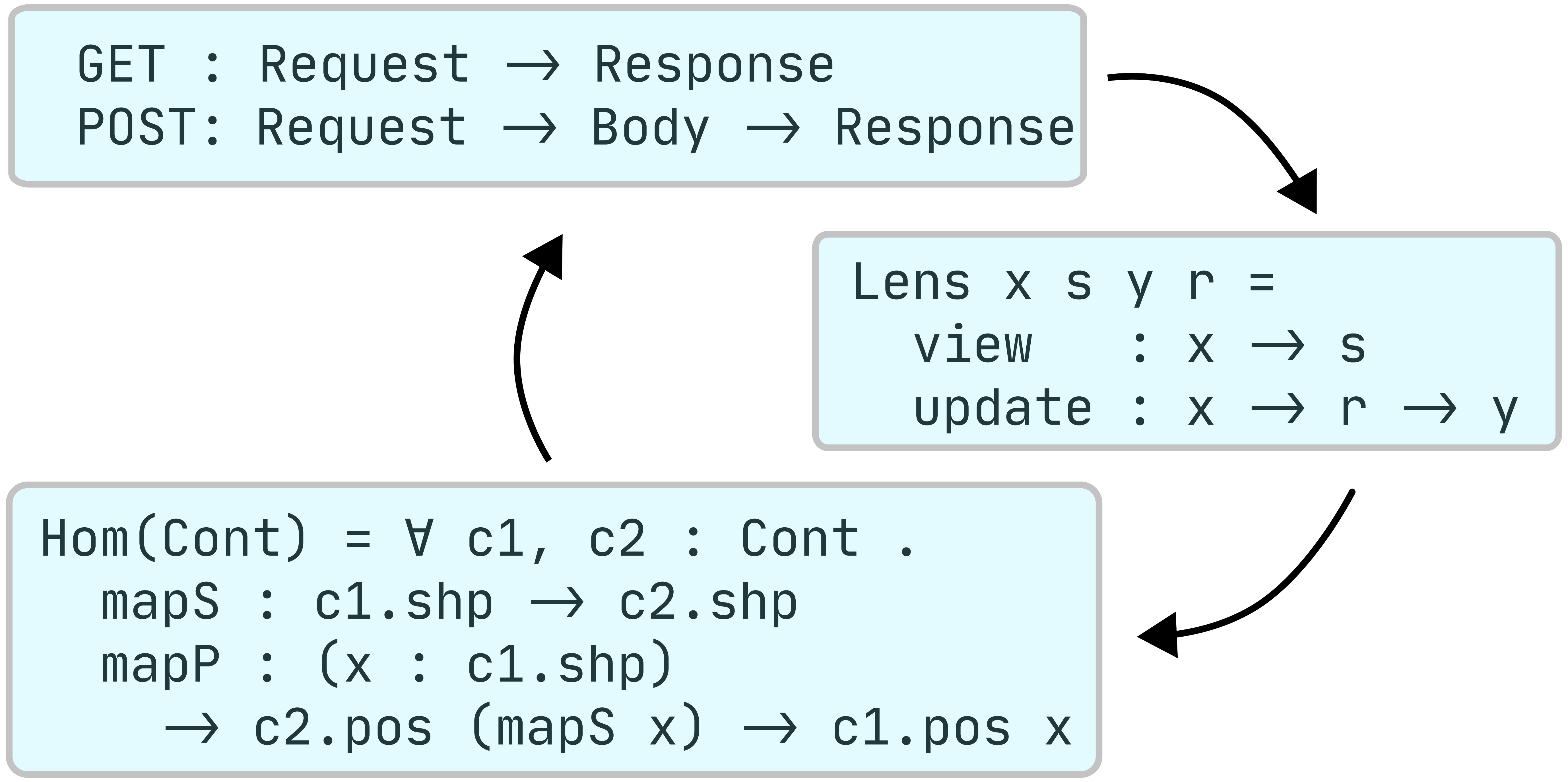
Servers X Lenses



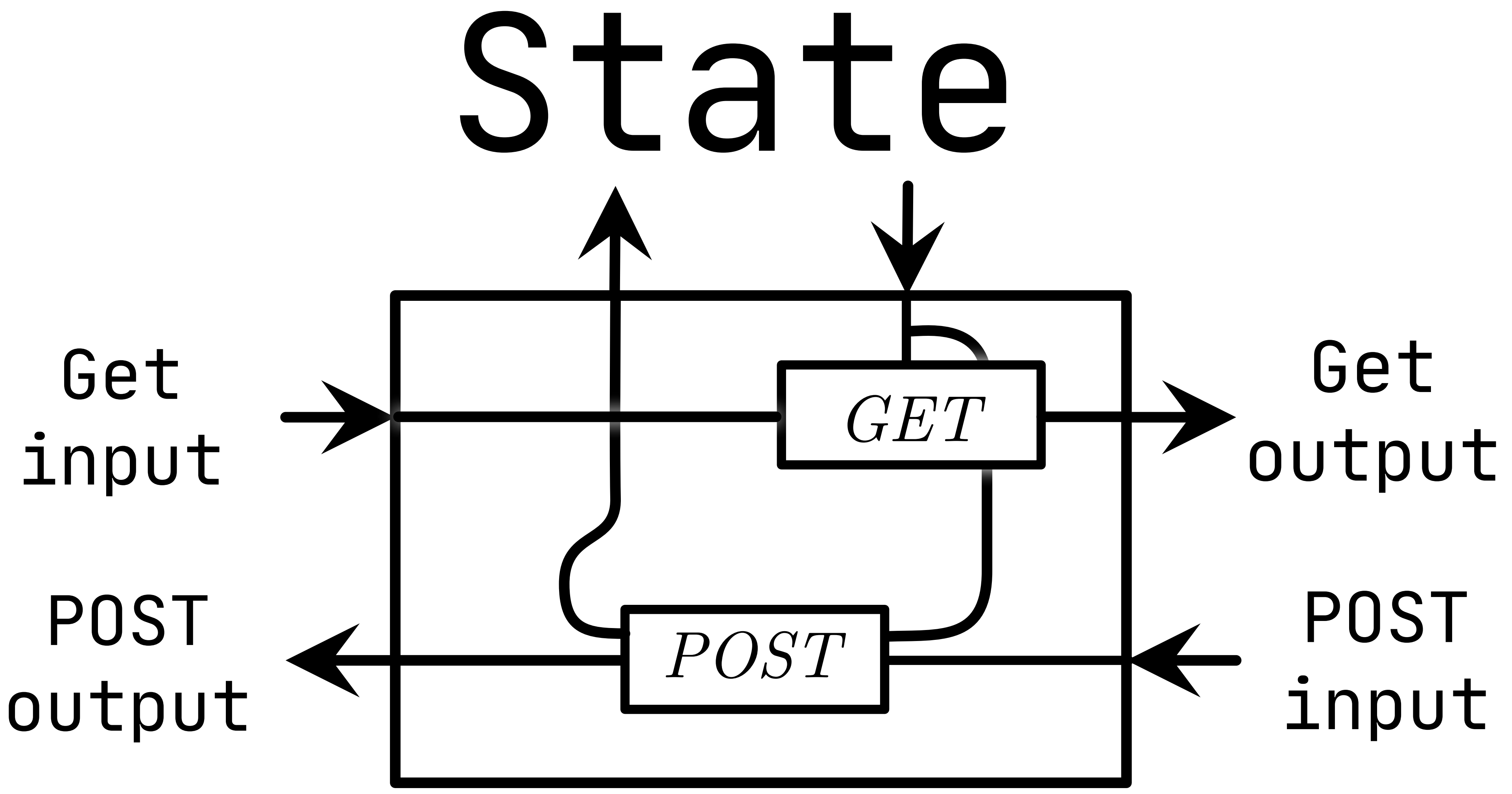
```
GET : Request → Response
POST: Request → Body → Response
```

```
Lens x s y r =
  view   : x → s
  update : x → r → y
```

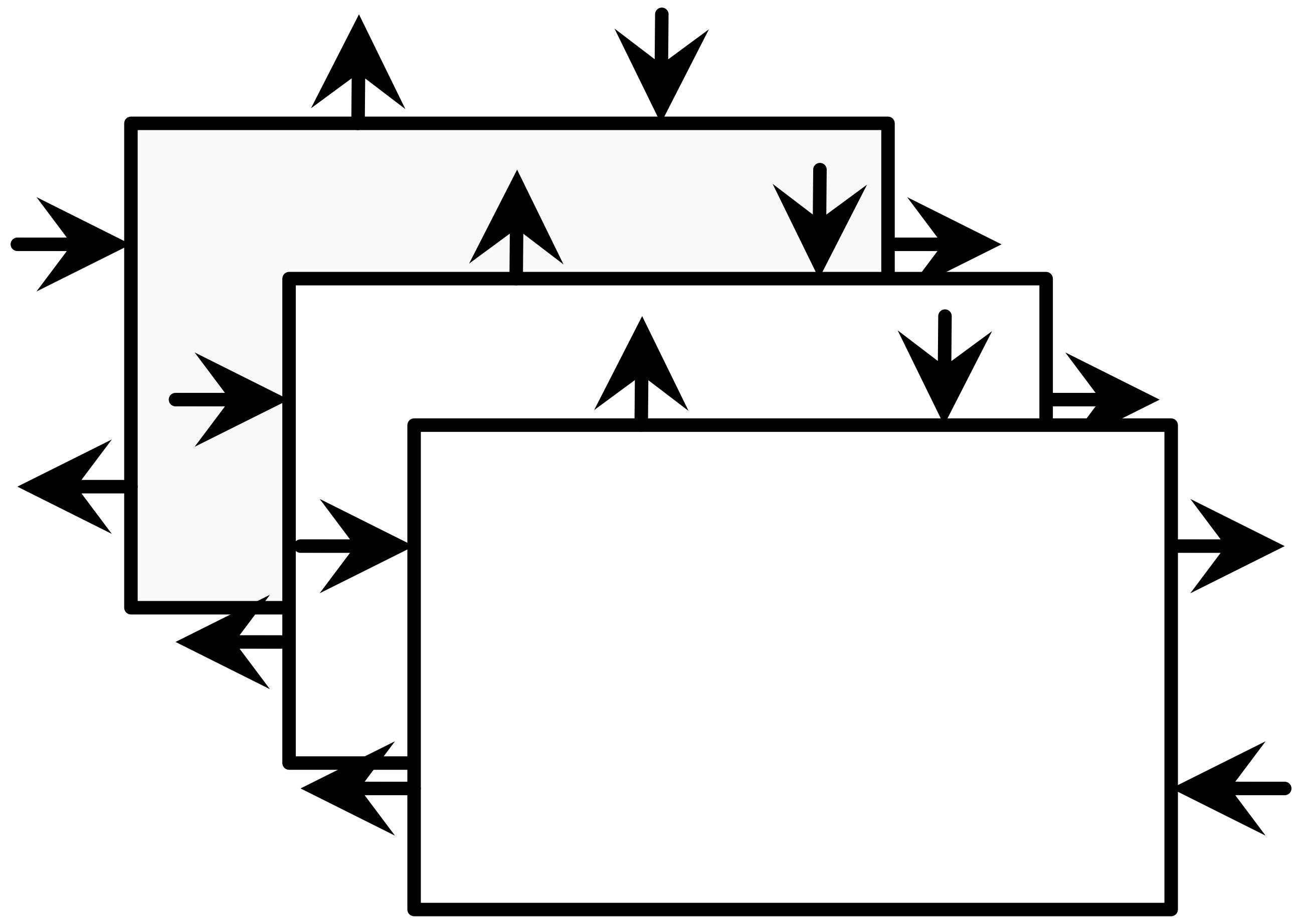
```
Hom(Cont) = ∀ c1, c2 : Cont .
  mapS : c1.shp → c2.shp
  mapP : (x : c1.shp)
    → c2.pos (mapS x) → c1.pos x
```

Web servers & Lenses

By noticing that GET and POST requests from servers look like lens, we employ them to define and implement servers. By making use of the *Para* construction we model state, and by making use of *Container Morphisms* as dependent lenses we combine multiple servers together.



A server as a lens



Choice of server

Implementing servers

We can instanciate lenses as servers by requiring that each input is parsable from a URI and each output is serialisable as a HTTP response body. Those constraints are closed under products and coproducts which we use to combine multiple servers together. Starting a server on localhost:8080 is as easy as:

```
mySerer = "path" / "to" / "resource" / Lens
      &&& "Other" / "resource" / Lens

main : IO ()
main = runServer Normal myServer def
```