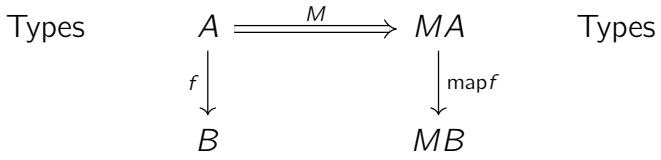
# 1 Monádok

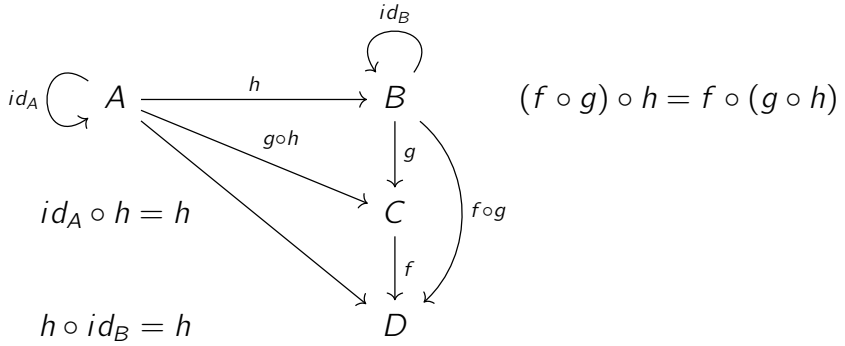
A **monád** egy olyan típuskészítő operáció (typeformer) a típusos-funkcionális programozásban (Simple/Dependent Type Theory), amely az imperatív programozási nyelvek működését modellezi funkcionálisan. Egyfelől input-output formába írja át a parancsokból felépülő, sokszor *mellékhatásokat* tartalmazó programokat, másfelől emulálja a parancsokból álló programokat. A monádok leggyakoribb intuitív ábrázolása a dobozolás:

$$a : A \mapsto \boxed{a} : MA$$

A típuselméletben a monád, egy olyan leképezés (endofunktor), amely egyrészt típusokból típusokat készít (polimorf típuskonstruktor), másfelől függvényekből függvényeket:




Monád és képe között két tovább művelettel lehet közlekedni, amelyek tulajdonságai szoros kapcsolatban vannak a típuselmélet kategóriaelméleti modelljével. Mint már láttuk, a típusok kategóriájában vannak kitüntetett függvények, és ezek között kitünetett műveleti tulajdonságok:




így az identitás és a kompozíció lesznek a műveletek, és a bal és jobb egységelem tulajdonságok és az asszociativitás a műveleti tulajdonságok. A két, a monád képével kapcsolatot teremtő operáció a return és a bind (és nem a fenti map, mely persze kifejeztehtő binddal).

Definíció szerint, ha $M : \text{Type} \to \text{Type}$ típusokból típusokat készítő leképezés,

$$\text{return}_A : A \to MA$$

függvény és

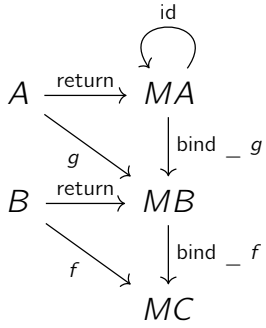
$$\text{bind}_{A,B} : MA \to (A \to MB) \to MB)$$

(másodrendű) függvény, akkor ez a hármas monád, ha

$$\begin{aligned}
\text{bind}_{A,A}\,\text{return}_A &= \text{id}_A \\
(\text{bind}_{A,B}\,g) \circ \text{return}_A &= g \qquad (g : A \to MB) \\
(\text{bind}_{B,C}\,f) \circ (\text{bind}_{A,B}\,g) &= \text{bind}_{A,C}((\text{bind}_{B,C}\,f) \circ g) \qquad (g : A \to MB, f : B \to MC)
\end{aligned}$$

Vegyü készre, hogy bind bemenet először is egy $MA$, és utána egy $A \to MB$ függvény, és $MB$-t ad vissza. Hogy az első bemenete $MA$, azt úgy jelöltük, hogy tettünk egy _ jelet a jobb oldalára. Bind tehát egyszerre bedobozoló és kidobozoló operáció:

$$\text{bind}\boxed{a}f = \boxed{f\,a}$$




Right identity rule: bind_ return = id

Left identity rule: (bind_ g) ∘ return = g

Associative rule: (bind_ f)∘(bind_ g) = bind_ ((bind_ f)∘g)

# 2 Coq implementáció

```
Structure Monad : Type := mk_monad {
  M : Type -> Type;

  bind : forall {A B : Type}, M A -> (A -> M B) -> M B;
  ret : forall {A : Type}, A -> M A;

  left_id_law : forall (A B : Type) (a : A) (f : A -> M B),
                bind (ret a) f = f a;

  right_id_law : forall (A : Type) (ma : M A),
                 bind ma ret = ma;

  assoc_law : forall (A B C : Type) (ma : M A) (f : A -> M B) (g : B -> M C),
              bind (bind ma f) g = bind ma (fun x => bind (f x) g)
}.
```

Ebben a változatban a `bind` első argumentuma a monádikus érték (`M A`), a második pedig a transzformáló függvény (`A -> M B`). Itt jól látszik a szabályok értelme:

1. left: a bind jelentése a függvénykiszámítás: kicsomagolja *ma*-t, és *a*-n kiszámolja *f*-et.

3. assoc: a bind tényleg a függvénykompozíció jelölési sorrendjét alakítja át jobb-balból bal-jobb-ba: $g(f(a))$-t cseréli le $a >>= f; f\,a >>= g$-re.

2. right: a program return-nel ér véget.

### Az `option` típus, mint monád

Az `option` típushoz tartozó monádikus műveletek definíciói:

```
Definition ret_opt (A : Type) := fun (a : A) => Some a.

Definition bind_opt {A B : Type} (ma : option A) (f : A -> option B) :=
  match ma with
  | Some a => f a
  | None => None
  end.
```

### A `list` típus, mint monád

Az `list` típushoz tartozó monádikus műveletek definíciói:

```
Definition ret_list (A : Type) := fun (a : A) => [a].

Fixpoint bind_list {A B : Type} (l : list A) (f : A -> list B) :=
  match l with
  | nil => nil
  | a :: l' => (f a) ++ (bind_list l' f)
  end.
```

## Mi köze van ennek a logikához?

A monádok típusai gyakran azokat a szabályokat testesíTik meg, amelyekben a spéci logikák különböznek a klasszikustól. Pl.

$$(A \to B) \to B) \to A$$

a continuation passing style

$$(A \to B) \to A) \to A$$

a call/cc alaptípusa, és ezek nem levezethetők, bár az első megfordítása és a második maga, a kasszikus logikában igaz.