concept that admits two equivalent definitions.[3]

Proof of the theorem. First, we need to check that L and R are functors.

For $f \in \mathcal{C}(A,B)$ and $g \in \mathcal{C}(B,C)$,

$$\begin{aligned} L(g) \texttt{<=<} L(f) &= \mu_C \circ M(\eta_C \circ g) \circ \eta_B \circ f \\ &= (\mu_C \circ M(\eta_C)) \circ (M(g) \circ \eta_B) \circ f \\ \text{(unit law + naturality of } \eta\text{)} \quad &= \mathrm{id}_{M(C)} \circ \eta_C \circ g \circ f \\ &= L(g \circ f) \end{aligned}$$

Similarly we can check that $R(g') \circ R(f') = R(g' \texttt{<=<} f')$, and that L and R preserve identities. For the adjunction, note that

$$\mathcal{C}_M(L(A), B) = \mathcal{C}(A, M(B)) = \mathcal{C}(A, R(B))$$

and, for $f \in \mathcal{C}(A', A)$ and $g \in \mathcal{C}_M(B, B')$, we have

$$\mathcal{C}_M(L(f), g) = (h \in \mathcal{C}(A, M(B)) \mapsto g \texttt{<=<} h \circ f) = \mathcal{C}(f, R(g))$$

so we have an equality of functors (i.e. equality on both objects and morphisms)

$$\mathcal{C}_M(L(-), -) = \mathcal{C}(-, R(-))$$

and two equal functors are naturally isomorphic by a family of identity morphisms. The unit of the adjunction is the image of the identity for $L(A)$ in $\mathcal{C}_M$ via the natural bijection... but this identity in $\mathcal{C}_M$ is η_A and the bijection sends it to itself.

Finally, the counit is $\varepsilon_A = \mathrm{id}_{M(A)} \in \mathcal{C}(R(A), R(A)) = \mathcal{C}_M(L(R(A)), A)$, therefore $R(\varepsilon_{L(A)}) = R(\varepsilon_A) = \mu_A \circ M(\mathrm{id}_{M(A)}) = \mu_A$: the induced monad is (M, μ, η). □

[3]The internal monoid definition explains this meme: `https://stackoverflow.com/questions/3870088/a-monad-is-just-a-monoid-in-the-category-of-endofunctors-whats-the-problem`