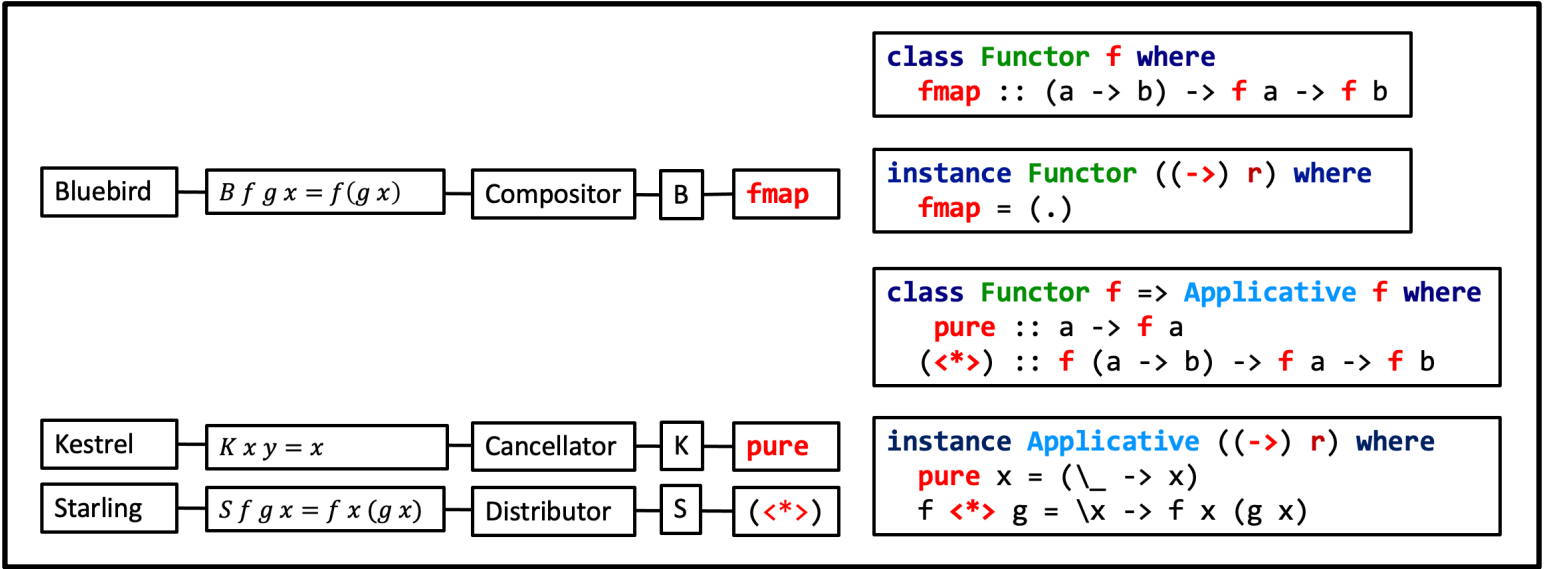


```
-- Haskell Curry's combinators S, B and I
s f g x = f x (g x)
b f g x = f (g x)
i x = x
```

```
-- Raymond Smullyan's combinator bird names
starling = s
blueBird = b
idiot = i
```

```
-- Alternative combinator names
distributor = s
compositor = b
identity = i
```



recursive definition

```
filter :: (a -> Bool) -> [a] -> [a]
filter _ [] = []
filter p (x:xs) = if (p x) then x:(filter p xs) else filter p xs
```

folding

```
filter p = foldr (\x -> if p x then (:) x else id) []
```

folding with combinators

```
filter p = foldr (s (b (bool i) (:)]) p) []
```

Curry's combinator names

```
filter p = foldr (starling (blueBird (bool idiot) (:)]) p) []
```

Smullyan's combinator names

```
filter p = foldr (distributor (compositor (bool identity) (:)]) p) []
```

Alternative combinator names

```
filter p = foldr (<*>) ((.) (bool id) (:)]) p []
```

prefix function invocation

```
filter p = foldr (((bool id) . (:)]) <*> p) []
```

infix function invocation

```
filter p = foldr (bool id . (:)]) <*> p []
```

no superfluous parentheses

```
-- returns x when c is False
-- and y when c is True
bool x y c = if c
             then y
             else x
```

