Purely functional distributed programming

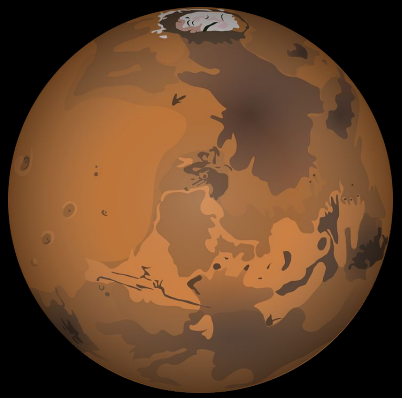
for collaborative applications.


@aidyIns

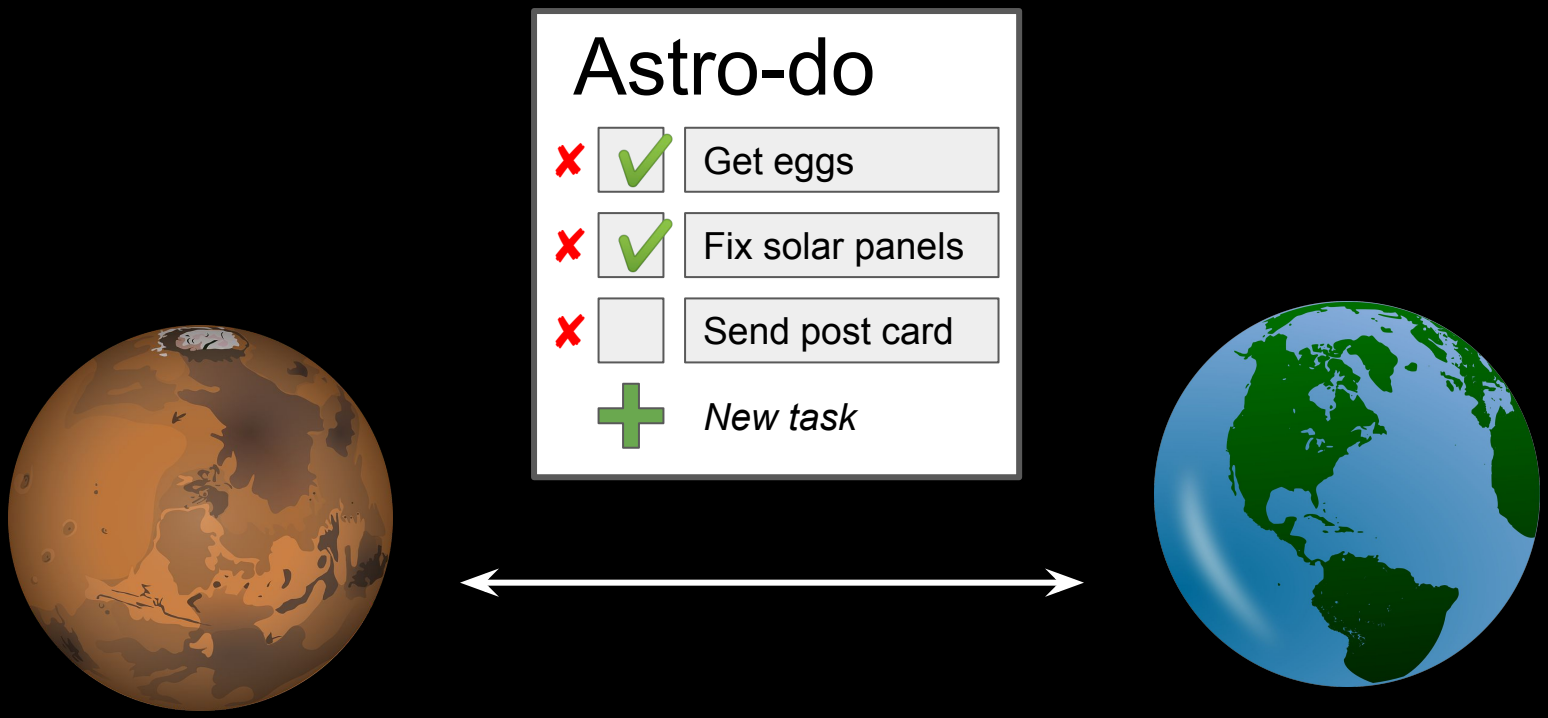

@parenthetical

Adriaan Leijnse
Universidade NOVA de Lisboa

Work supported by Protocol Labs FCT FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA NOVALINCS



Why is writing this application hard?



Not enough functional programming!

Please ask questions!

ping

```
client(Pid) ->
  {server, Pid} ! {ping,self()};
  receive
    pong ->
      client(Pid)
  end.
```

```
server()
  receive
    {ping, Pid} ->
      Pid ! pong;
      server()
  end.
```

ping

```
client(Pid) ->
  {server, Pid} ! {ping,self()};
  receive
    pong ->
      client(Pid)
  end.
```

Can we do better than ickiness?

impure

ickiness

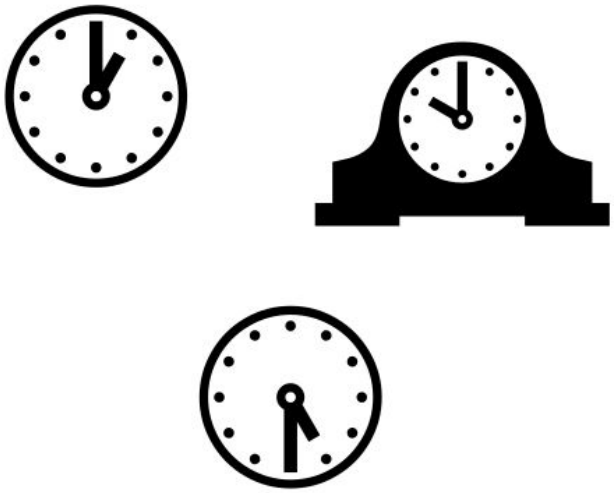
```
server()
  receive
    {ping, Pid} ->
      Pid ! pong;
      server()
  end.
```

What *is* a distributed program?

"places"

Space

"moments"



Time

$$\texttt{Spacetime} \longrightarrow a$$

```
temperature ::
      Spacetime → Kelvin
```

Relativistic Functional Reactive Programming

$$\texttt{Behavior}\ a = \texttt{Spacetime} \longrightarrow a$$

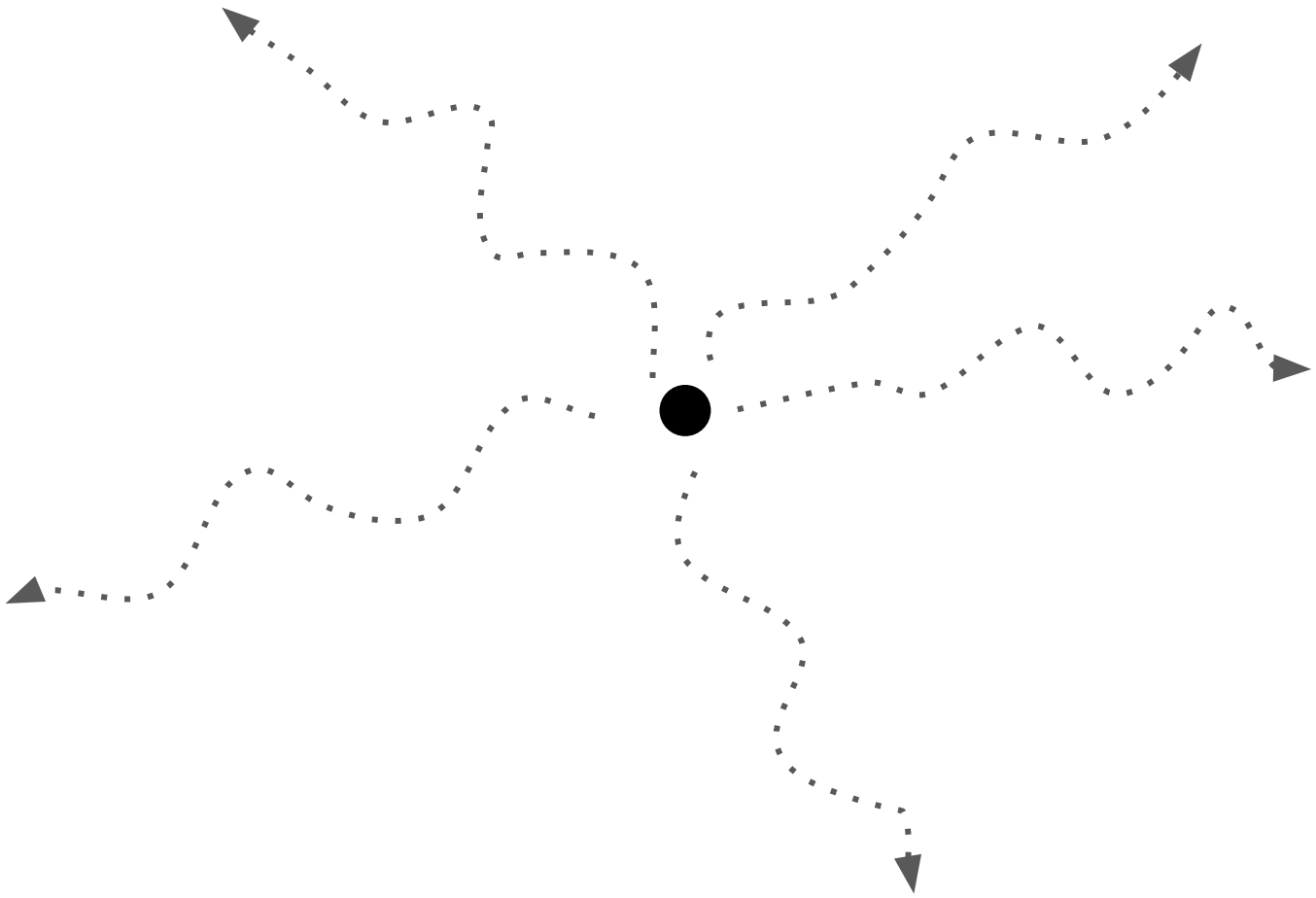
$$\texttt{Event}\ a = \texttt{Map Spacetime}\ a$$

Relativistic Functional Reactive Programming

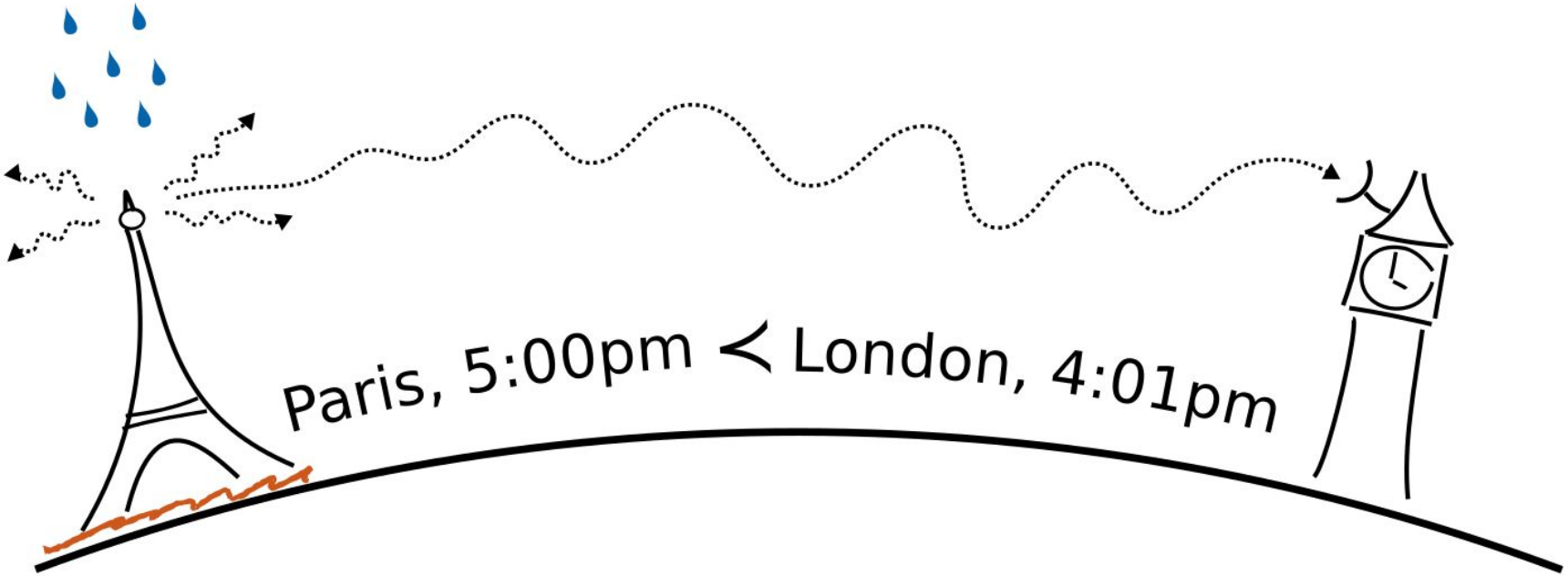
Conal Elliott & Paul Hudak, 1997

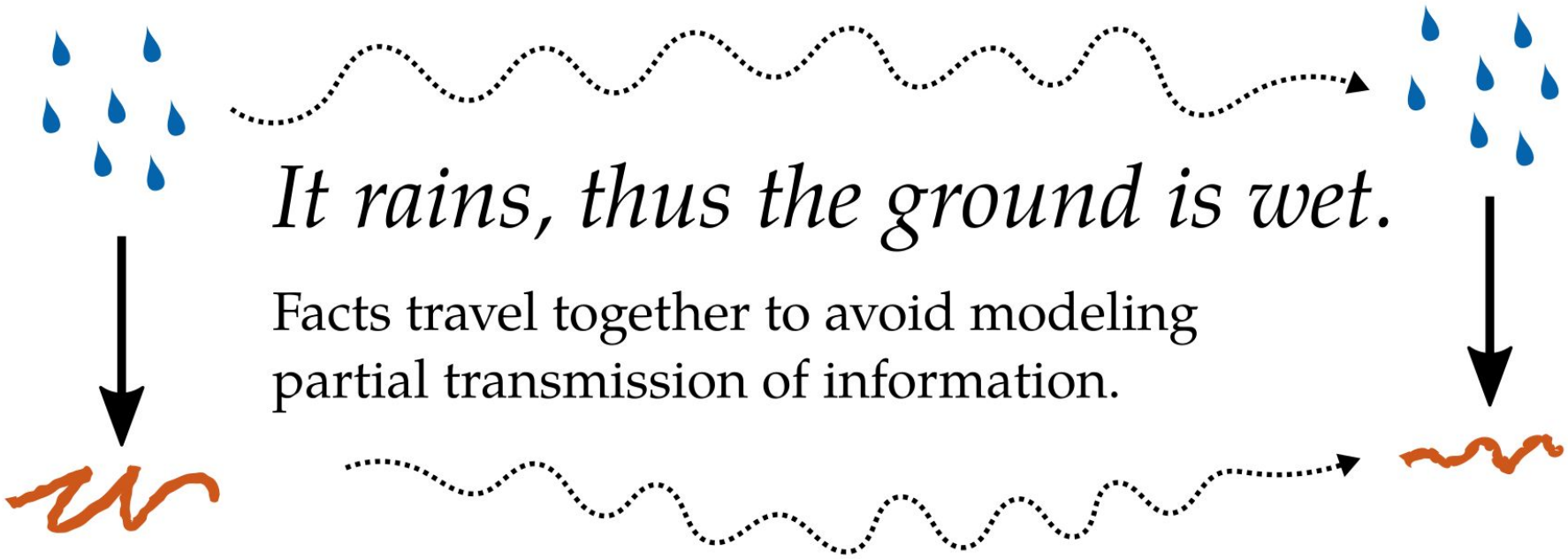
```
Behavior a = Spacetime → a

Event a = Map Spacetime a
```

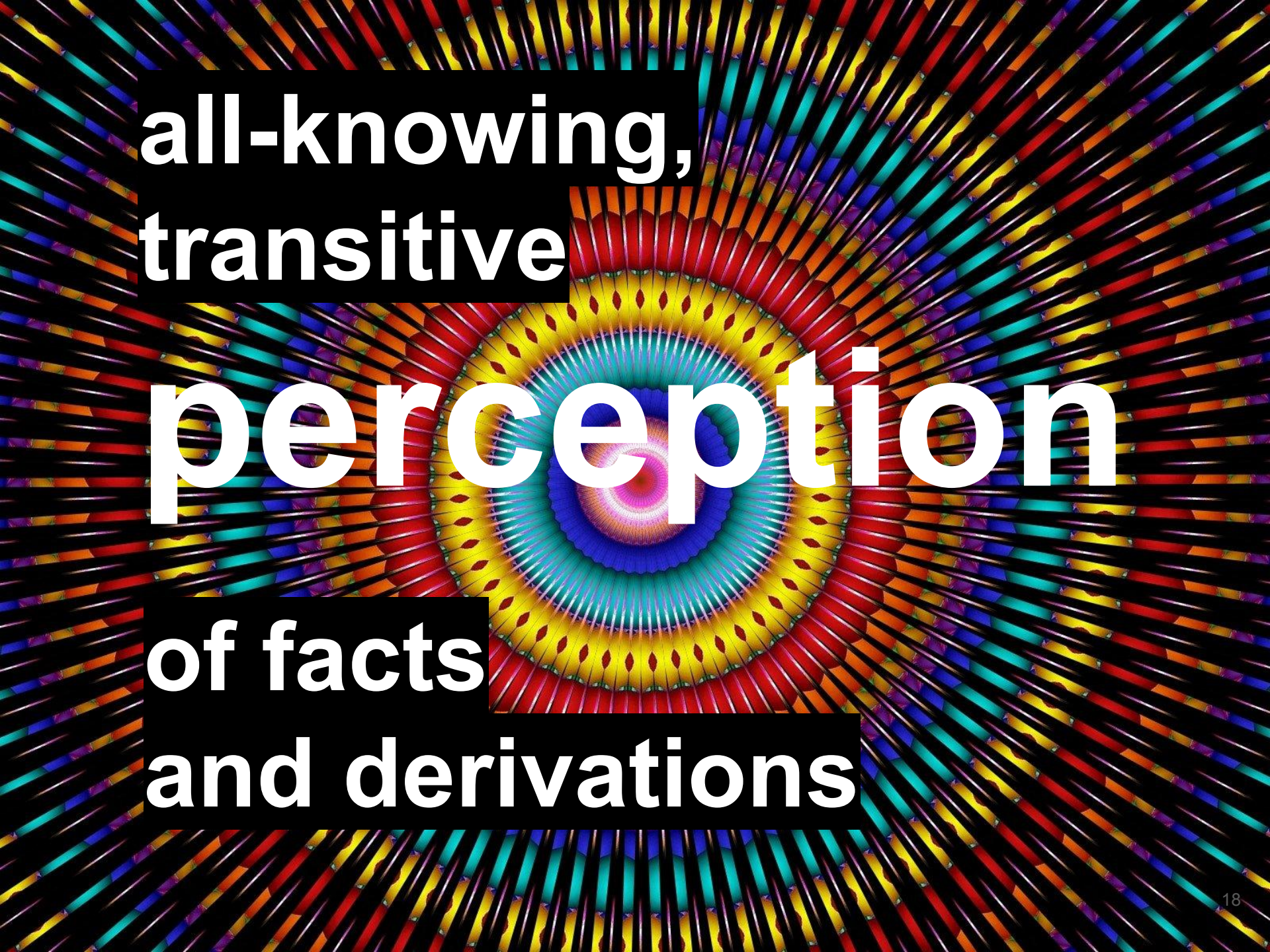
What about communication?



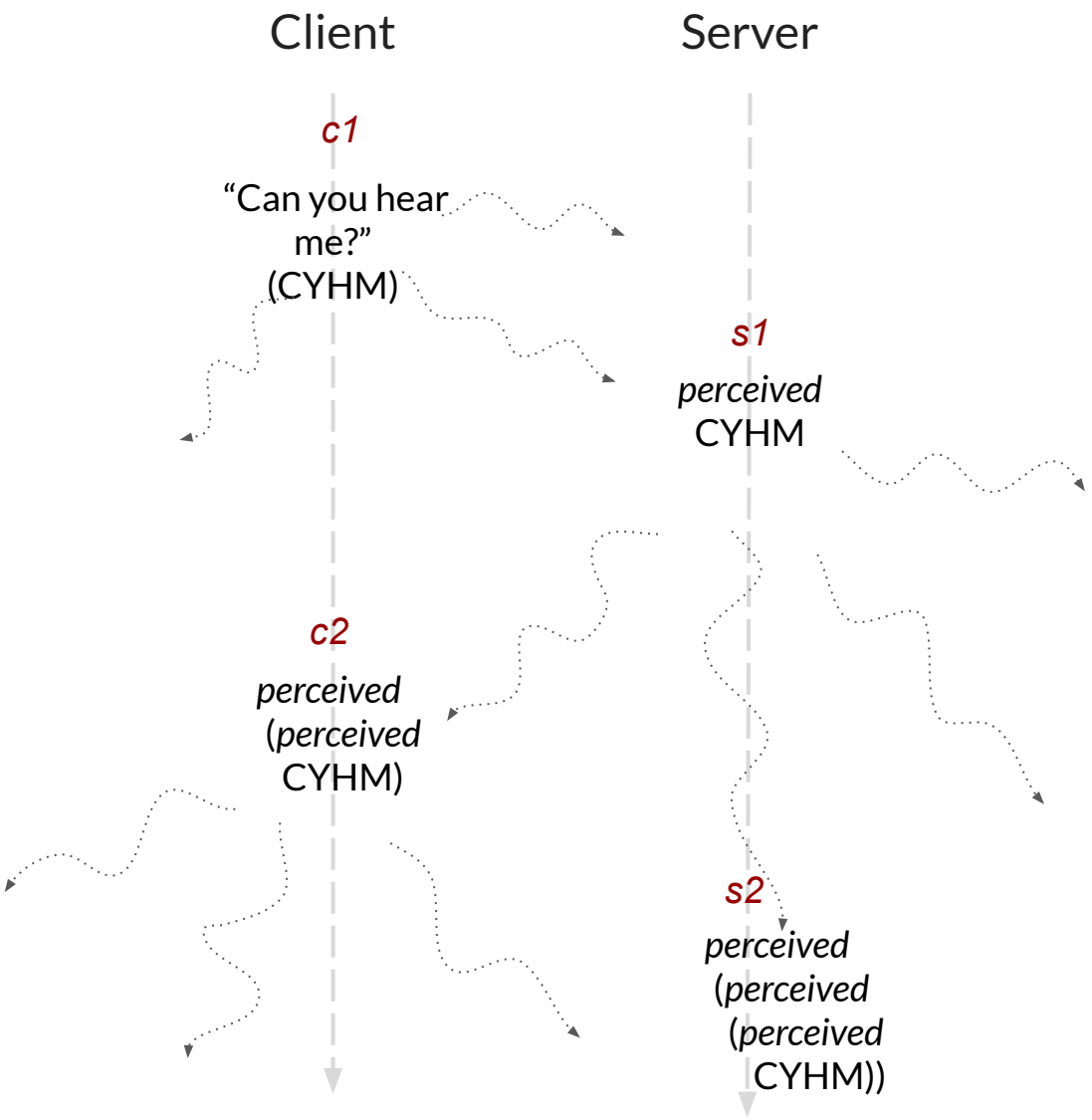

perception


Paris, 5:00pm ≺ London, 4:01pm


It rains, thus the ground is wet.
Facts travel together to avoid modeling
partial transmission of information.


all-knowing,
transitive
perception
of facts
and derivations

ping *through the lens of perception*

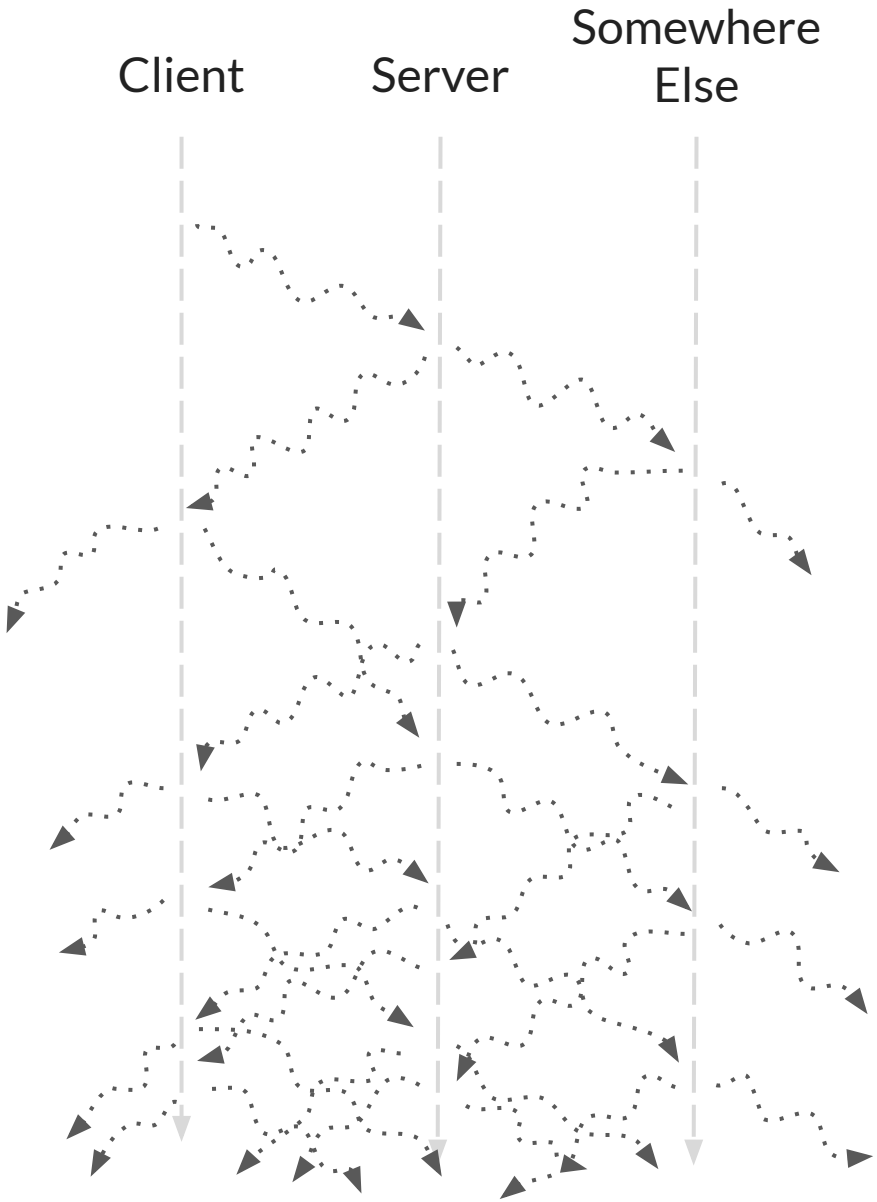



```
ping : Event ()
ping =
  perceptionsAt client
    (perceptionsAt server
       (canYouHearMe <> ping)
```

```
canYouHearMe : Event ()
canYouHearMe = { c1 }
```

```
ping : Event ()
ping = { c2, c3, c4, … }
```

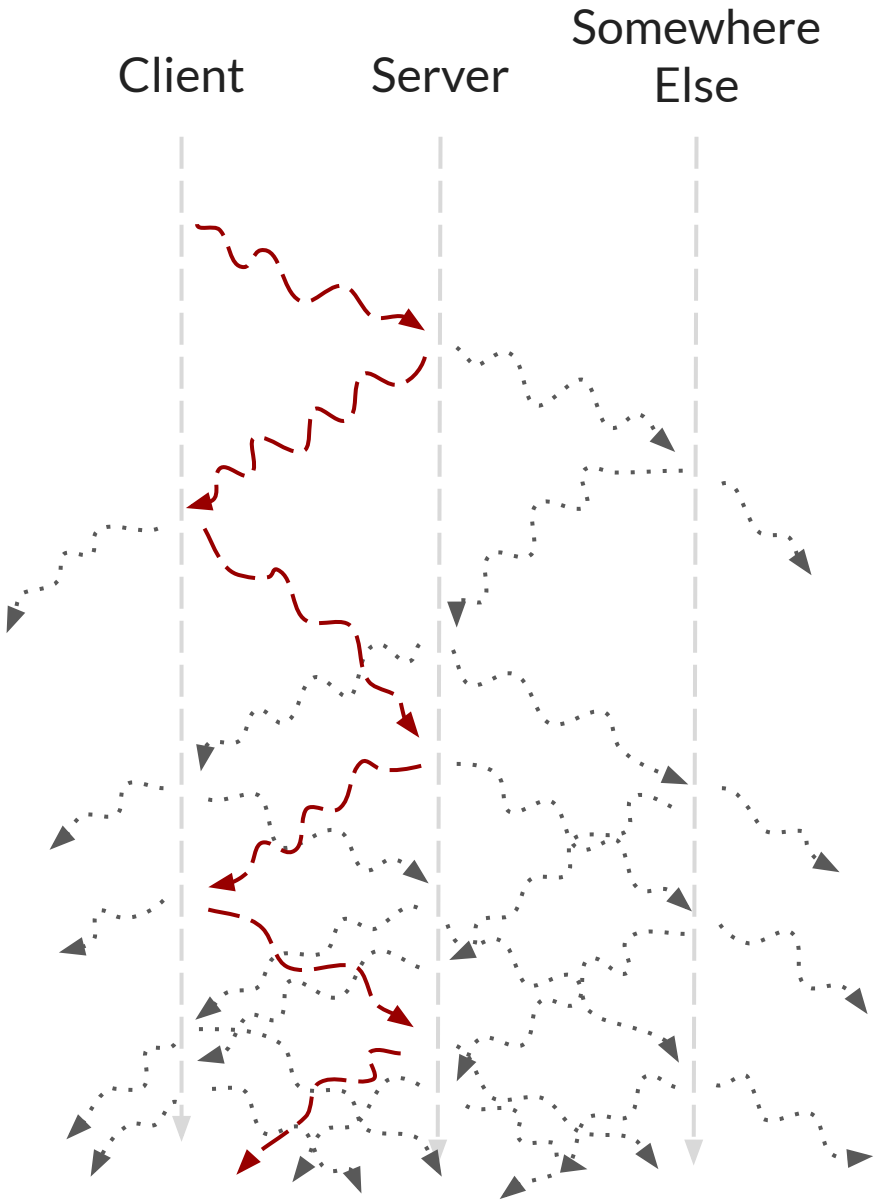
Location-free programming




```
pingish : Event () → Event ()
pingish start =
  perceptions
    (perceptions
       (start <> pingish)
```

```
ping : Event () → Event ()
ping start =
  at client
    (restrict [client, server]
      (pingish start))
```

Location-free programming

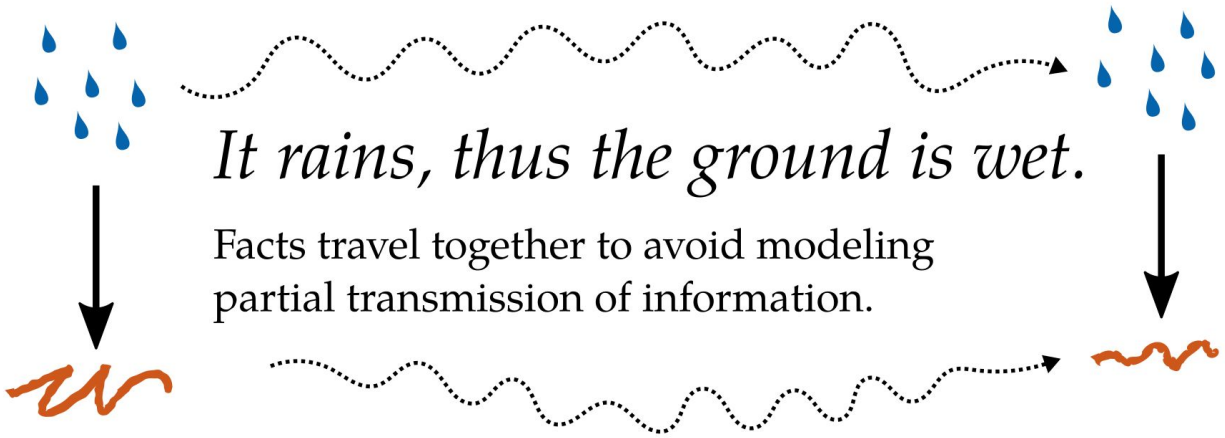



```
pingish : Event () → Event ()
pingish start =
  perceptions
    (perceptions
       (start <> pingish)
```

```
ping : Event () → Event ()
ping start =
  at client
    (restrict [client, server]
      (pingish start))
```

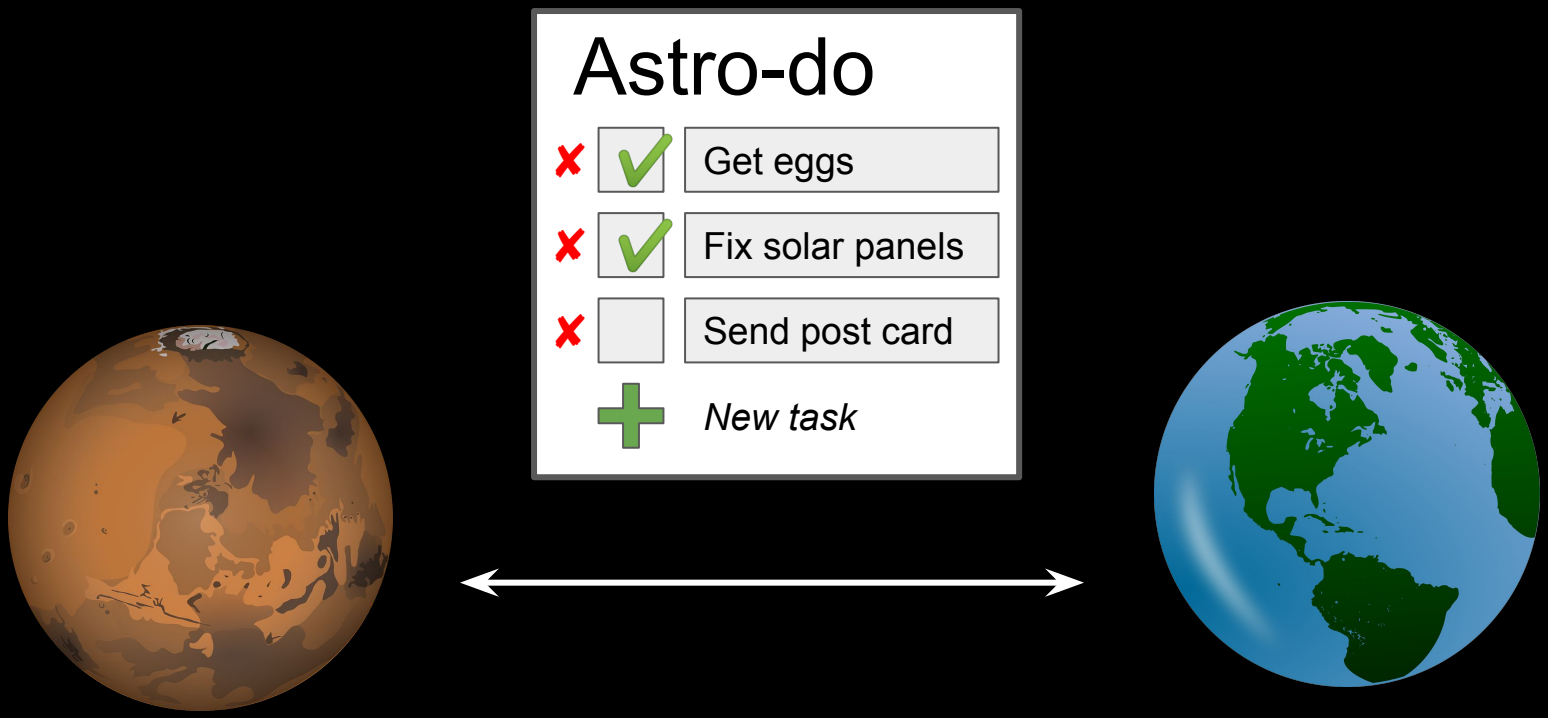
Relativistic Functional Reactive Programming

```
Behavior a = Spacetime → a
Event a = Map Spacetime a
```

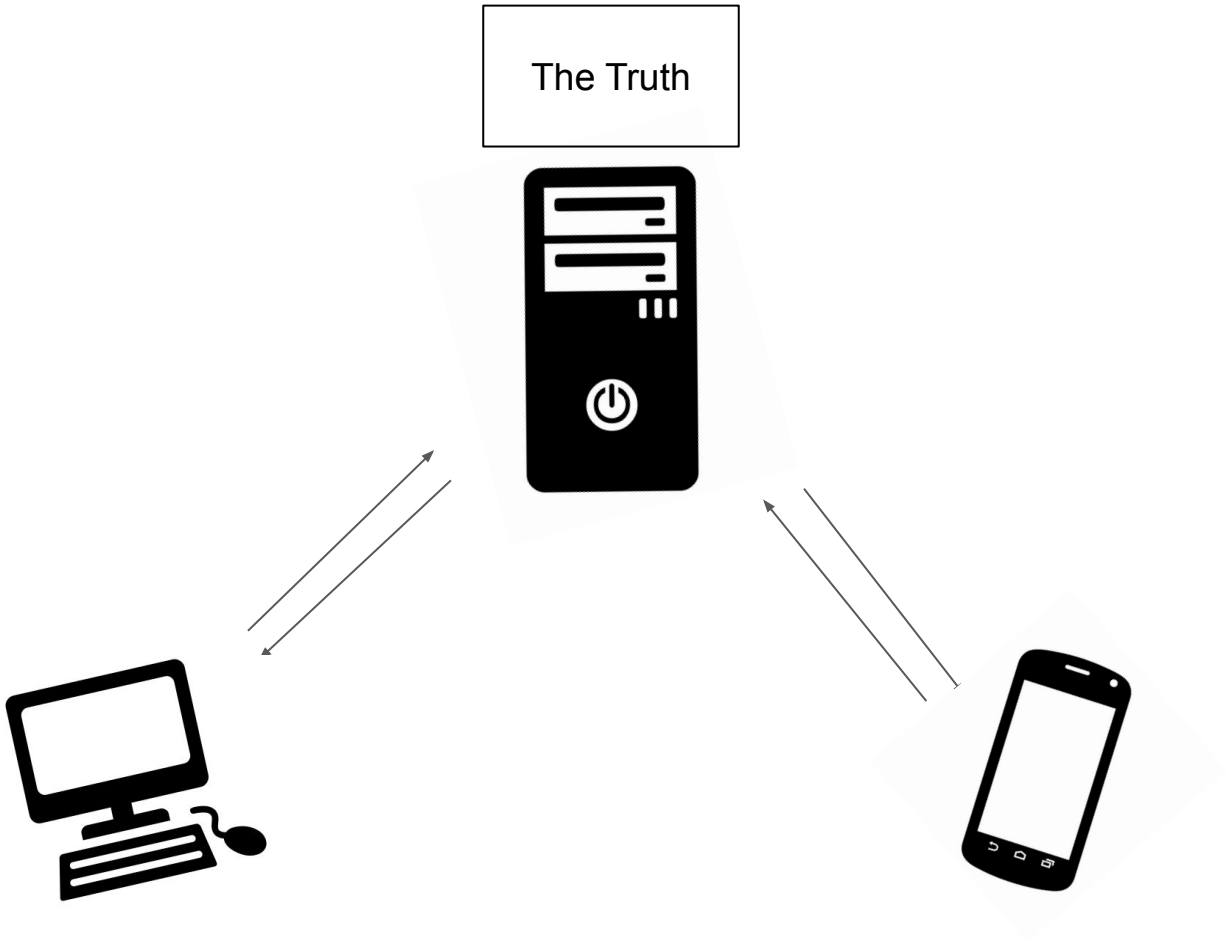


Perception is transitive

$a < b \land b < c \Rightarrow a < c$



4-20 minutes to communicate

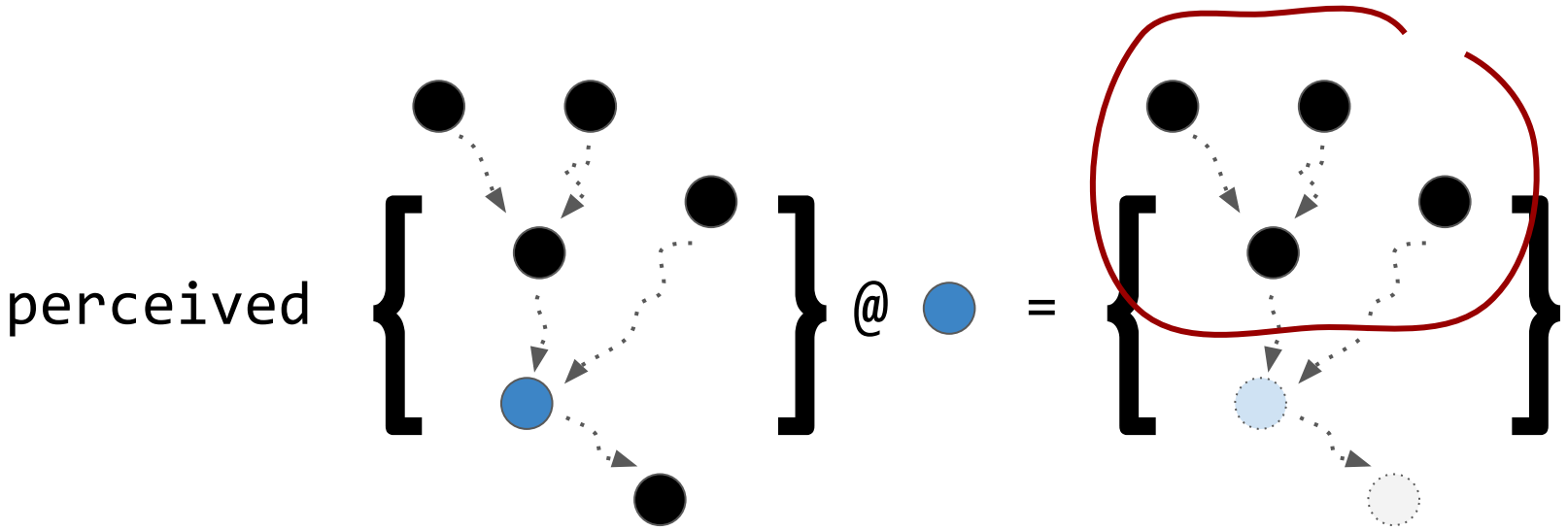

The Truth


重庆
watsons
MIX PARK
Paradise Walk

Strong Eventual Consistency

"predictably derive a result from known operations"

"predictably derive a result from known operations"

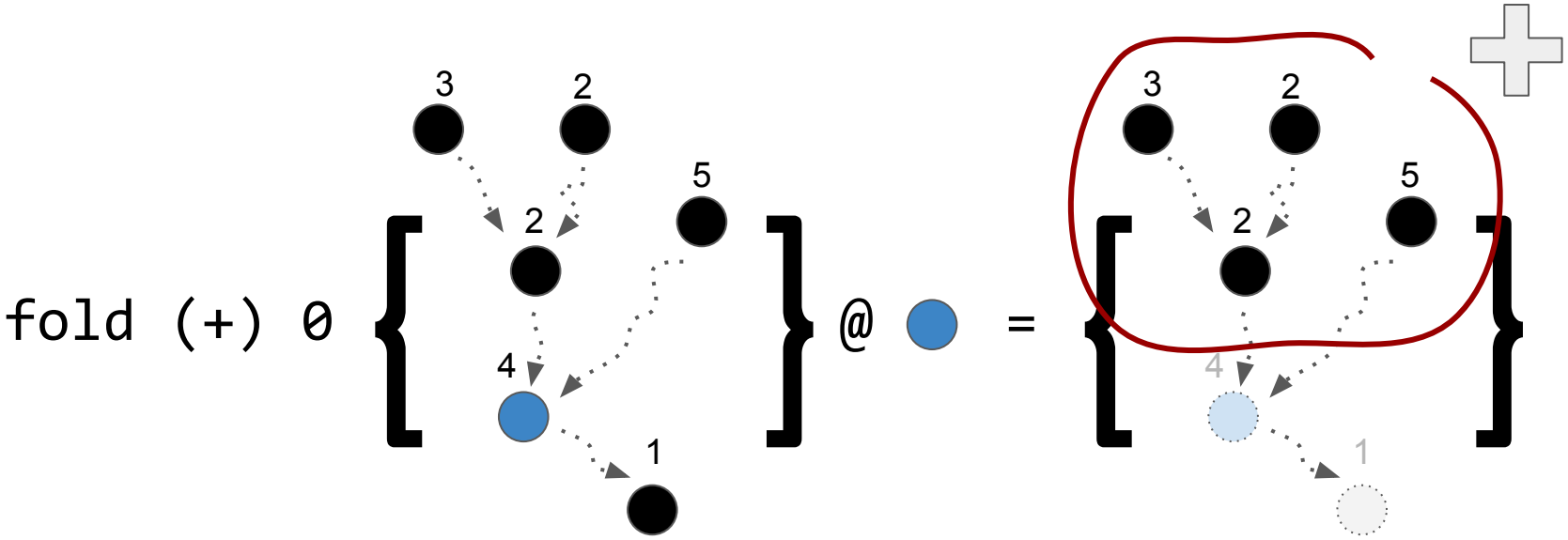


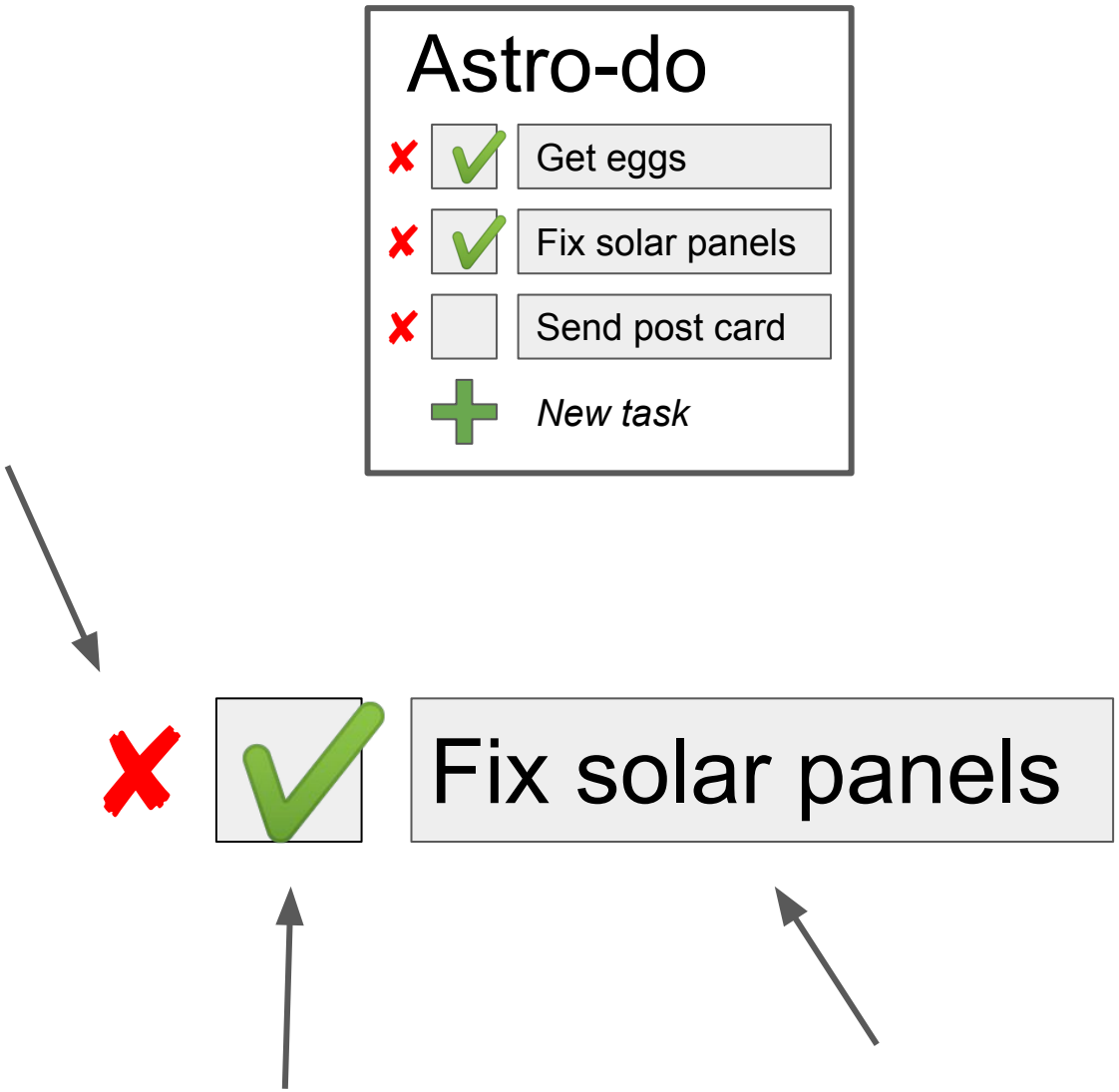
"predictably derive a result from known operations"

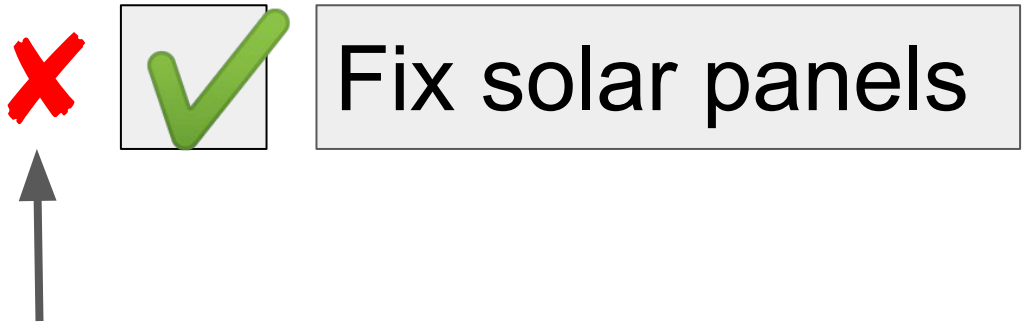
```
fold :: (a → a → a) → a → Event a → Behavior a
```

Commutative & associative combining function

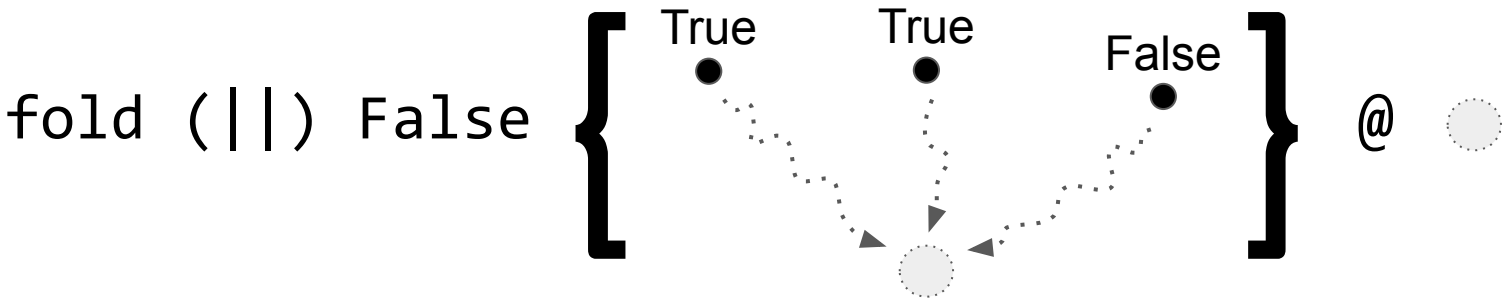
initial value




Astro-do
Get eggs
Fix solar panels
Send post card
New task
Fix solar panels


Fix solar panels

Is "Fix solar panels" deleted? ✘




```
= True || True || False
= True
```

Conflict-Free Replicated Data Types

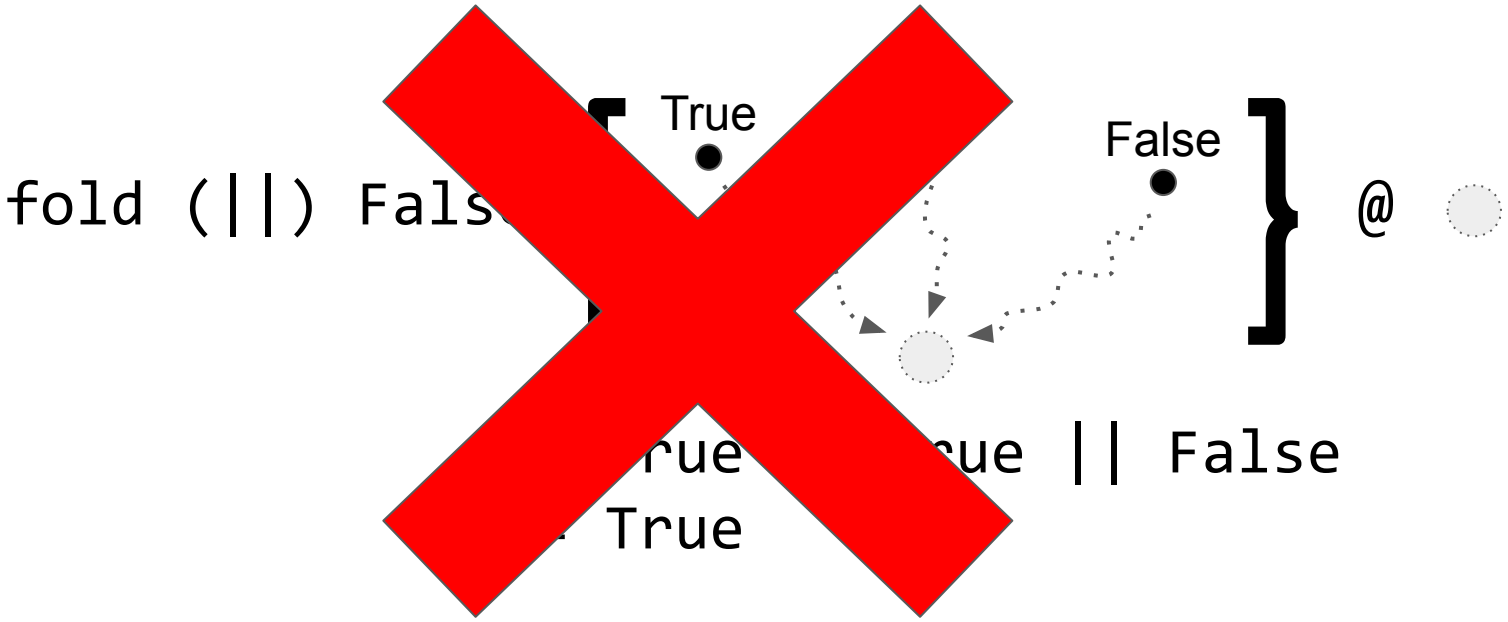
```
type Crdt operations values =
    Event operations → Behavior values
```

Enable-Once Flag CRDT

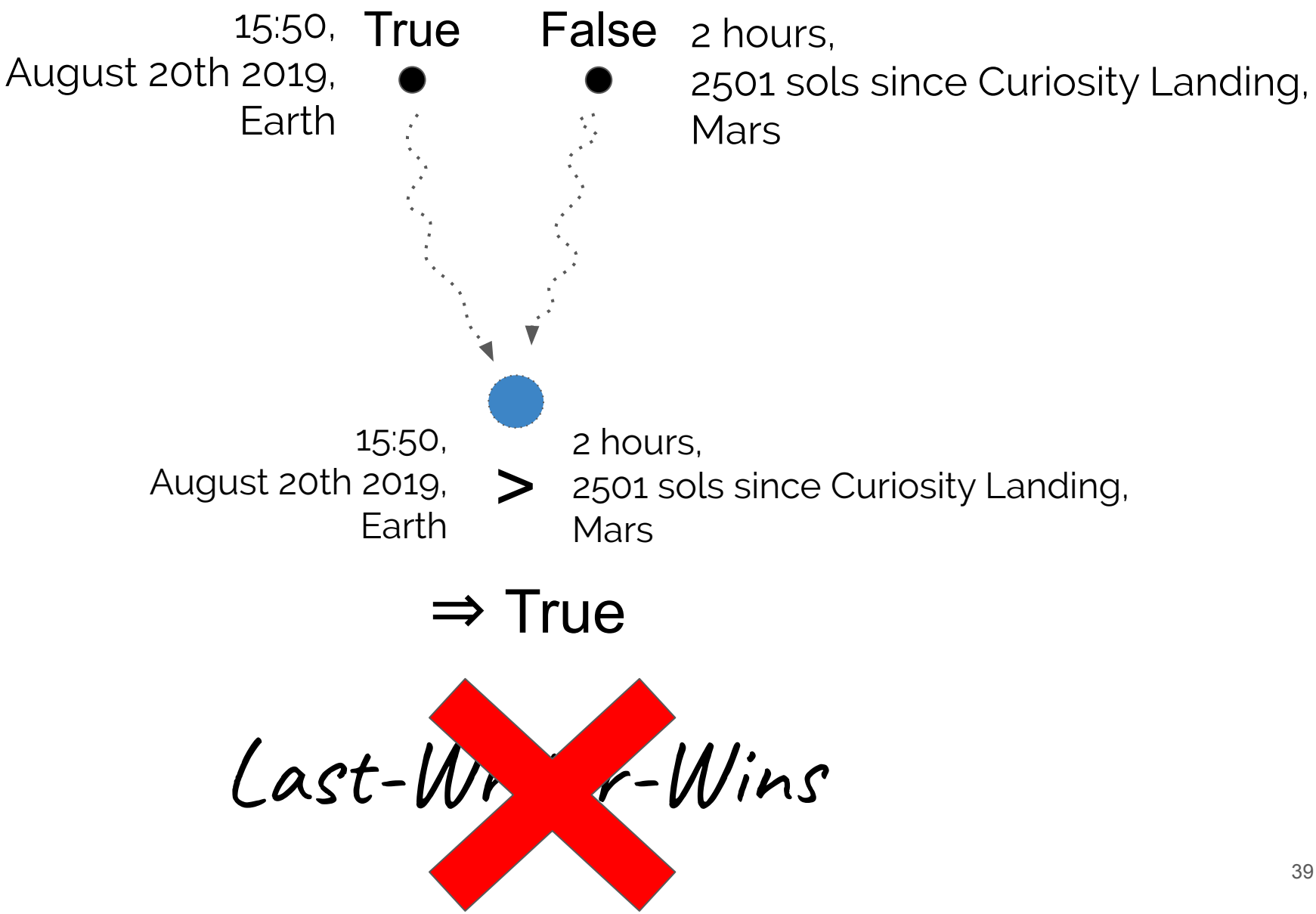
```
eoflag :: Crdt Bool Bool
eoflag = fold (||) False
```

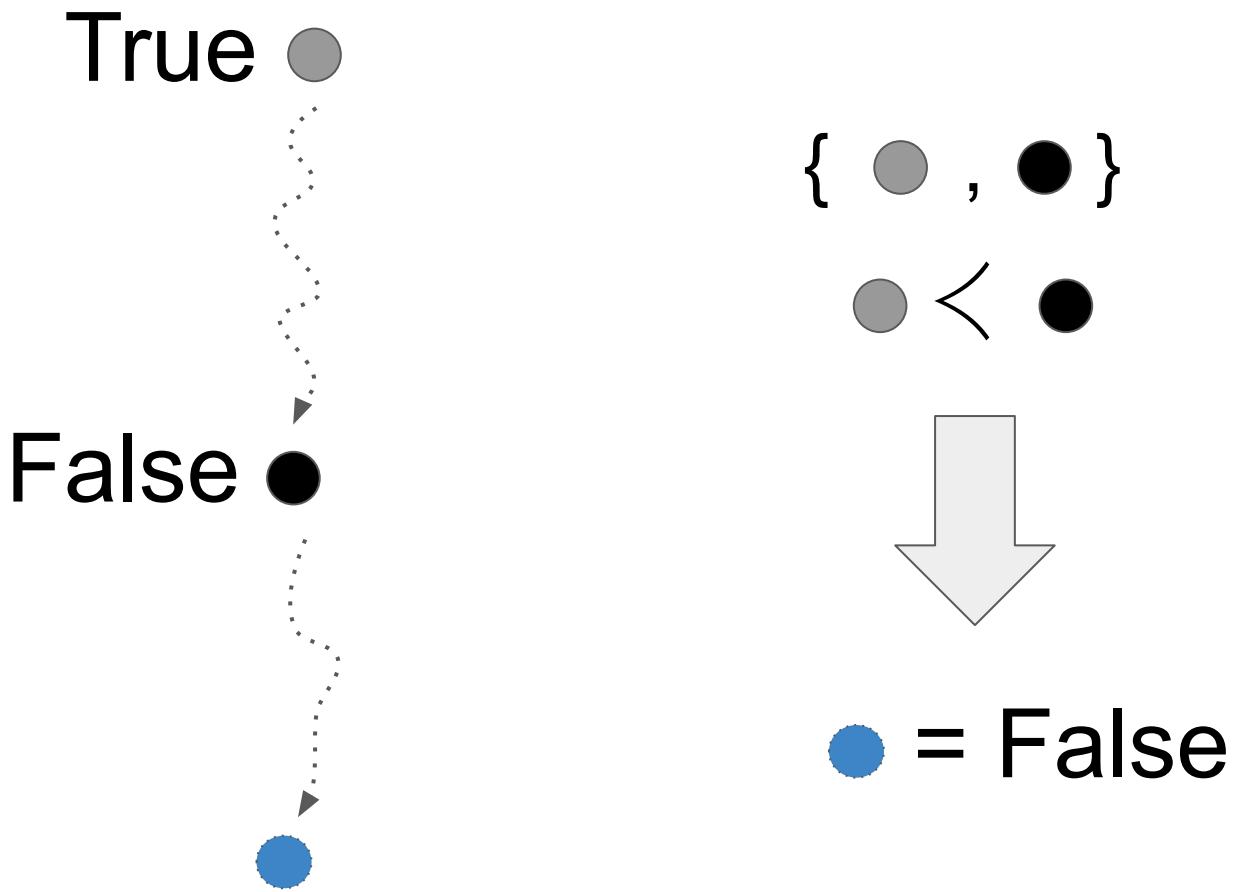

Fix solar panels

Do the solar panels need fixing?



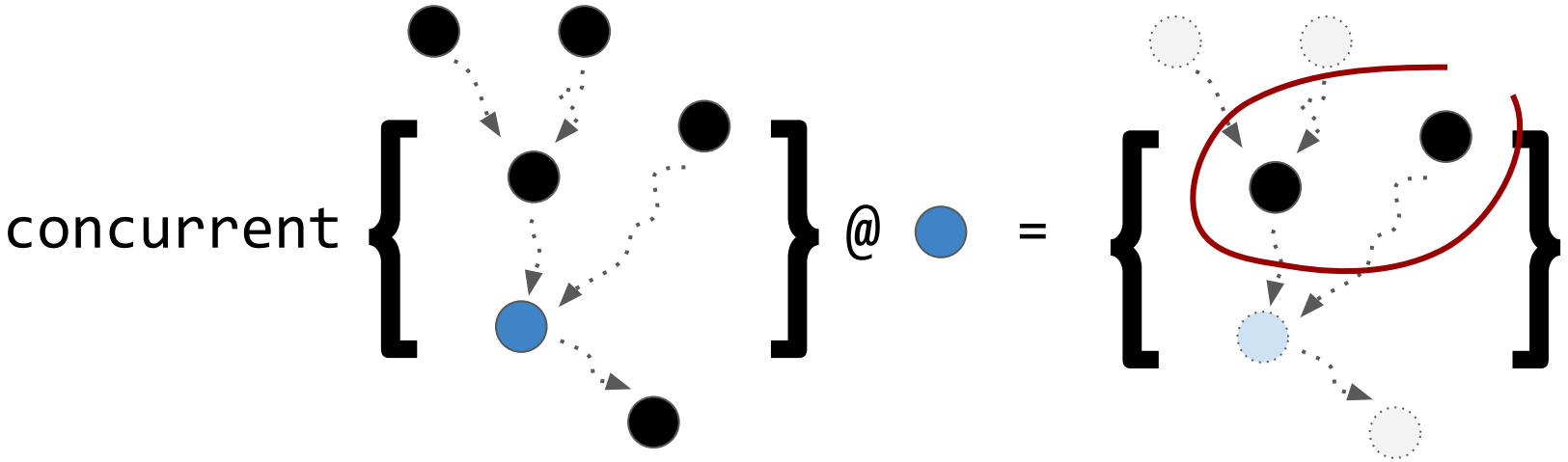
"Overriding" decisions?

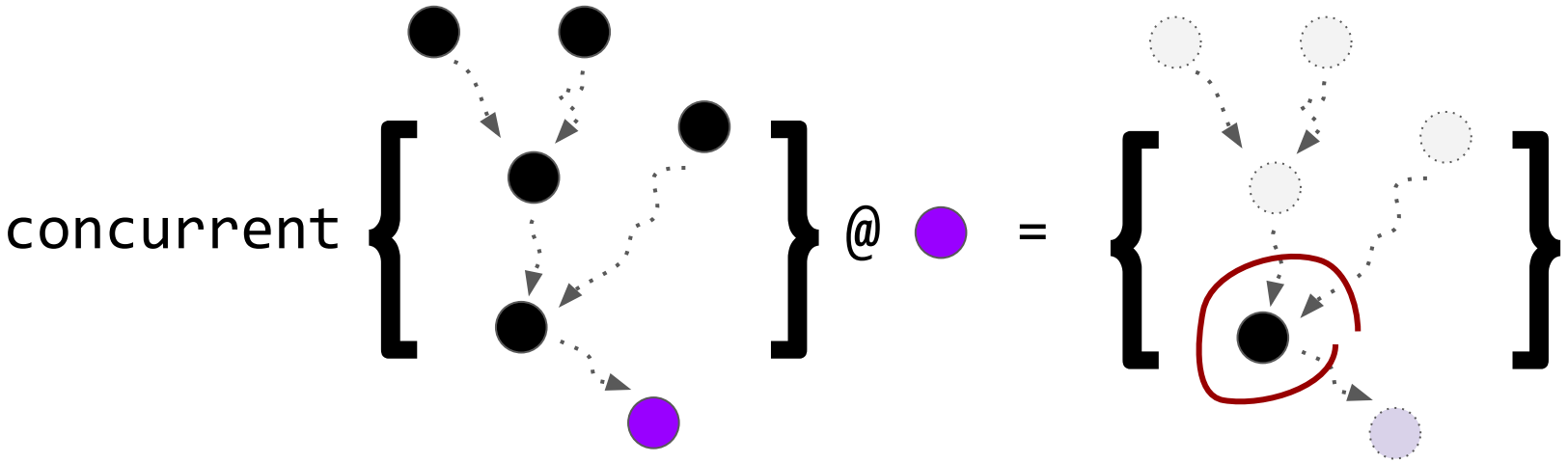

15:50,
August 20th 2019,
Earth
True
False
2 hours,
2501 sols since Curiosity Landing,
Mars
15:50,
August 20th 2019,
Earth
>
2 hours,
2501 sols since Curiosity Landing,
Mars
⇒ True
Last-Wr r-Wins



Concurrent events

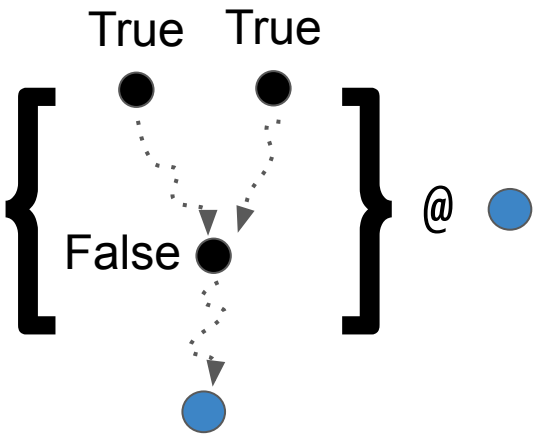
```
concurrent :: Event a → Behavior (Event a)
```





Do the solar panels need fixing?

```
(eoflag =<< concurrent)

= False
```



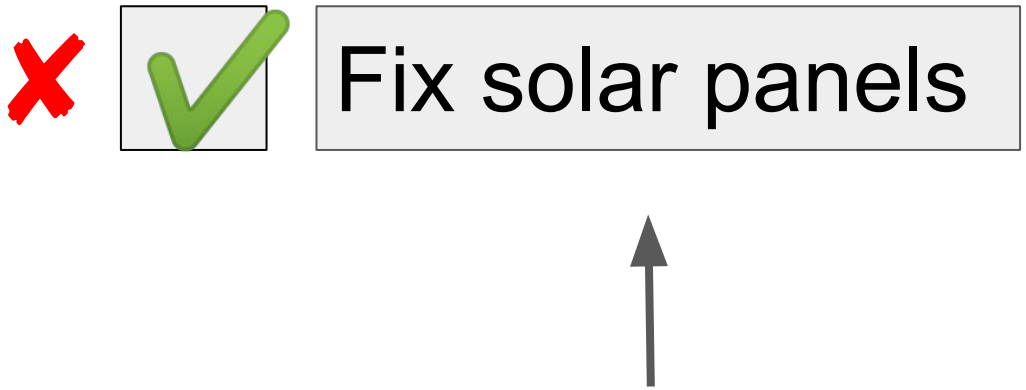

```
(=<<) :: Behavior a → (a → Behavior b) → Behavior b
```

Enable-Once Flag CRDT

```
eoflag :: Crdt Bool Bool
eoflag = fold (||) False
```

Enable-Wins Flag CRDT

```
ewflag :: Crdt Bool Bool
ewflag = (eoflag =<< concurrent)
```


Fix solar panels

Fix solar panels

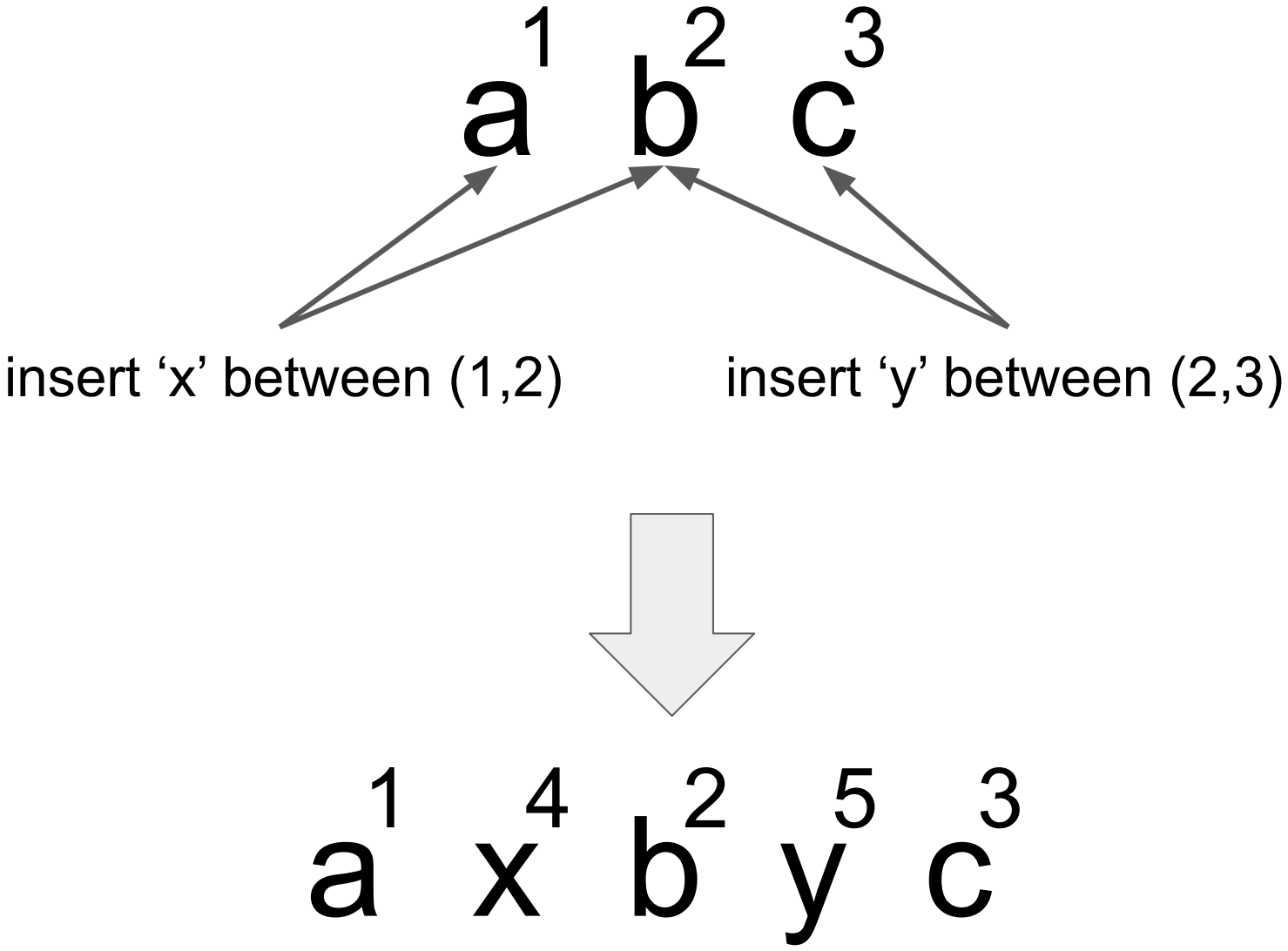
1. Insert ‘x’ @ 10
2. Insert ‘y’ @ 10

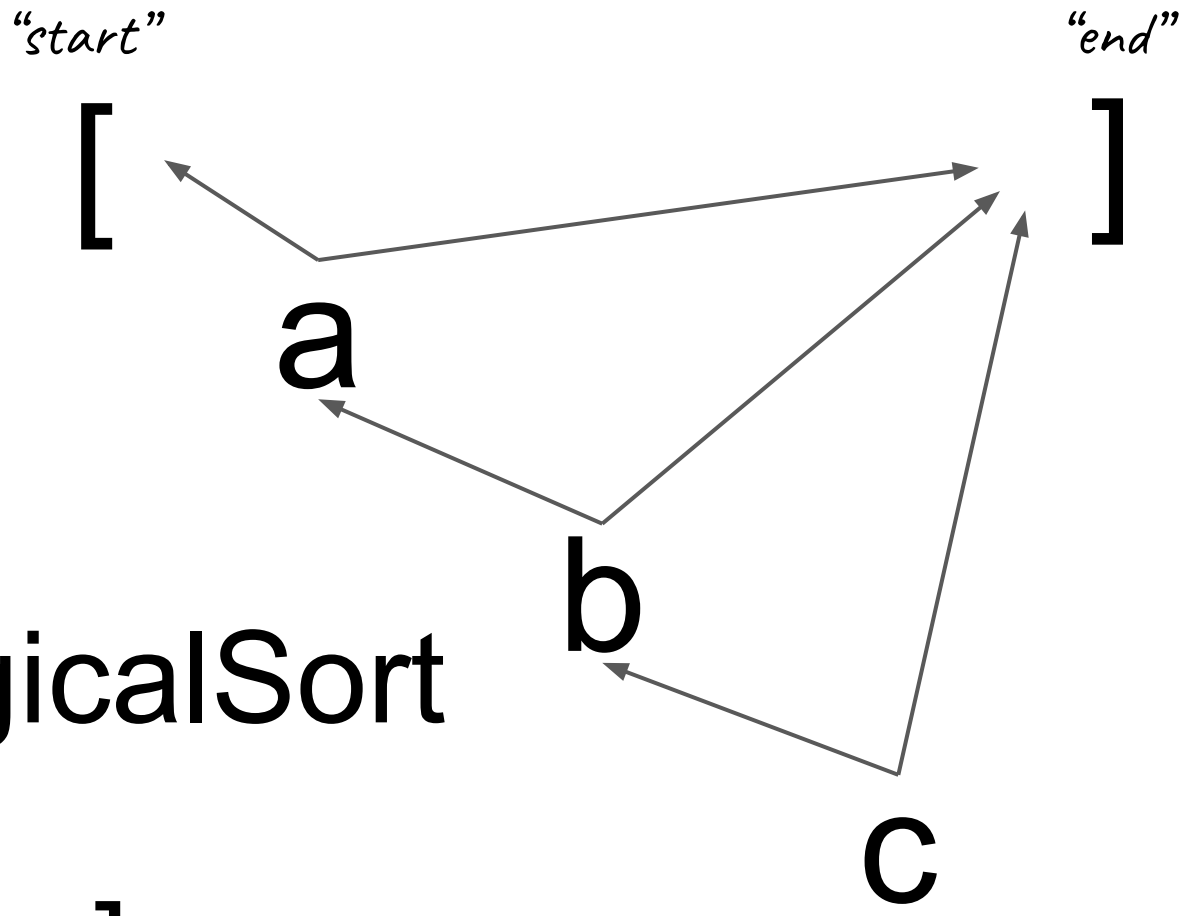
≠

1. Insert ‘y’ @ 10
2. Insert ‘x’ @ 10

Fix solar xypanels

Fix solar yxpanels

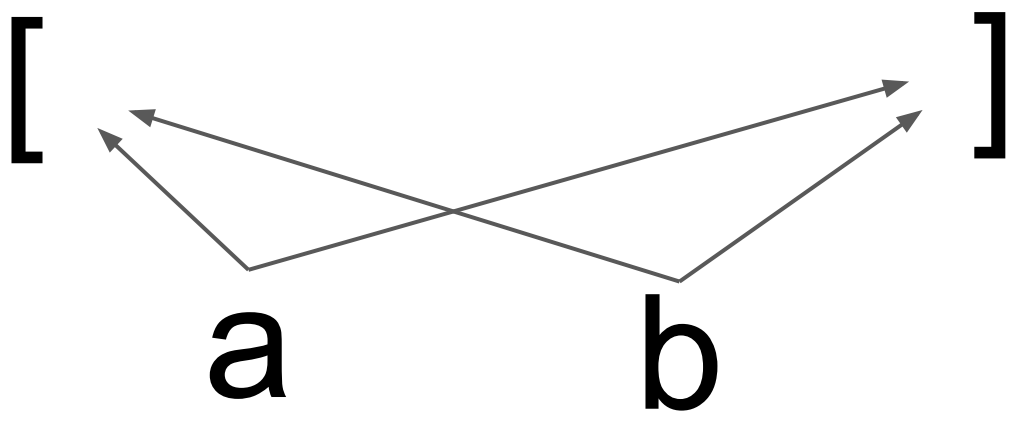

a 1 b 2 c 3
insert ‘x’ between (1,2)
insert ‘y’ between (2,3)
a 1 x 4 b 2 y 5 c 3



topologicalSort

= [a b c]

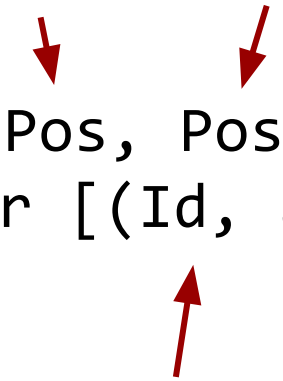
topologicalSort



[a b] *or* [b a] *?*

Tie breaking on identifiers

Let's program sequences!



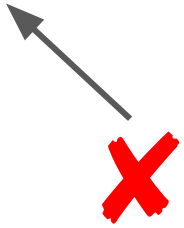
```
sequence :: Event (Pos, Pos, a)
         → Behavior [(Id, a)]


data Pos = Start
         | Middle Id
         | End


type Id = Spacetime


tagWithSpacetime :: Event a
                 → Event (Spacetime, a)
```

```
sequence :: Event (Pos, Pos, a)
         → Behavior [(Id, a)]
sequence e =
  let idsE = tagWithSpacetime e
      graphB = fold Set.union Set.empty
                    (mapE Set.singleton idsE)
  in mapB topologicalSort graphB


topologicalSort :: Set (Id, (Pos, Pos, a))
                → [(Id, a)]

mapE :: (a → b) → Event a → Event b

mapB :: (a → b) → Behavior a → Behavior b
```

Sequence with deletion

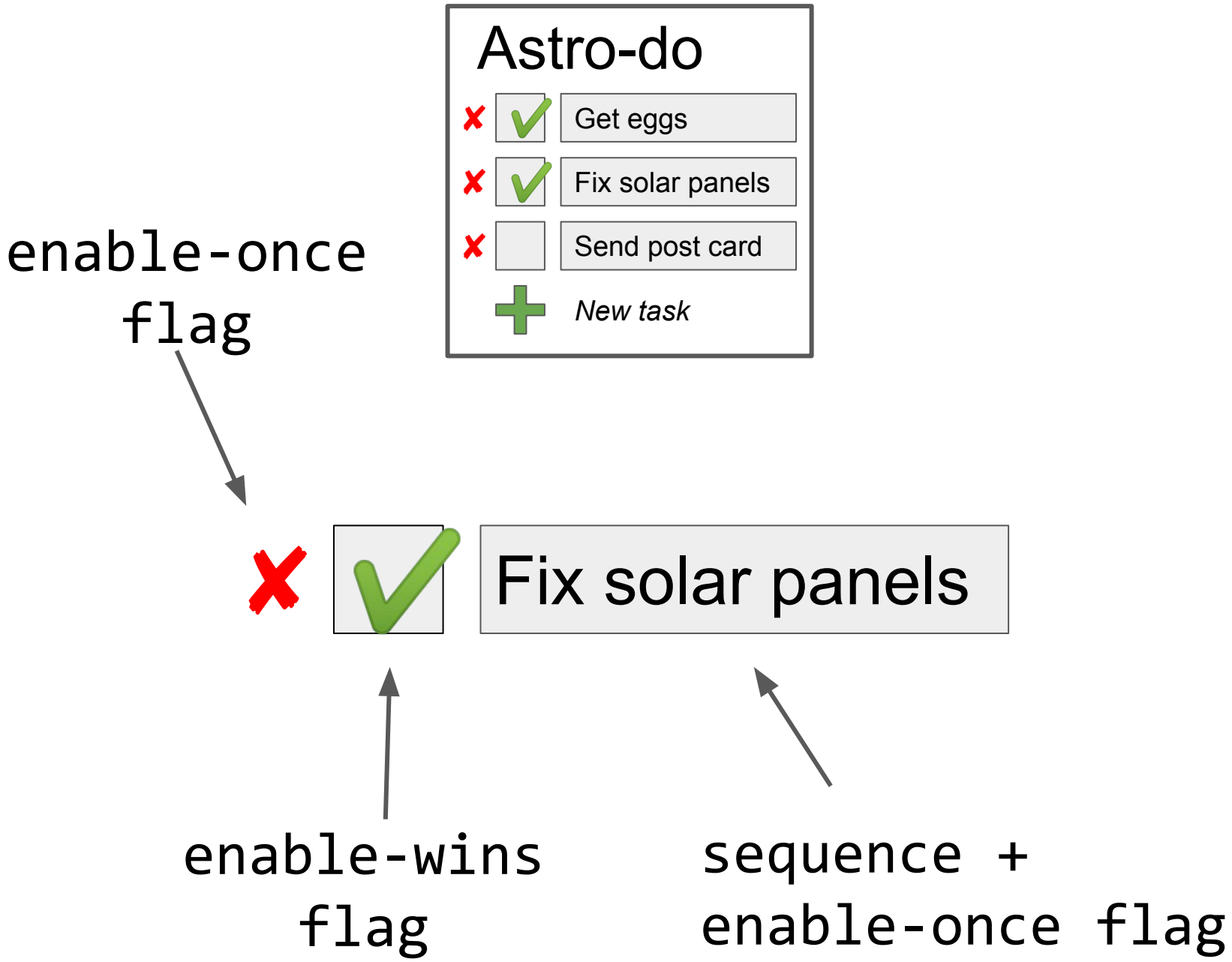
```
sequence :: Event (Pos, Pos, a)
         → Behavior [(Id, a)]
```

```
a = Behavior (Bool, a')
```



```
Behavior [(Id, Behavior Bool a)]
```

```
filter + (=<<)  ⇝  Behavior [(Id, a)]
```


Astro-do
Get eggs
Fix solar panels
Send post card
New task
enable-once flag
Fix solar panels
enable-wins flag
sequence + enable-once flag

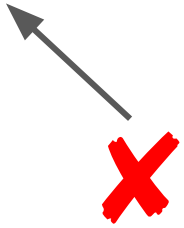
WIP
Eventually Consistent
RFRP

- Library UX for GUI
- Extending Reflex!
  https://github.com/reflex-frp/reflex
  => Production quality P2P apps
- Hard problems (PhD)

Sequence with deletion

```
sequence :: Event (Pos, Pos, a)
         → Behavior [(Id, a)]
```

```
a = Behavior (Bool, a')
```

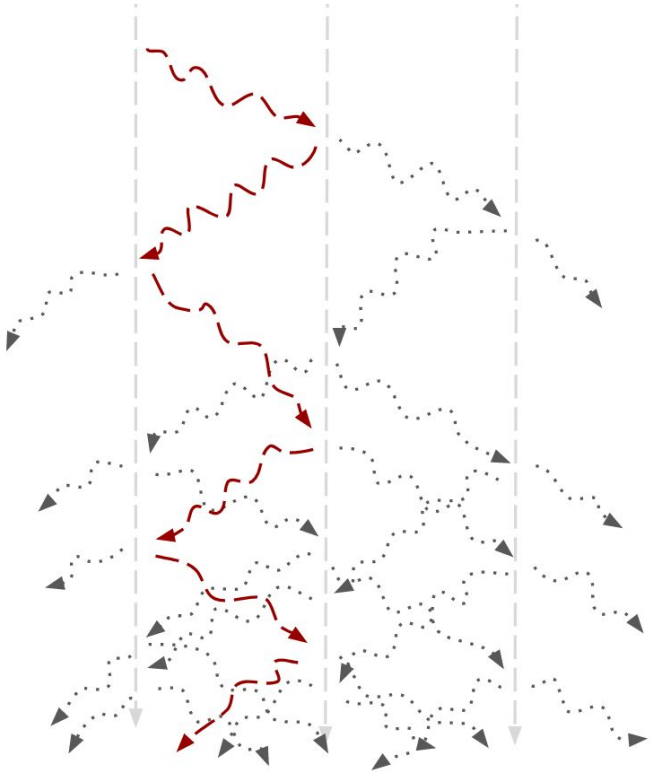


```
Behavior [(Id, Behavior Bool a)]
```

```
filter + (=<<)  ⇝  Behavior [(Id, a)]
```

Relativistic Functional Reactive Programming

```
Behavior a = Spacetime → a
Event a = Map Spacetime a
```



@aidyIns

@parenthetical

```
fold :: (a → a → a)
     → a
     → Event a
     → Behavior a
```

(Commutative & associative)

```
concurrent :: Event a
           → Behavior (Event a)
```

Peer-to-peer apps for free!

Probabilistic

Relativistic

Functional Reactive Programming

```
Behavior a → (Spacetime, Probability, a)
```

```
temperature :: Spacetime → Kelvin
```


Astro-do
Get eggs
Fix solar panels
Send post card
New task


Get eggs


Fix solar panels