Containers for Compiler architecture

Inspirations / sources

- Query-based compilers (https://ollef.github.io/blog/posts/query-based-compilers.html)
- Build-systems à la carte (https://www.microsoft.com/en-us/research/publication/build-systems-la-carte/)
- Compilers as databases (https://www.youtube.com/watch?v=w1ca4KL9UXc)
- The compiler forest (https://homepages.inf.ed.ac.uk/gdp/publications/compiler-forest.pdf)
- Interactive programs in Dependent Type Theory (https://link.springer.com/chapter/10.1007/3-540-44622-2_21)
- Ornaments (https://personal.cis.strath.ac.uk/conor.mcbride/pub/OAAO/LitOrn.pdf)

Requirements for a modern compiler

- Fast
- Multicore
- Incremental
- Interactive
- Online multi-player
- Easy to extend

~~Compiler functions~~

~~Compiler databases~~

Compilers are video games

Compiler Implementation challenges

- Interactive
- Always adding components
- Mutable state everywhere
- Input layer/ logic layer/ rendering engine

Games	Compilers
Object Tree	Syntax Tree
Ray cast	Cursor
Rendering engine	Editor output
Controller input	Keyboard input
Physics engine	Typechecker

Convinced?

Where are the containers?

Addressing extensibility

Trees are hard

```
data Lambda : Type where
  Var : String → Lambda
  Lam : String → Lambda → Lambda
  App : Lambda → Lambda → Lambda
```

Trees are hard

```
data LambdaF : Type → Type where
  VarF : a → String → LambdaF a
  LamF : a → String → LambdaF a → LambdaF a
  AppF : a → LambdaF a → LambdaF a → LambdaF a
```

Trees are hard

```
data Scoped : Nat → Type where
  VarS : Fin n → Scoped n
  LamS : Scoped (S n) → Scoped n
  AppS : Scoped n → Scoped n → Scoped n
```

Trees are hard

```
data ScopedF : Nat → Type → Type where
  VarSF : a → Fin n → ScopedF n a
  LamSF : a → ScopedF (S n) a → ScopedF n a
  AppSF : a → ScopedF n a → ScopedF n a → ScopedF n a
```

Solution?

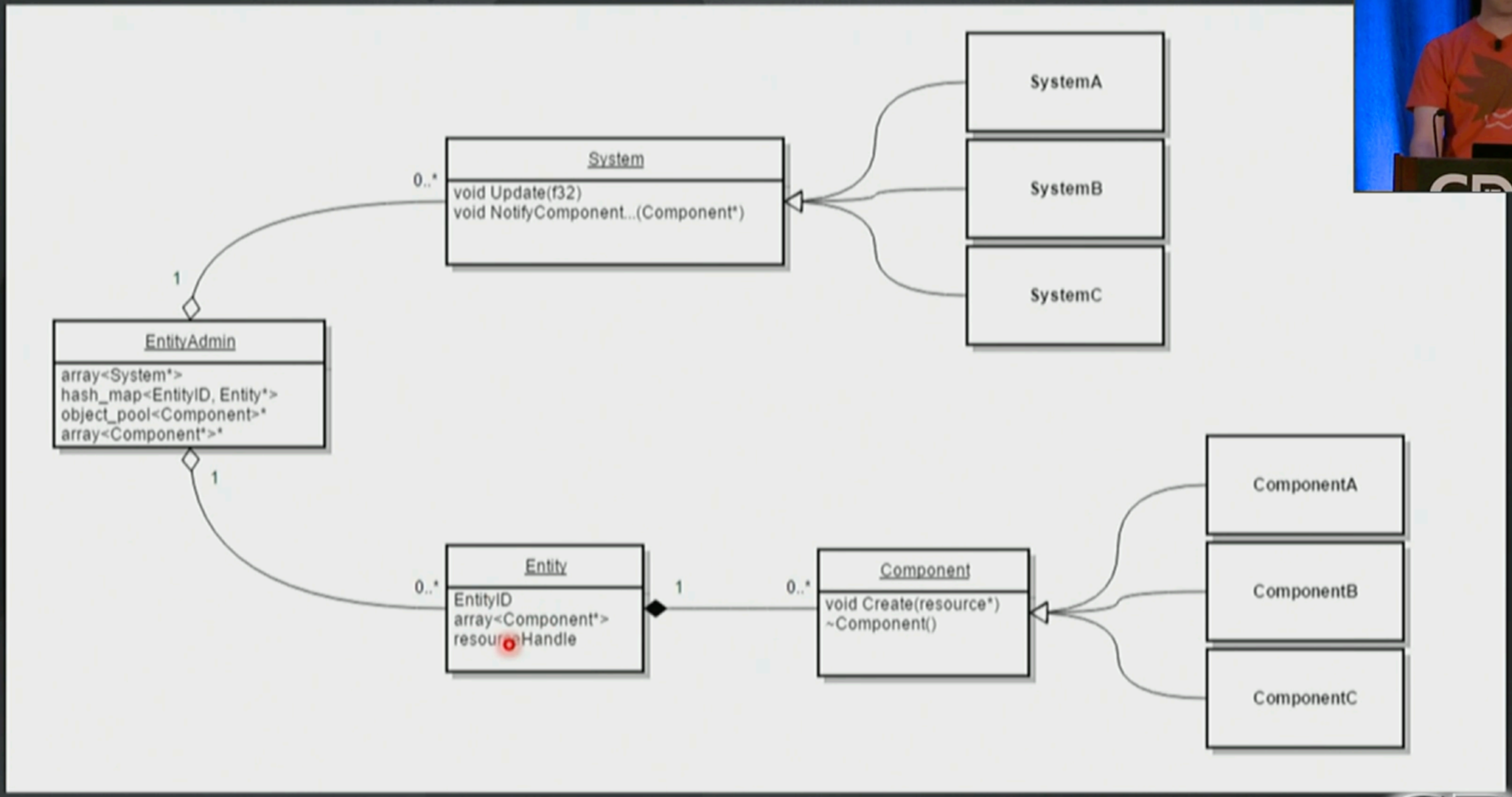
Ornaments for Trees

Demo

- Can't add constructors
- Not fast

Addressing speed

ECS for compiler architecture



source: https://www.gdcvault.com/play/1024001/-Overwatch-Gameplay-Architecture-and

Entities

Player

Projectile

Particle

SFX

Components

Movable

SubjectToPhysics

NetworkConn

HealthPoints

Systems

Physics engine

Renderer

NetworkRollBack

Damage calculation

Entities

Cursor

SyntaxFragment

Module

DisplayWindow

Components

Checkable

HasPosition

NetworkConn

HasConstraints

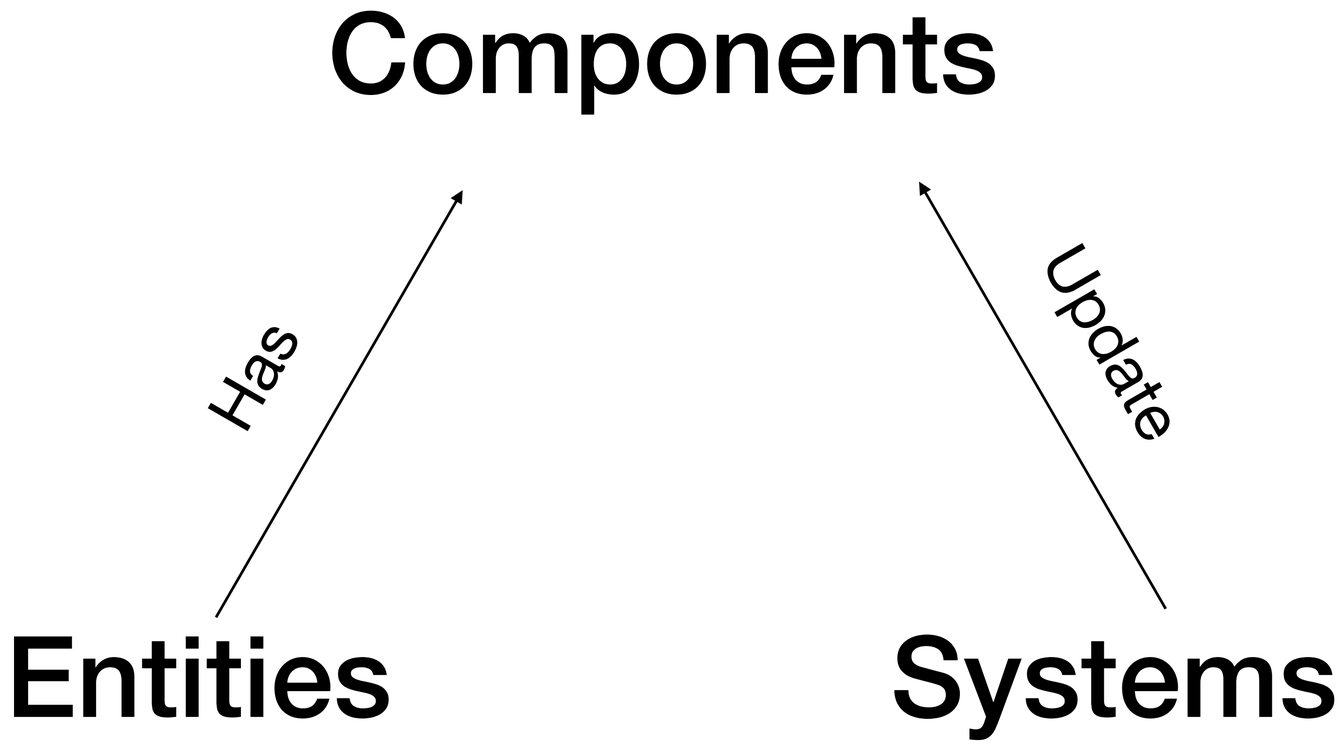
Systems

Typecheck

Elaborate

Render

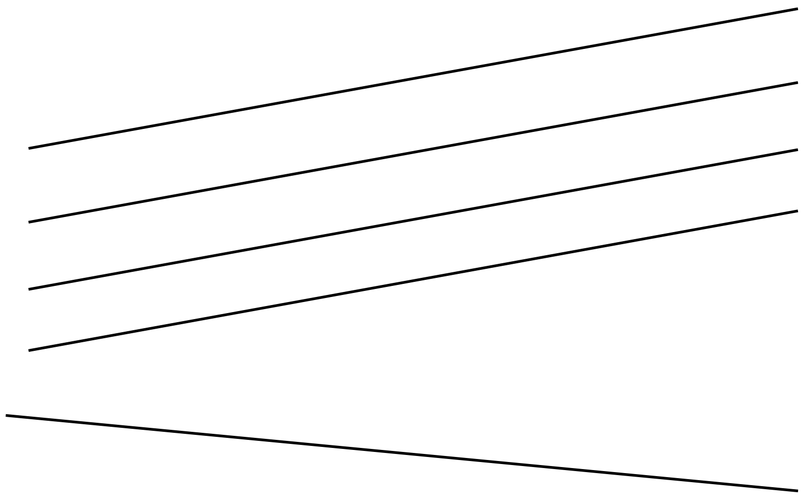
NetworkRollBack


Components
Has
Update
Entities
Systems

Player

Components

- Movable
- SubjectToPhysics
- NetworkSync
- AudioObject
- Render

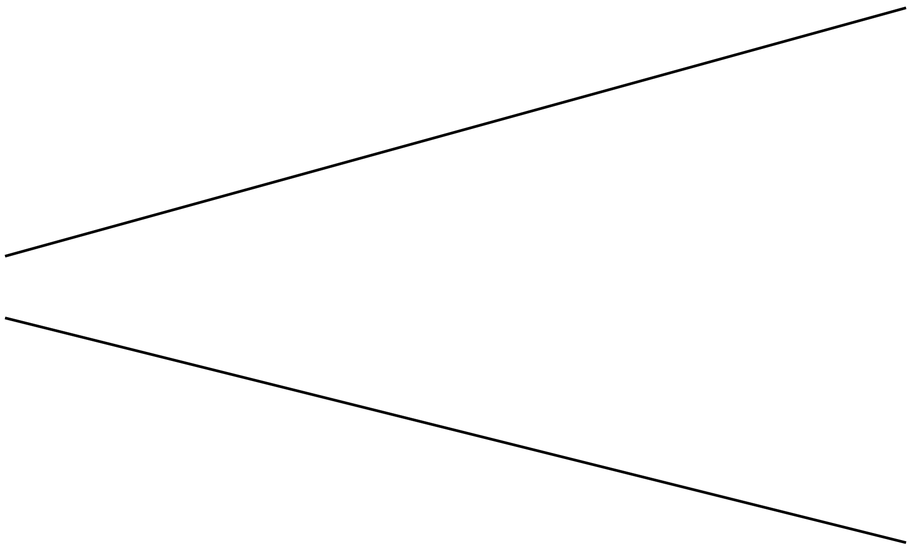


- Movable
- SubjectToPhysics
- NetworkSync
- AudioObject
- Despawn
- FluidSimulation
- UIRender
- Render
- Camera

Camera

Components

- Camera
- Movable



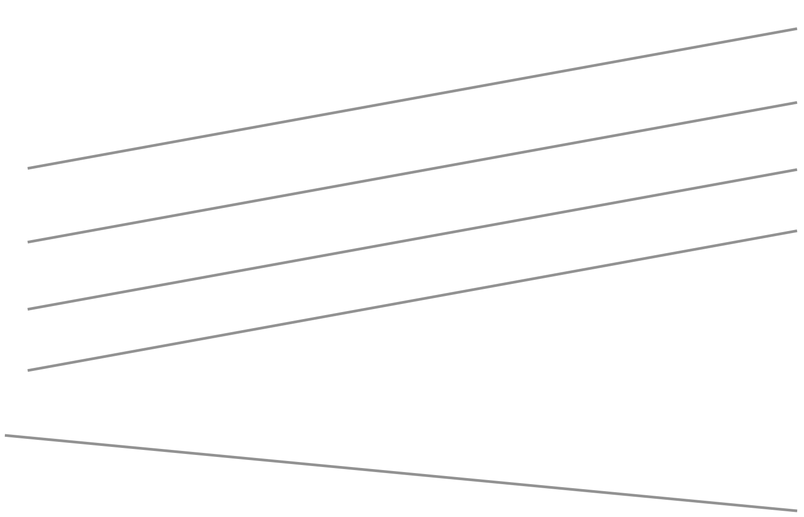
- Movable
- SubjectToPhysics
- NetworkSync
- AudioObject
- Despawn
- FluidSimulation
- UIRender
- Render
- Camera

Player

Components

- Movable
- SubjectToPhysics
- NetworkSync
- AudioObject
- Render

Thinning: 111100010



- Movable
- SubjectToPhysics
- NetworkSync
- AudioObject
- Despawn
- FluidSimulation
- UIRender
- Render
- Camera

Camera

Components

- Camera
- Movable

Thinning: 100000001

- Movable
- SubjectToPhysics
- NetworkSync
- AudioObject
- Despawn
- FluidSimulation
- UIRender
- Render
- Camera

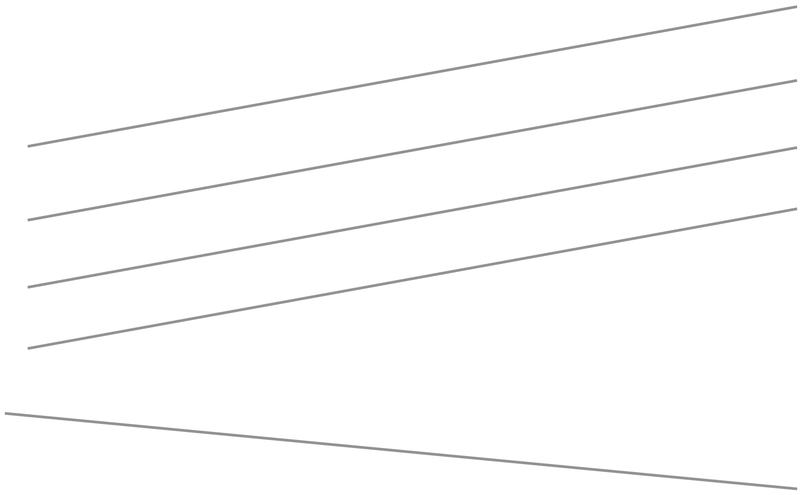
EntityID	Signature	Component1	Component2
be67f1c9-5a5c-4e7c-9489-293ee4928ba4	10000011	94190	0
2a2bf2e7-c953-4139-a359-0133f39c3201	11110011	5241	33862
4eb37d3a-2d5c-4dd8-8b62-676857e49367	10110001	82704	0
a4d4965e-7c02-49ec-aefa-721832d10d5a	10110001	28569	0

Player

Components

- Movable
- SubjectToPhysics
- NetworkSync
- AudioObject
- Render

Thinning: 1111000100



- Movable
- SubjectToPhysics
- NetworkSync
- AudioObject
- Despawn
- FluidSimulation
- UIRender
- Render
- Camera
- DetachLogic

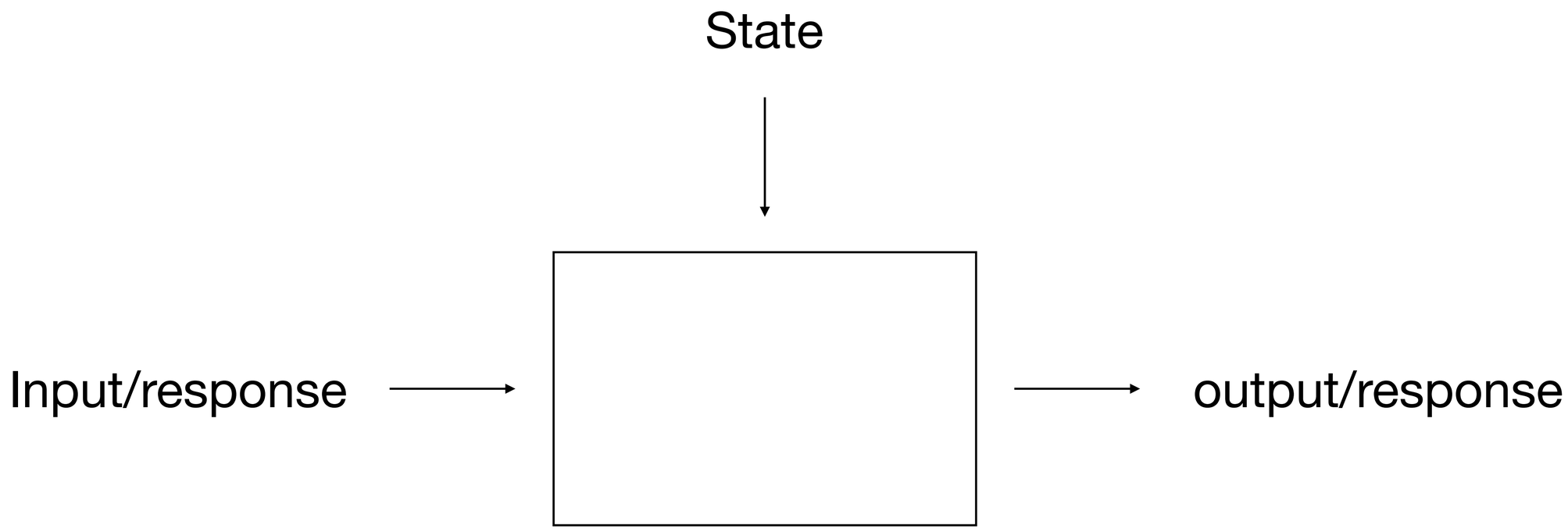
111100010

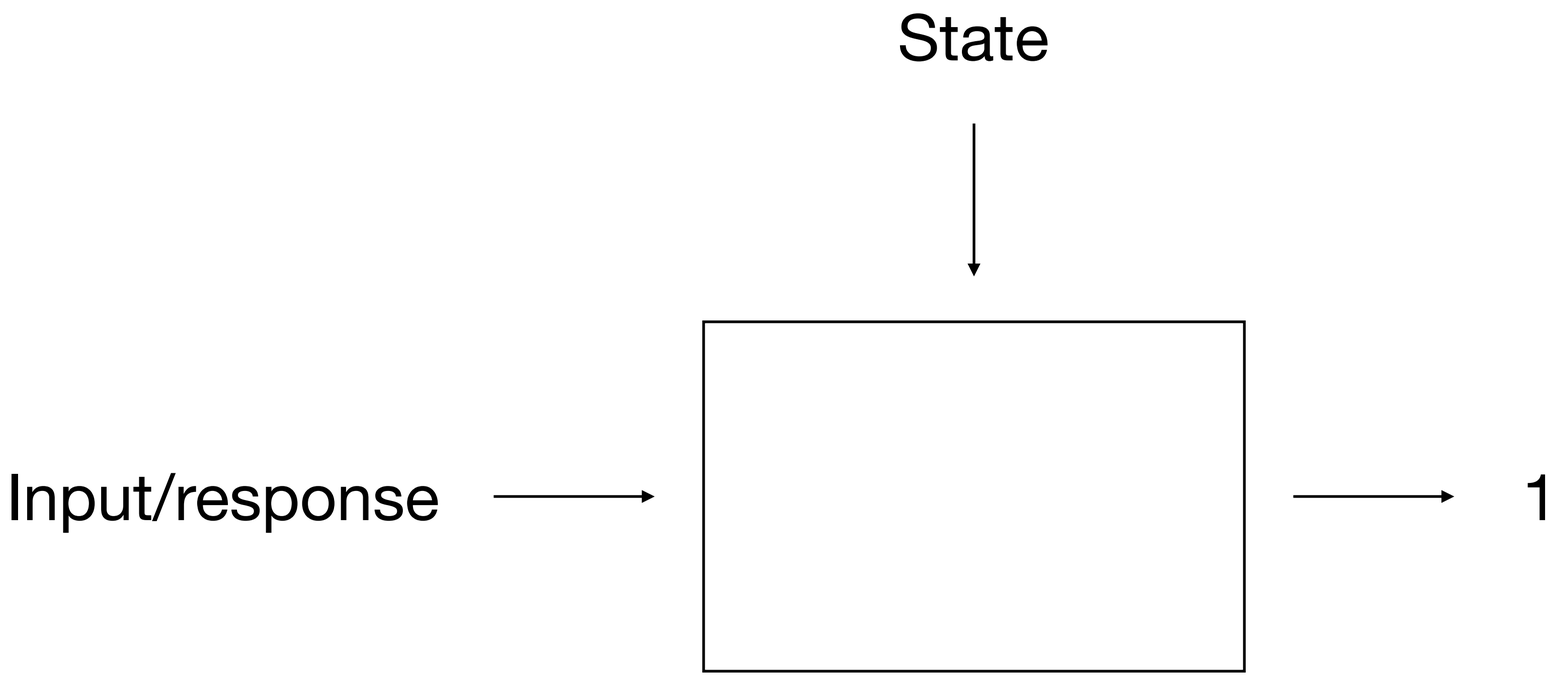
1111000100

Demo?


Demo?

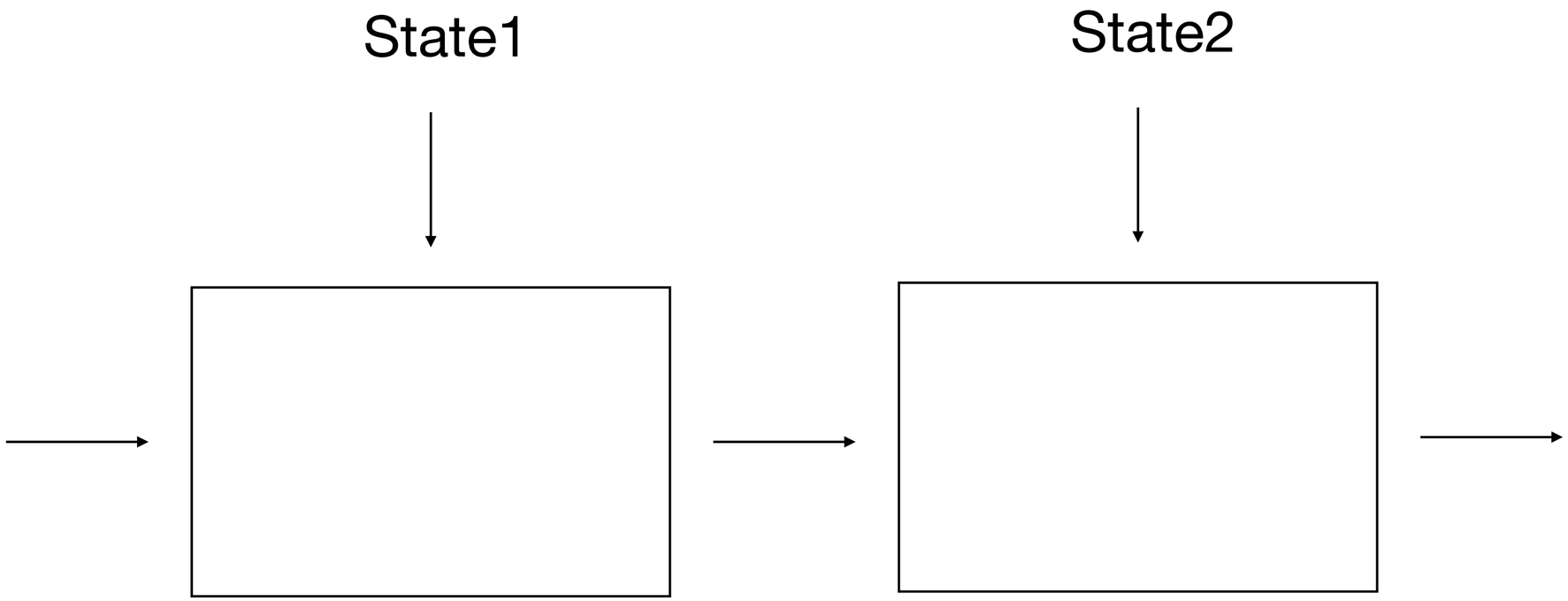
Addressing the interaction model

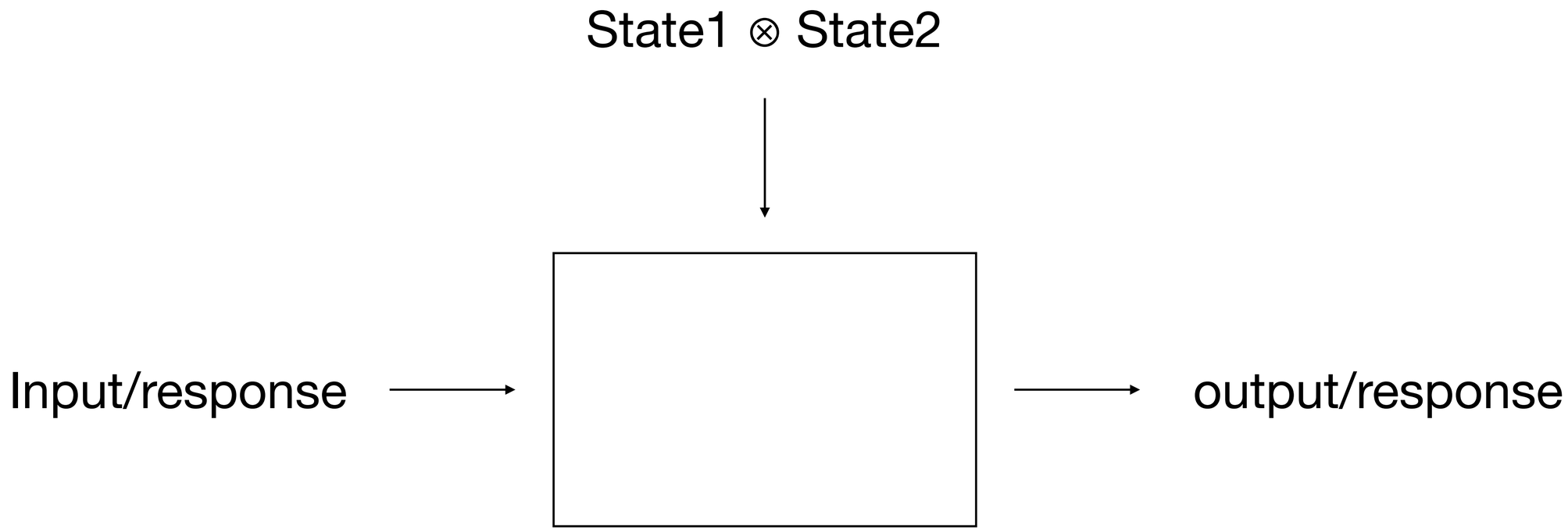

State
Input/response
output/response

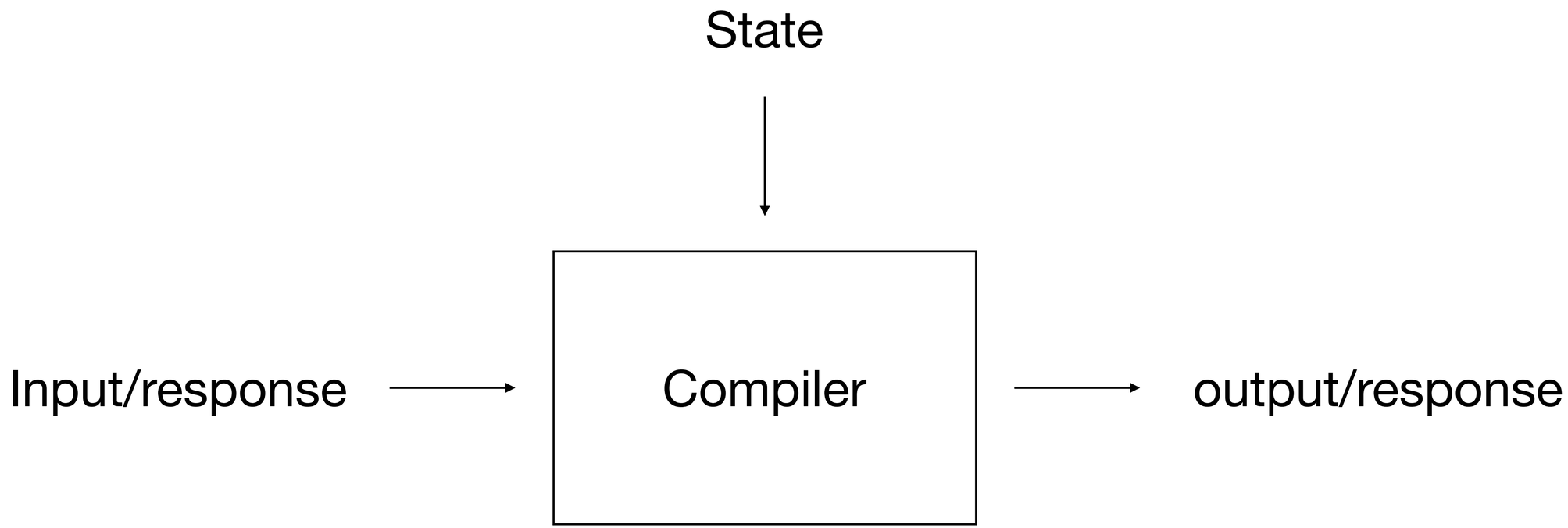


$$\sim$$

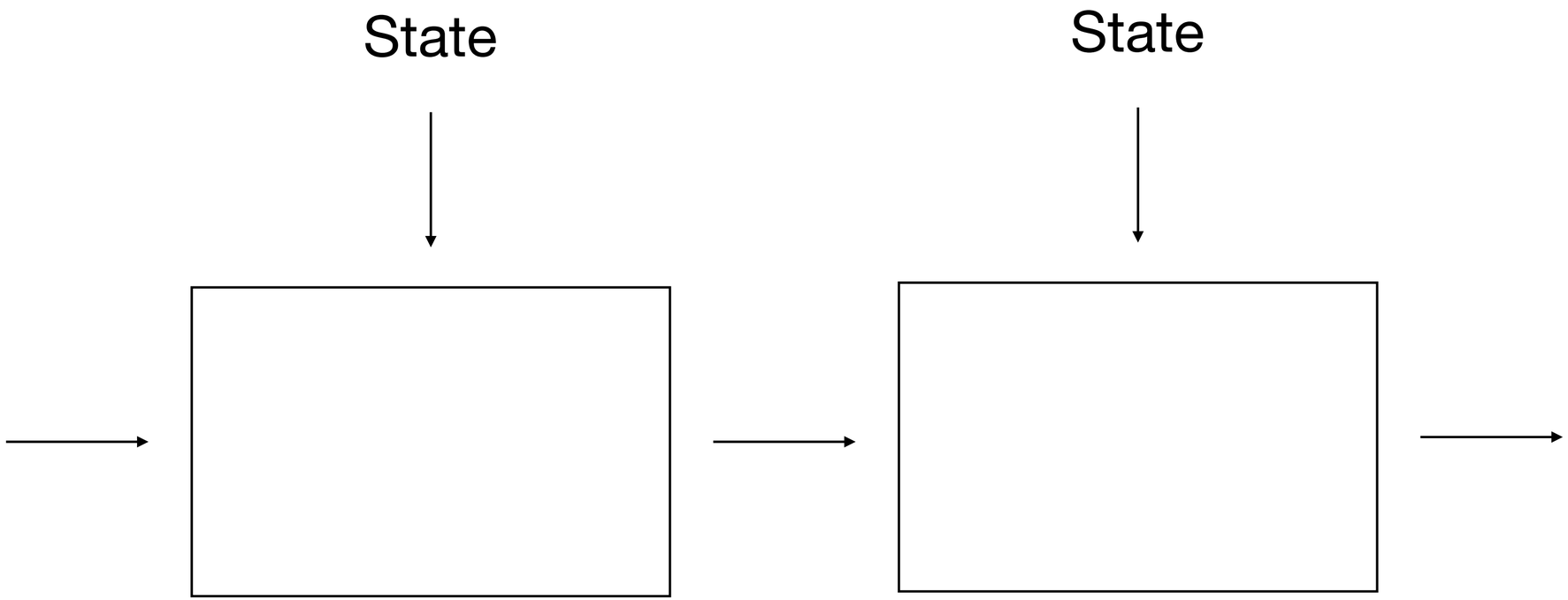
$$\text{State} \times \text{Input} \to \text{State} \times \text{response}$$

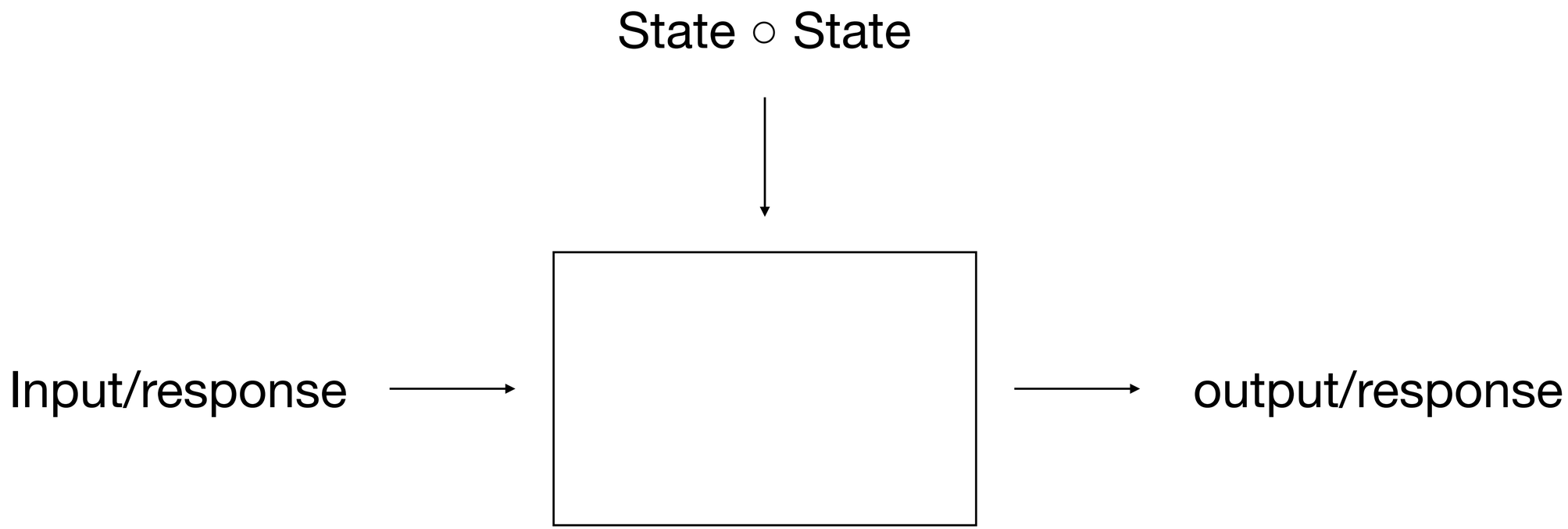

State1
State2

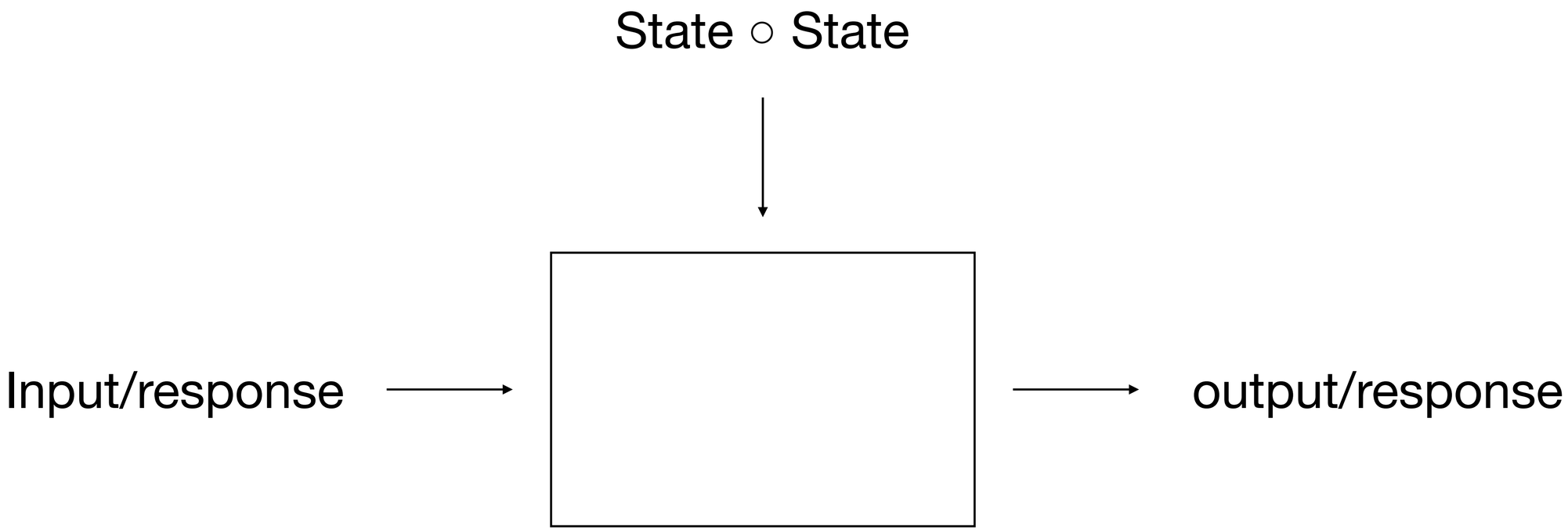

State1 ⊗ State2
Input/response
output/response


State
Input/response
Compiler
output/response

Addressing concurrency


State
State


State ○ State
Input/response
output/response



(Incremental response?)

Implementing compilers

Tactics for architecture

We've come full circle

- Less-pointless lenses
- Container morphisms as tactics

Demo

Thank you