Toward an Algebraic Theory of Systems

Christian Matt, Ueli Maurer, Christopher Portmann, Renato Renner, Björn Tackmann

$$\mathcal{S}_0 := \left\{\gamma_{\{r_1^2,e^2\}}(\gamma_{\{s_0^2,e^1\}}(\gamma_{\{s_0^1,r_1^1\}}(s_0 \parallel r_1 \parallel e))) \mid e \in \mathfrak{S} \wedge \lambda(e) \cap \lambda(s_0) = \emptyset = \lambda(e) \cap \lambda(r_1) \wedge e^1, e^2 \in \lambda(e) \wedge e^1 \neq e^2\right\},$$

$$\mathcal{S}_1 := \left\{\gamma_{\{e^2,r_2^2\}}(\gamma_{\{s_1^2,r_2^1\}}(\gamma_{\{s_1^1,e^1\}}(s_1 \parallel e \parallel r_2))) \mid e \in \mathfrak{S} \wedge \lambda(e) \cap \lambda(s_1) = \emptyset = \lambda(e) \cap \lambda(r_2) \wedge e^1, e^2 \in \lambda(e) \wedge e^1 \neq e^2\right\},$$

$$\mathcal{S}_= := \left\{\gamma_{\{r_1^2,r_2^2\}}(\gamma_{\{e^2,r_2^1\}}(\gamma_{\{e^1,r_1^1\}}(e \parallel r_1 \parallel r_2))) \mid e \in \mathfrak{S} \wedge \lambda(e) \cap \lambda(r_1) = \emptyset = \lambda(e) \cap \lambda(r_2) \wedge e^1, e^2 \in \lambda(e) \wedge e^1 \neq e^2\right\}.$$

Then, $\mathcal{S}_0 \cap \mathcal{S}_1 \cap \mathcal{S}_= \neq \emptyset$.

Proof. Note that $\mathcal{S}_0$ corresponds to the set of systems for all possible e and $a = 0$ in Fig. 2d, $\mathcal{S}_1$ corresponds to Fig. 2c for $a = 1$, and $\mathcal{S}_=$ corresponds to the systems in Fig. 2b. Let

$$t := \gamma_{\{r_1^2,r_2^2\}}(\gamma_{\{s_1^2,r_2^1\}}(\gamma_{\{s_0^1,r_1^1\}}(\gamma_{\{s_0^2,s_1^1\}}(s_0 \parallel s_1 \parallel r_1 \parallel r_2))))$$

be the system in Fig. 2e. By composition-order invariance, we have for $e = \gamma_{\{s_0^2,s_1^1\}}(s_0 \parallel s_1)$,

$$e \parallel r_1 \parallel r_2 = \gamma_{\{s_0^2,s_1^1\}}(s_0 \parallel s_1) \parallel r_1 \parallel r_2 = \gamma_{\{s_0^2,s_1^1\}}(s_0 \parallel s_1 \parallel r_1 \parallel r_2).$$

Hence, we obtain for $e^1 = s_0^1$, and $e^2 = s_1^2$, that $t = \gamma_{\{r_1^2,r_2^2\}}(\gamma_{\{e^2,r_2^1\}}(\gamma_{\{e^1,r_1^1\}}(e \parallel r_1 \parallel r_2))) \in \mathcal{S}_=$. Again using composition-order invariance, we further have for $\hat{e} = \gamma_{\{s_1^2,r_2^1\}}(s_1 \parallel r_2)$, $\hat{e}^1 = s_1^1$, and $\hat{e}^2 = r_2^2$,

$$t = \gamma_{\{r_1^2,r_2^2\}}(\gamma_{\{s_0^2,s_1^1\}}(\gamma_{\{s_0^1,r_1^1\}}(\gamma_{\{s_1^2,r_2^1\}}(s_1 \parallel r_2 \parallel s_0 \parallel r_1)))) = \gamma_{\{r_1^2,\hat{e}^2\}}(\gamma_{\{s_0^2,\hat{e}^1\}}(\gamma_{\{s_0^1,r_1^1\}}(\hat{e} \parallel s_0 \parallel r_1))) \in \mathcal{S}_0.$$

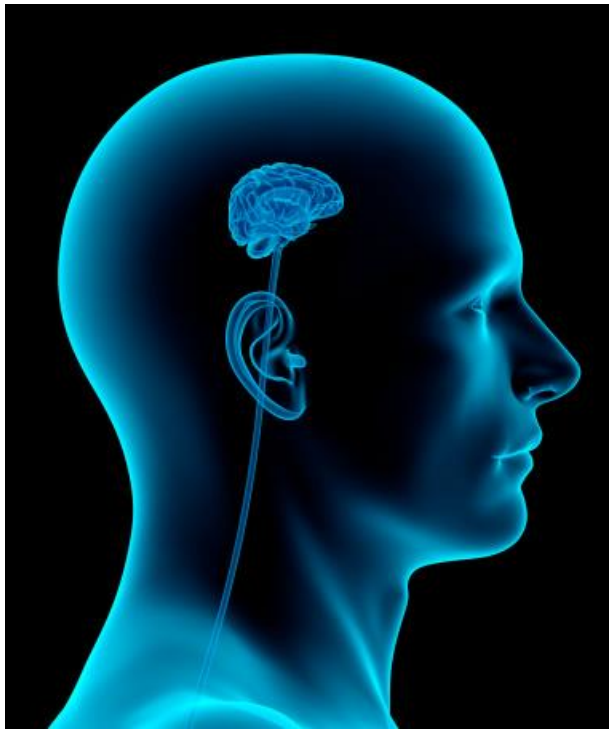
Finally, we have for $\tilde{e} = \gamma_{\{s_0^1,r_1^1\}}(s_0 \parallel r_1)$, $\tilde{e}^1 = s_0^2$, and $\tilde{e}^2 = r_1^2$,

$$t = \gamma_{\{r_1^2,r_2^2\}}(\gamma_{\{s_1^2,r_2^1\}}(\gamma_{\{s_0^2,s_1^1\}}(\gamma_{\{s_0^1,r_1^1\}}(s_0 \parallel r_1 \parallel s_1 \parallel r_2)))) = \gamma_{\{\tilde{e}^2,r_2^2\}}(\gamma_{\{s_1^2,r_2^1\}}(\gamma_{\{\tilde{e}^1,s_1^1\}}(\tilde{e} \parallel s_1 \parallel r_2))) \in \mathcal{S}_1.$$

Therefore, we have $t \in \mathcal{S}_0 \cap \mathcal{S}_1 \cap \mathcal{S}_= \neq \emptyset$. □

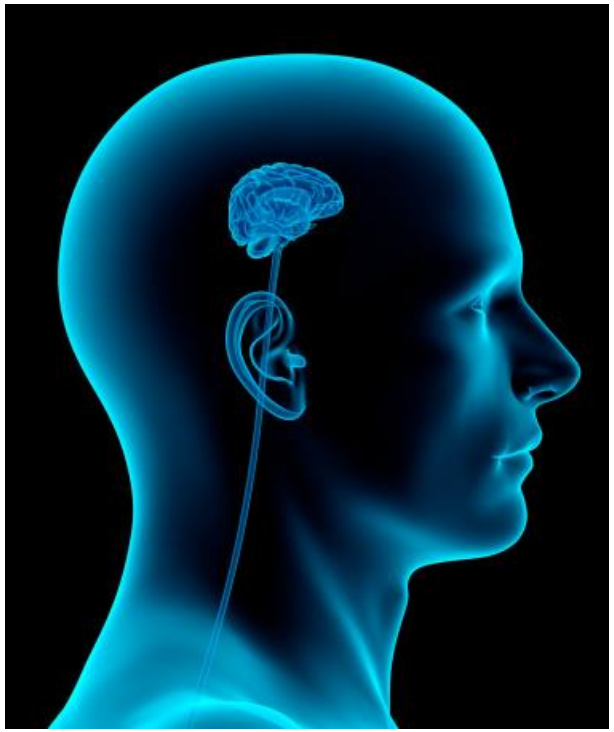
Abstraction

Abstraction

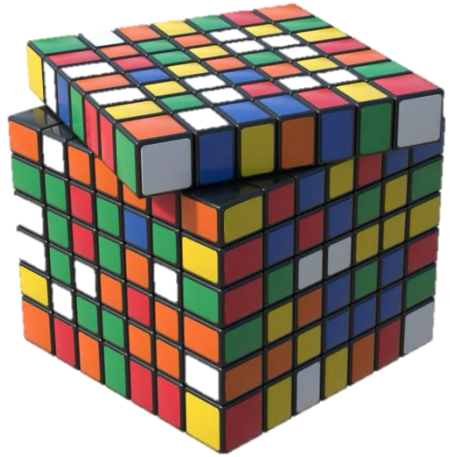




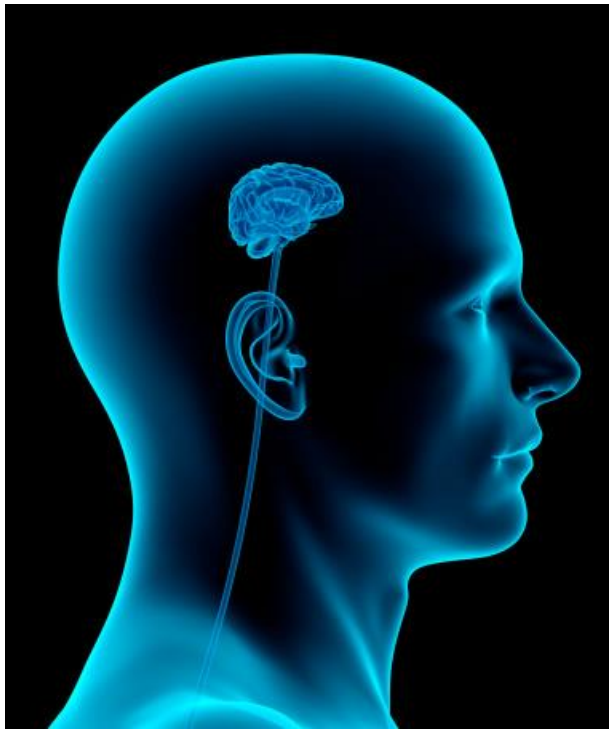
Abstraction



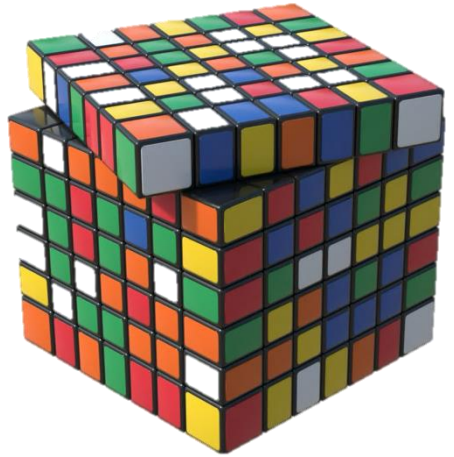


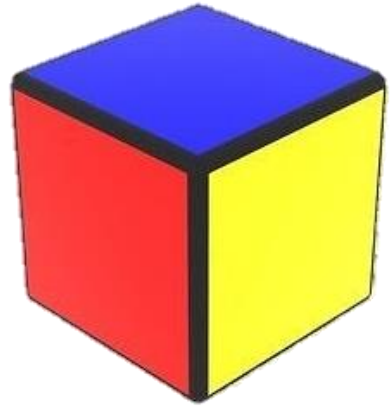


Abstraction

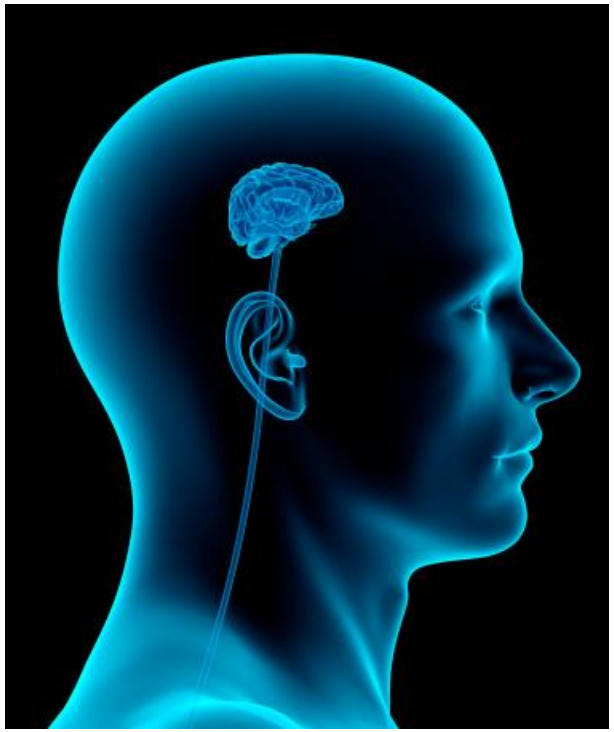




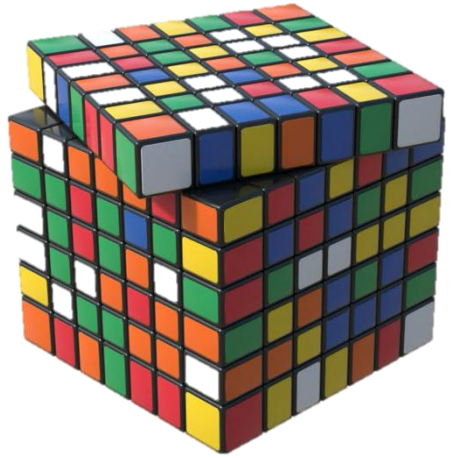


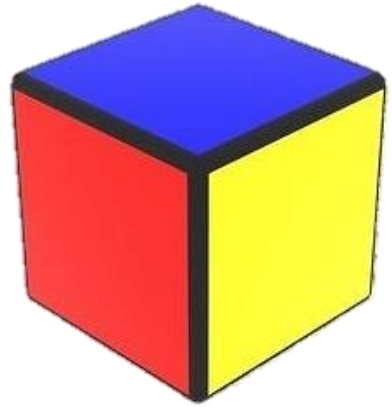


Abstraction





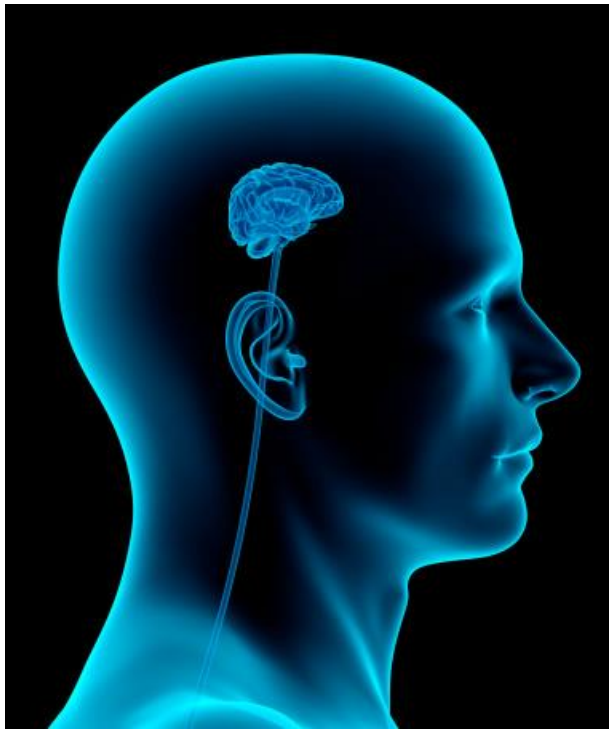




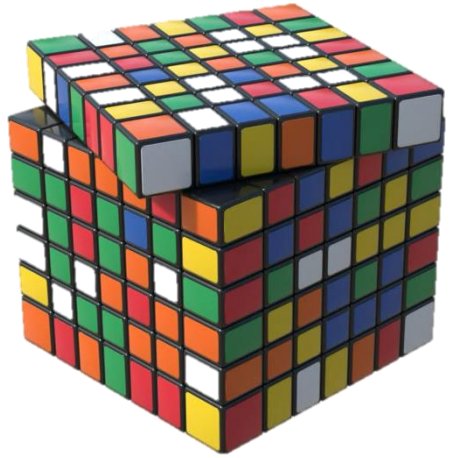


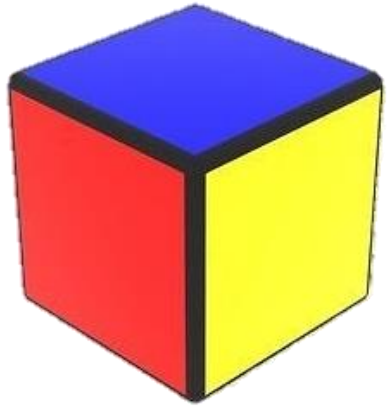


Abstraction





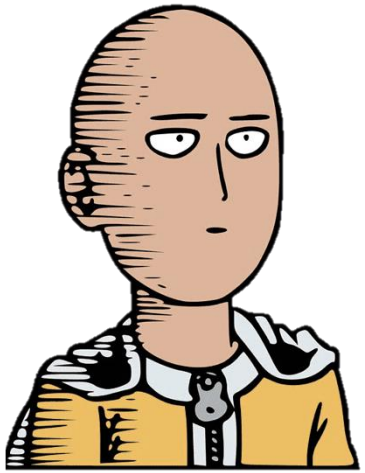












Same same, but different…

Same same, but different...







Same same, but different...









Different, but still the same…

Different, but still the same…

X +

Different, but still the same…

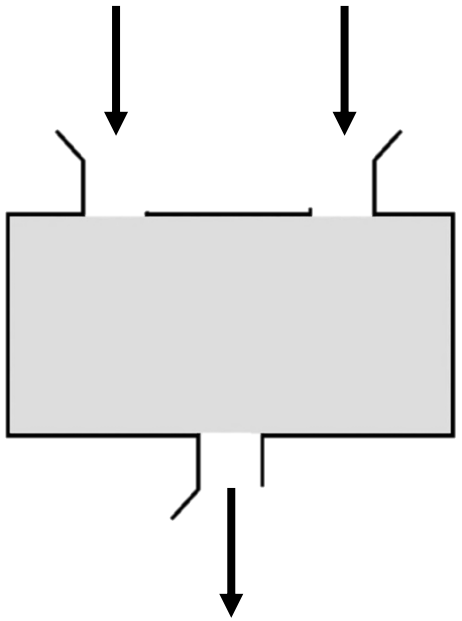
3 x 2 X 3 + 2

Different, but still the same…

X

+

3 x 2

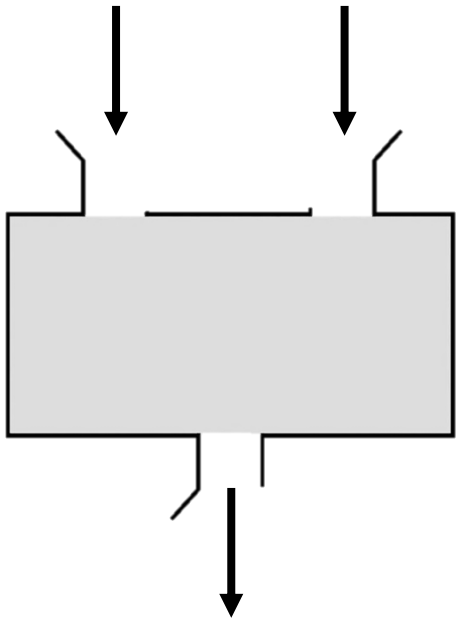


3 + 2

Different, but still the same…

3 x 2

X





+

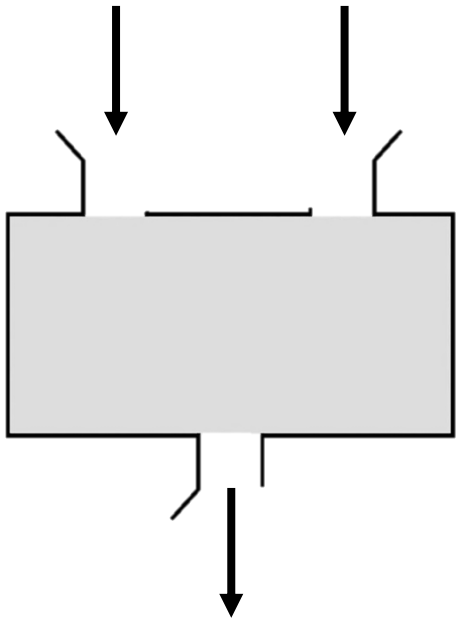
3 + 2

Different, but still the same…

X

+

3 x 2





3 + 2

$$(x\ y)\ z\ =\ x\ (y\ z)$$

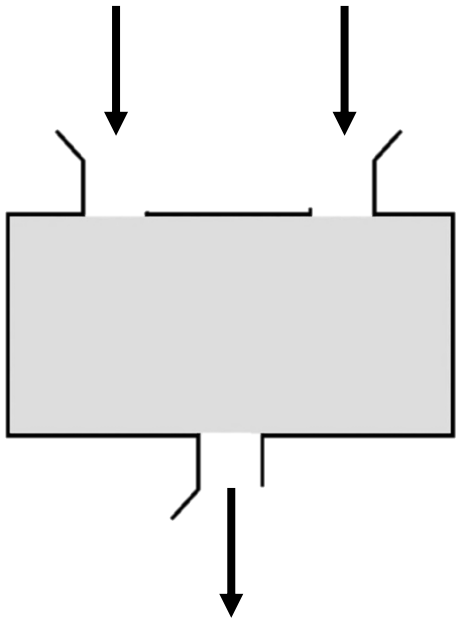
Different, but still the same…

X

+

3 x 2

3 + 2





$$y)\ z\ =\ x\ (y\ z)$$

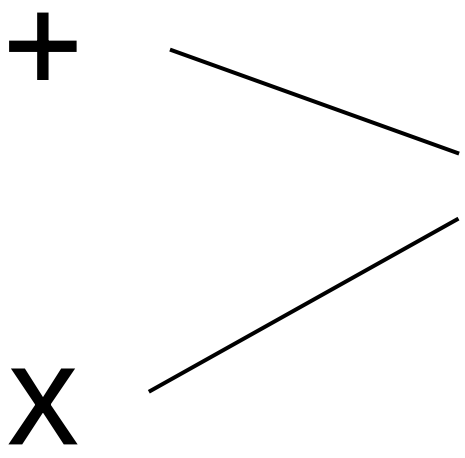
Algebraic Theories

Algebraic Theories

+

X

Algebraic Theories



Algebraic Theories

+

x

$$x.y = y.x$$

$$(x.y).z = x.(y.z)$$



Algebraic Theories

+

x

$$x.y = y.x$$
$$(x.y).z = x.(y.z)$$





Algebraic Theories

+

x

$$x.y = y.x$$
$$(x.y).z = x.(y.z)$$



Algebraic Theory



Algebraic Theories

+

x

$$x.y = y.x$$
$$(x.y).z = x.(y.z)$$



Algebraic Theory



...

Algebraic Theories

+

x

$$x.y = y.x$$
$$(x.y).z = x.(y.z)$$

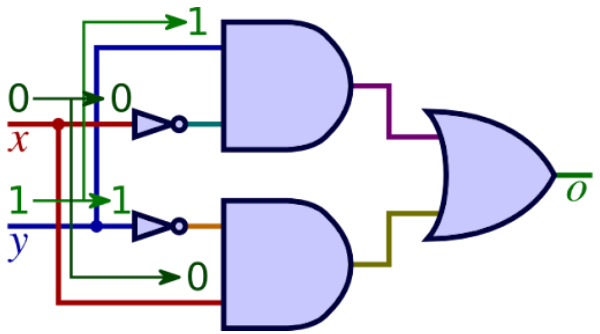




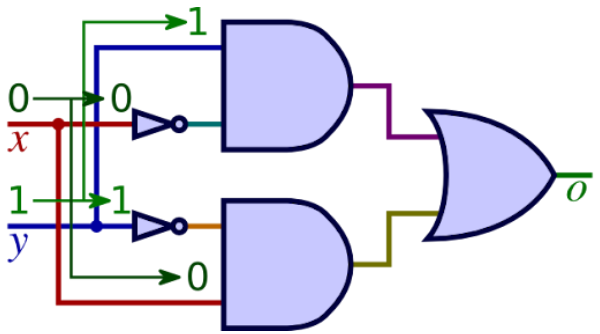


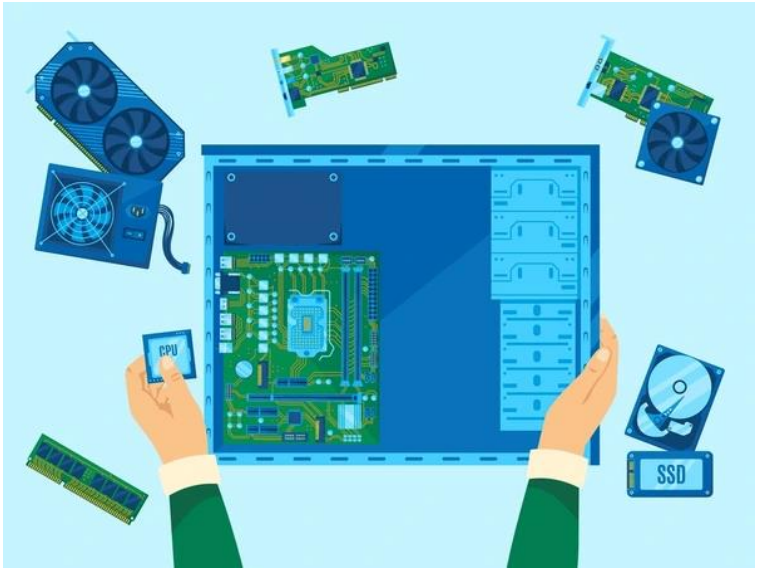
The Idea

The Idea

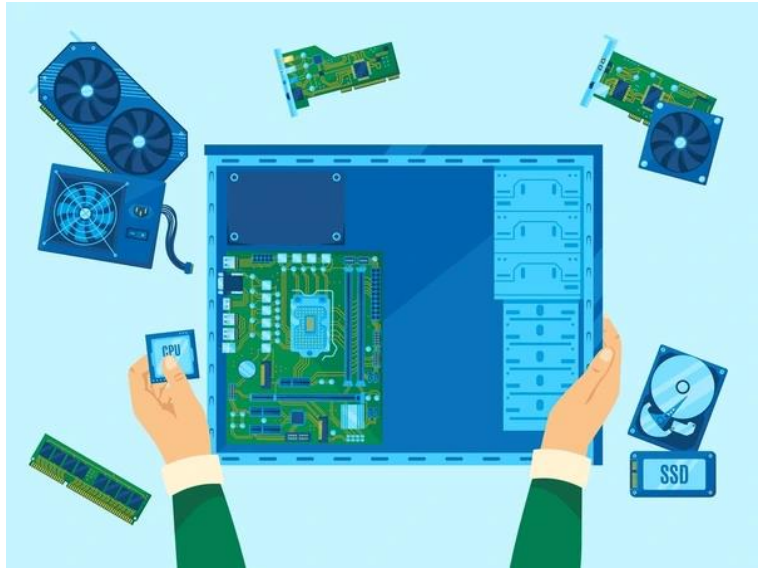


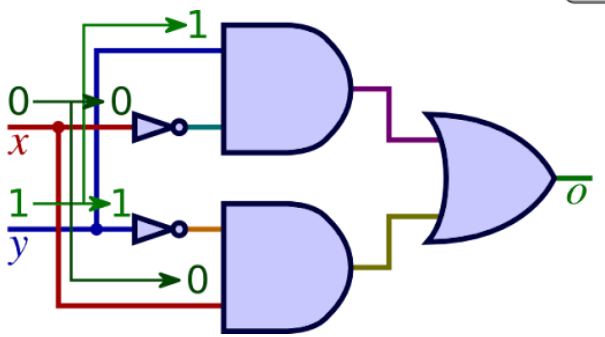
The Idea

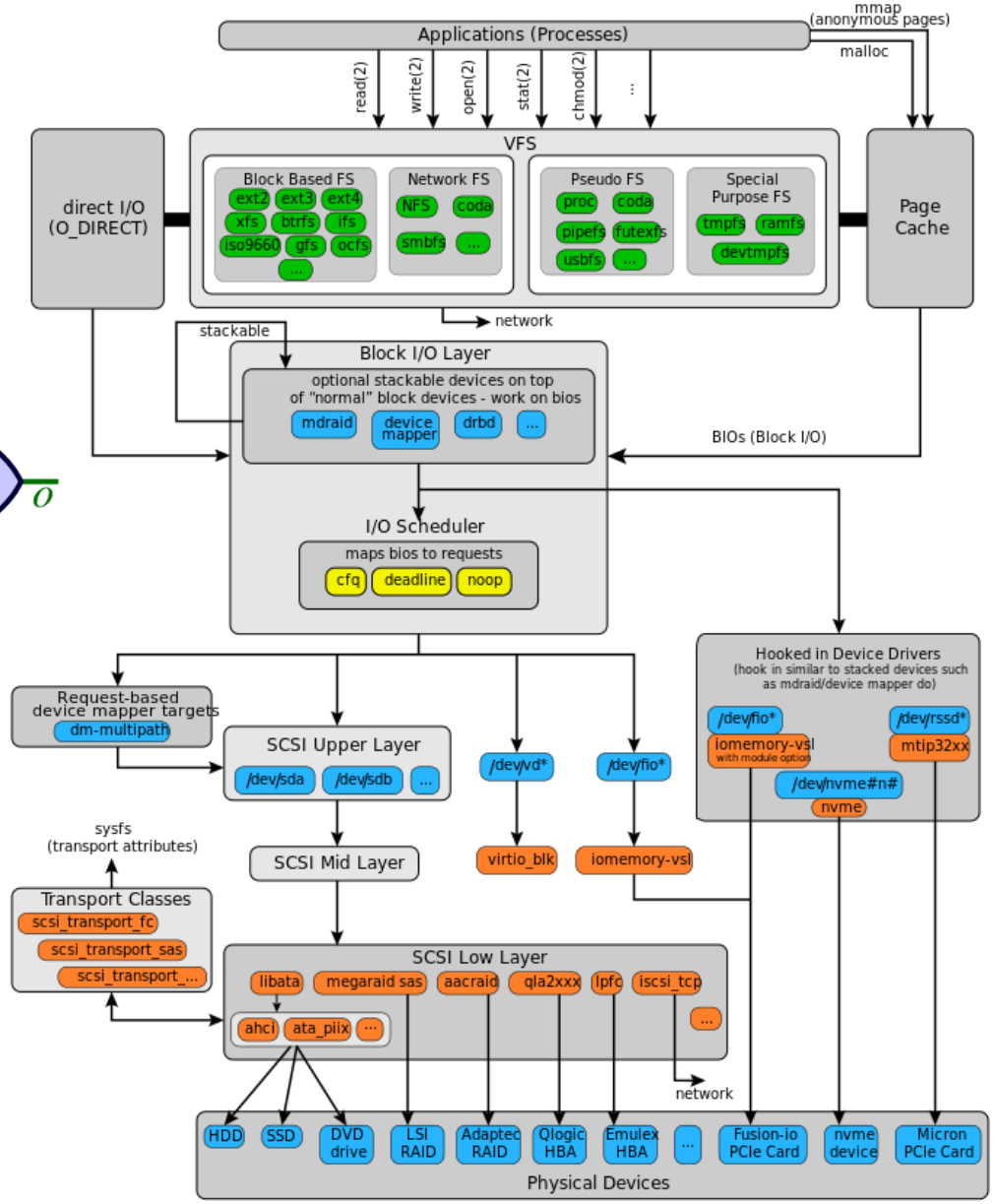




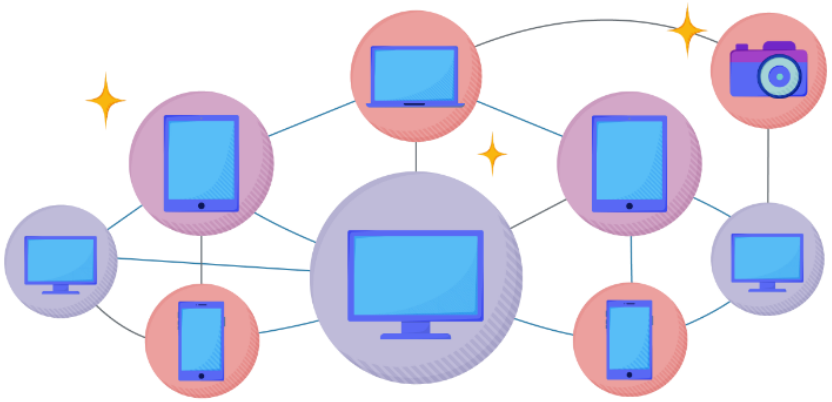
The Idea

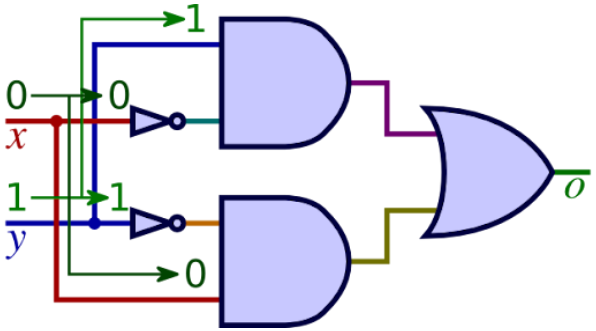

CPU
SSD

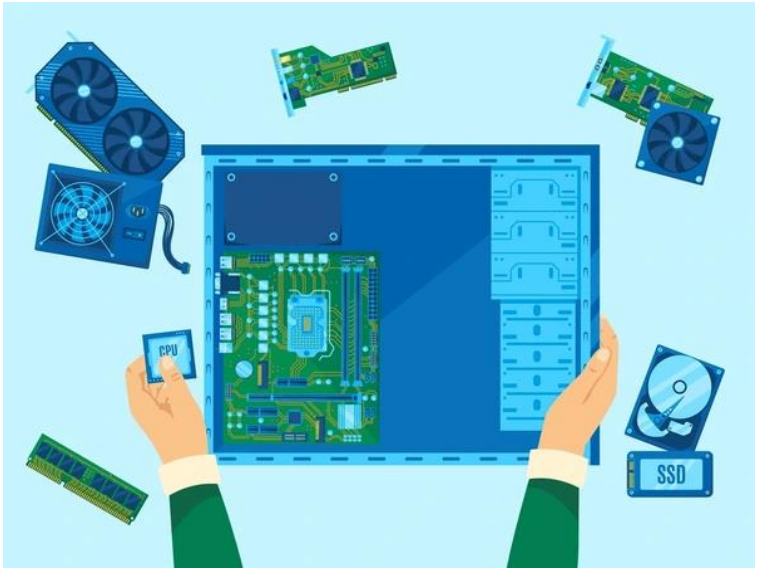

1
0
0
x
1
1
y
0
o

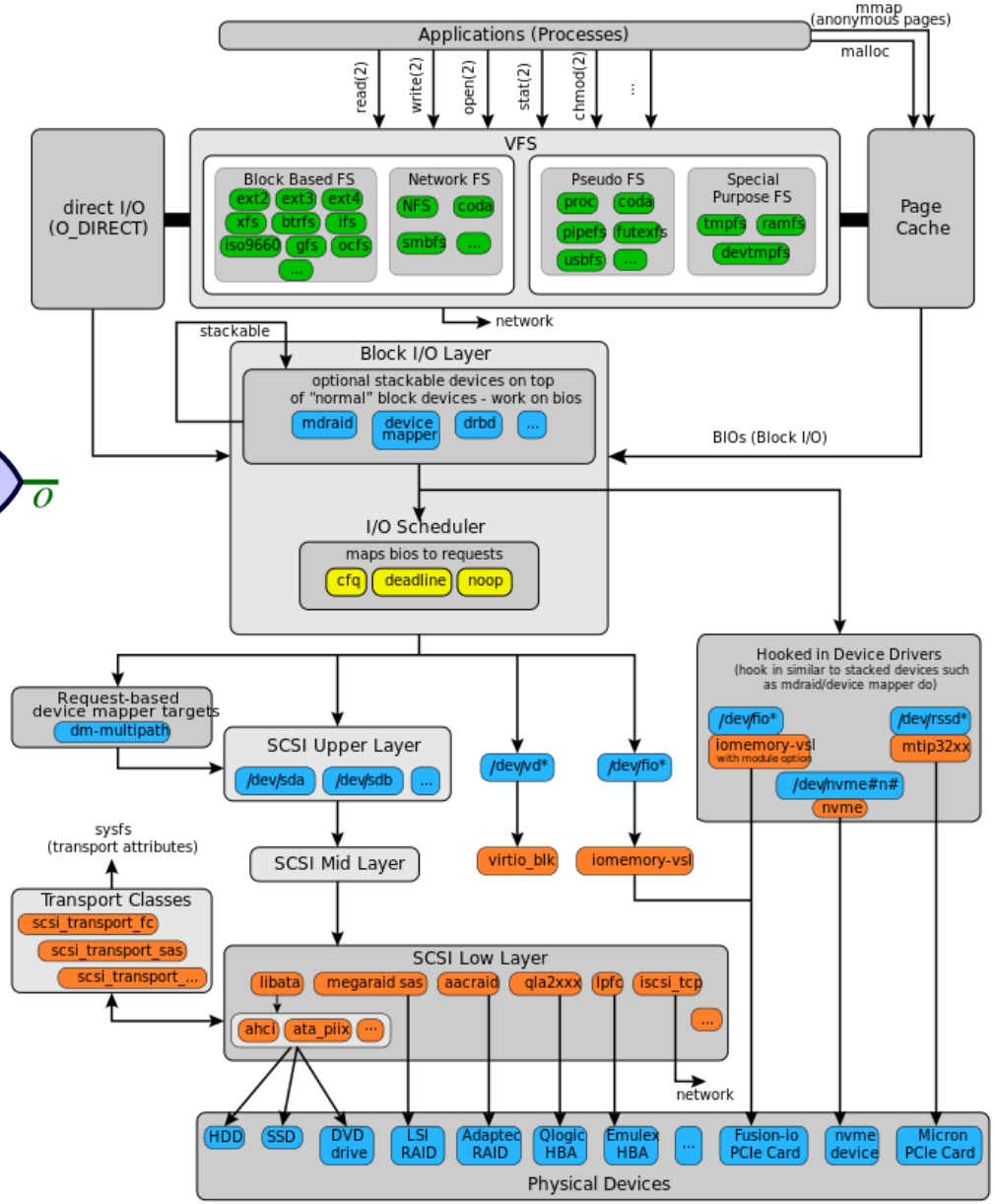

Applications (Processes)
mmap
(anonymous pages)
malloc
read(2)
write(2)
open(2)
stat(2)
chmod(2)
...
VFS
direct I/O
(O_DIRECT)
Block Based FS
ext2 ext3 ext4
xfs btrfs ifs
iso9660 gfs ocfs
...
Network FS
NFS coda
smbfs ...
Pseudo FS
proc coda
pipefs futexfs
usbfs ...
Special
Purpose FS
tmpfs ramfs
devtmpfs
Page
Cache
network
stackable
Block I/O Layer
optional stackable devices on top
of "normal" block devices - work on bios
mdraid
device
mapper
drbd
...
BIOs (Block I/O)
I/O Scheduler
maps bios to requests
cfq
deadline
noop
Hooked in Device Drivers
(hook in similar to stacked devices such
as mdraid/device mapper do)
/dev/fio*
/dev/rssd*
iomemory-vsl
with module option
mtip32xx
/dev/nvme#n#
nvme
Request-based
device mapper targets
dm-multipath
SCSI Upper Layer
/dev/sda
/dev/sdb
...
/dev/vd*
/dev/fio*
sysfs
(transport attributes)
SCSI Mid Layer
virtio_blk
iomemory-vsl
Transport Classes
scsi_transport_fc
scsi_transport_sas
scsi_transport_...
SCSI Low Layer
libata
megaraid sas
aacraid
qla2xxx
lpfc
iscsi_tcp
...
ahci
ata_piix
...
network
HDD
SSD
DVD
drive
LSI
RAID
Adaptec
RAID
Qlogic
HBA
Emulex
HBA
...
Fusion-io
PCIe Card
nvme
device
Micron
PCIe Card
Physical Devices

The Idea

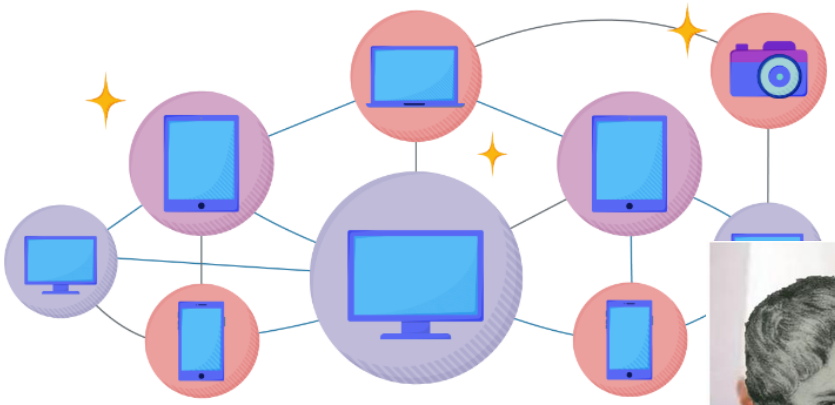


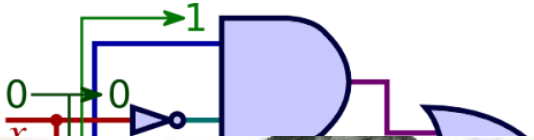

1
0
0
x
1
1
y
0
o


CPU
SSD

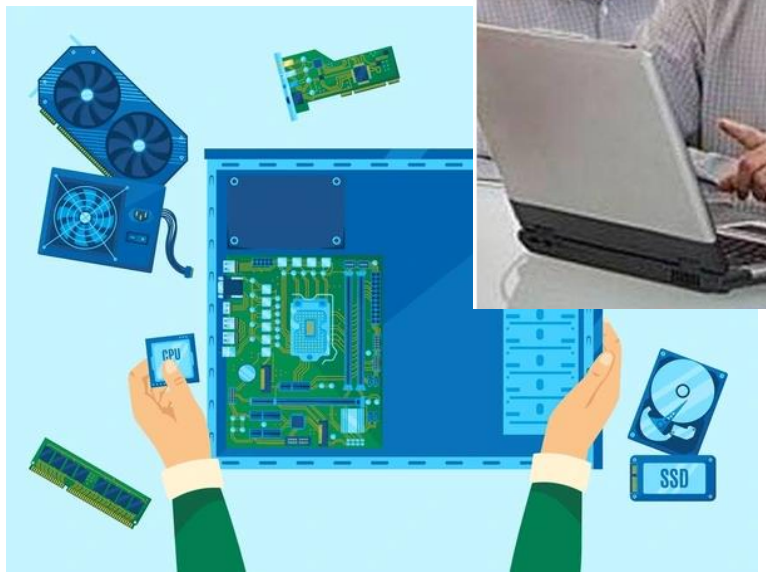

Applications (Processes)
mmap
(anonymous pages)
malloc
read(2)
write(2)
open(2)
stat(2)
chmod(2)
...
VFS
direct I/O
(O_DIRECT)
Block Based FS
ext2
ext3
ext4
xfs
btrfs
ifs
iso9660
gfs
ocfs
...
Network FS
NFS
coda
smbfs
...
Pseudo FS
proc
coda
pipefs
futexfs
usbfs
...
Special
Purpose FS
tmpfs
ramfs
devtmpfs
Page
Cache
network
stackable
Block I/O Layer
optional stackable devices on top
of "normal" block devices - work on bios
mdraid
device
mapper
drbd
...
BIOs (Block I/O)
I/O Scheduler
maps bios to requests
cfq
deadline
noop
Hooked in Device Drivers
(hook in similar to stacked devices such
as mdraid/device mapper do)
/dev/fio*
/dev/rssd*
iomemory-vsl
with module option
mtip32xx
/dev/nvme#n#
nvme
Request-based
device mapper targets
dm-multipath
SCSI Upper Layer
/dev/sda
/dev/sdb
...
/dev/vd*
/dev/fio*
sysfs
(transport attributes)
SCSI Mid Layer
virtio_blk
iomemory-vsl
Transport Classes
scsi_transport_fc
scsi_transport_sas
scsi_transport_...
SCSI Low Layer
libata
megaraid sas
aacraid
qla2xxx
lpfc
iscsi_tcp
...
ahci
ata_piix
...
network
HDD
SSD
DVD
drive
LSI
RAID
Adaptec
RAID
Qlogic
HBA
Emulex
HBA
...
Fusion-io
PCIe Card
nvme
device
Micron
PCIe Card
Physical Devices

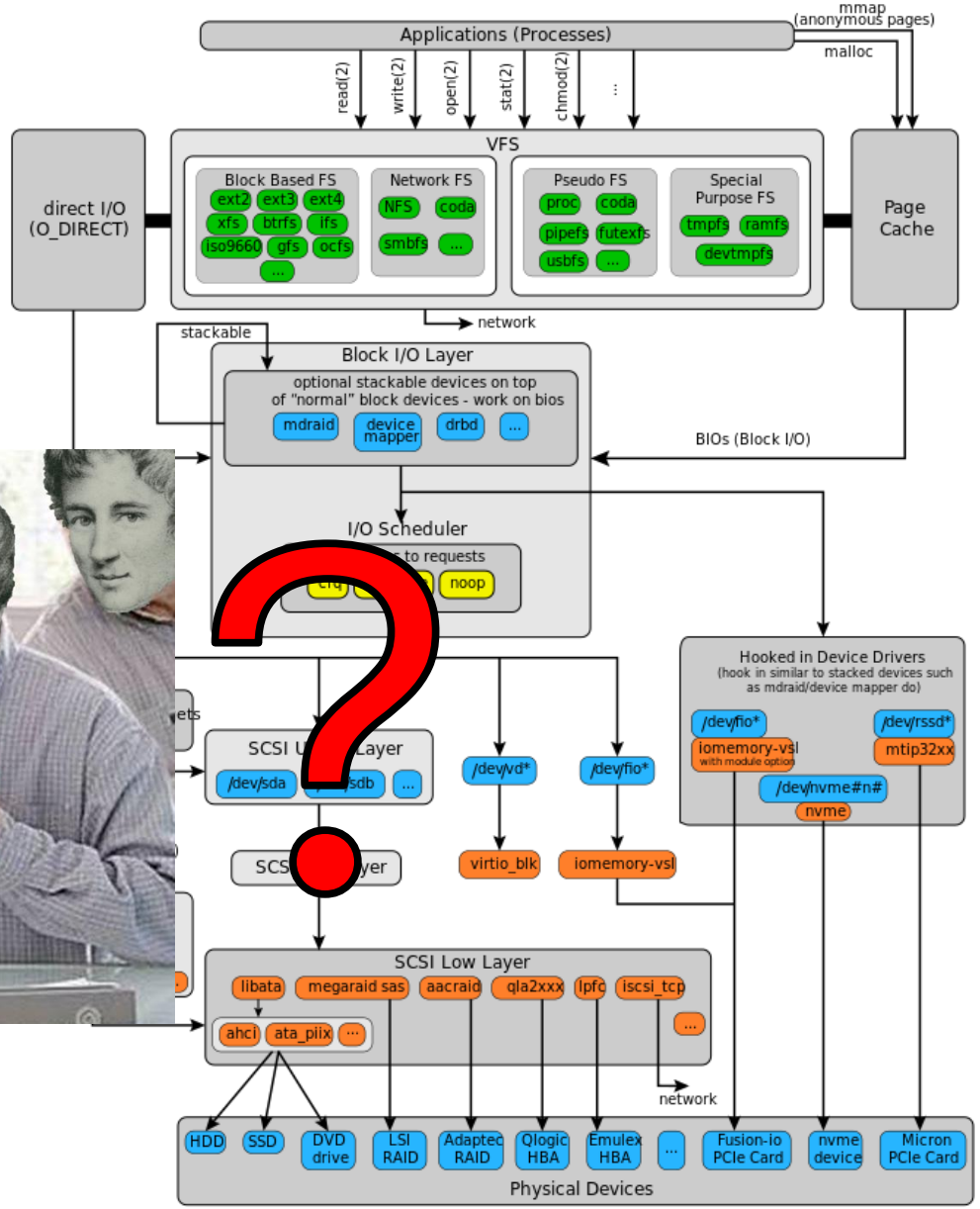
The Idea




1
0
0




CPU
SSD


Applications (Processes)
mmap
(anonymous pages)
malloc
read(2)
write(2)
open(2)
stat(2)
chmod(2)
VFS
direct I/O
(O_DIRECT)
Block Based FS
ext2
ext3
ext4
xfs
btrfs
ifs
iso9660
gfs
ocfs
Network FS
NFS
coda
smbfs
Pseudo FS
proc
coda
pipefs
futexfs
usbfs
Special
Purpose FS
tmpfs
ramfs
devtmpfs
Page
Cache
network
stackable
Block I/O Layer
optional stackable devices on top
of "normal" block devices - work on bios
mdraid
device
mapper
drbd
BIOs (Block I/O)
I/O Scheduler
noop
Hooked in Device Drivers
(hook in similar to stacked devices such
as mdraid/device mapper do)
/dev/fio*
/dev/rssd*
iomemory-vsl
with module option
mtip32xx
/dev/nvme#n#
nvme
/dev/sda
/dev/vd*
/dev/fio*
virtio_blk
iomemory-vsl
SCSI Low Layer
libata
megaraid sas
aacraid
qla2xxx
lpfc
iscsi_tcp
ahci
ata_piix
network
HDD
SSD
DVD
drive
LSI
RAID
Adaptec
RAID
Qlogic
HBA
Emulex
HBA
Fusion-io
PCIe Card
nvme
device
Micron
PCIe Card
Physical Devices

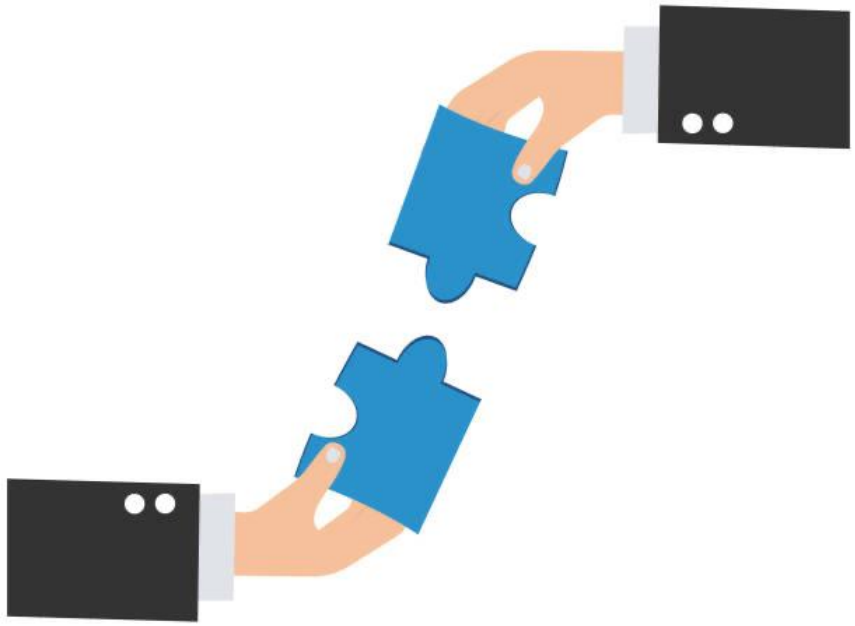
An ‘Algebraic Theory’ of Systems

An ‘Algebraic Theory’ of Systems



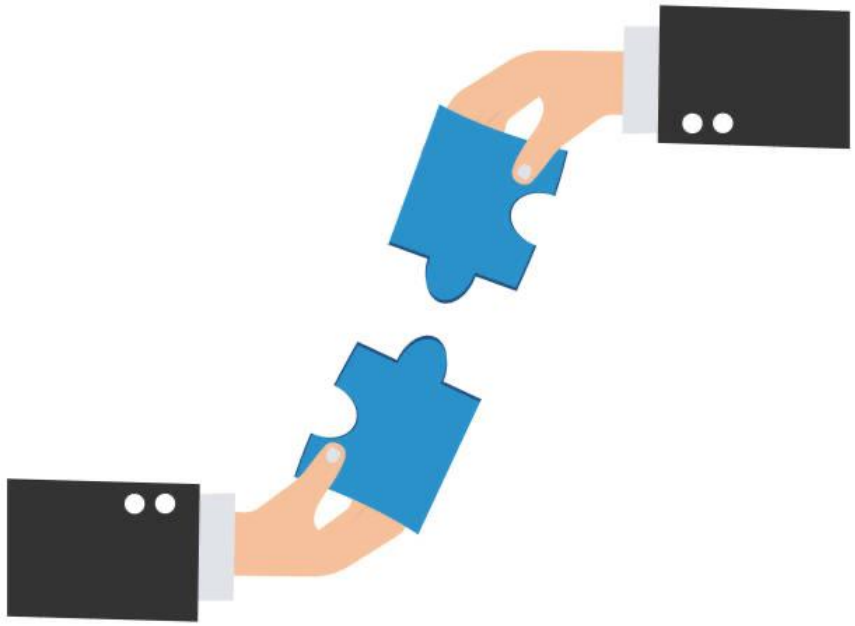
An ‘Algebraic Theory’ of Systems





An ‘Algebraic Theory’ of Systems

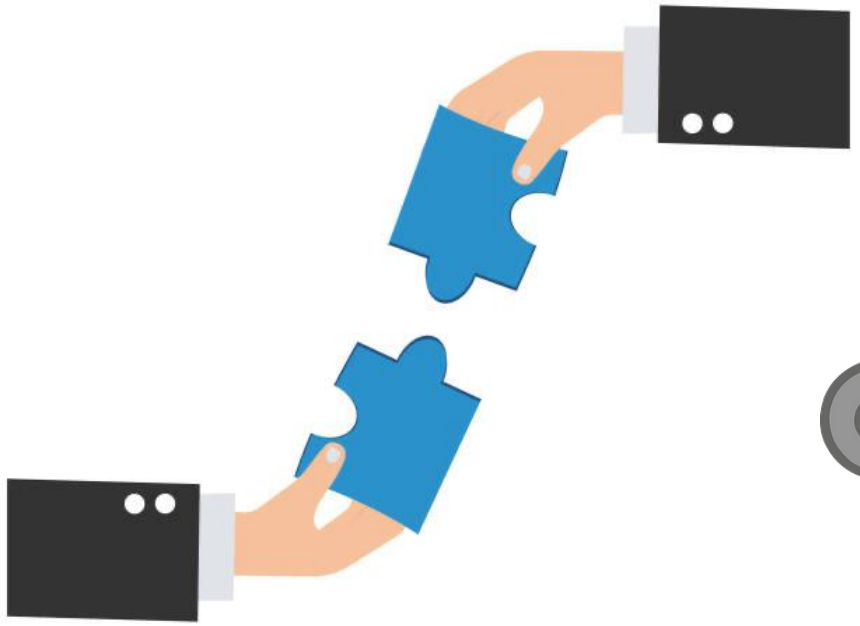




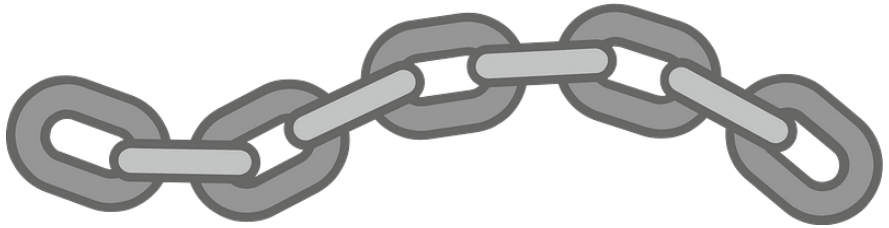


An ‘Algebraic Theory’ of Systems



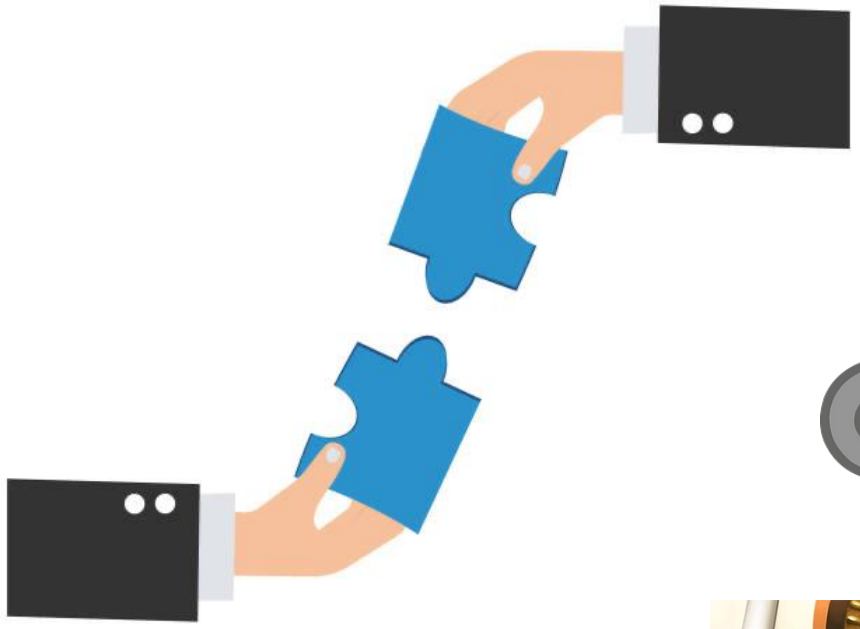




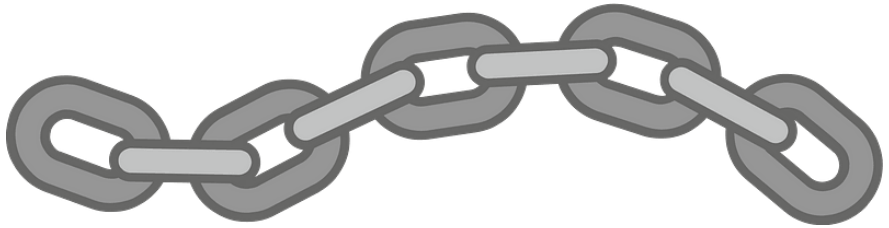


An ‘Algebraic Theory’ of Systems





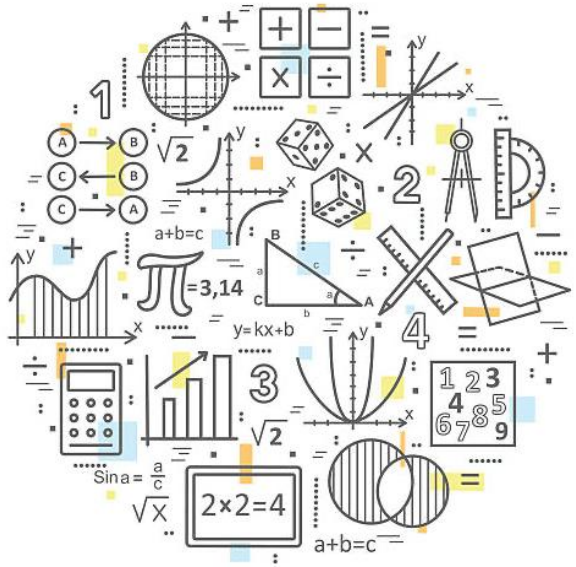




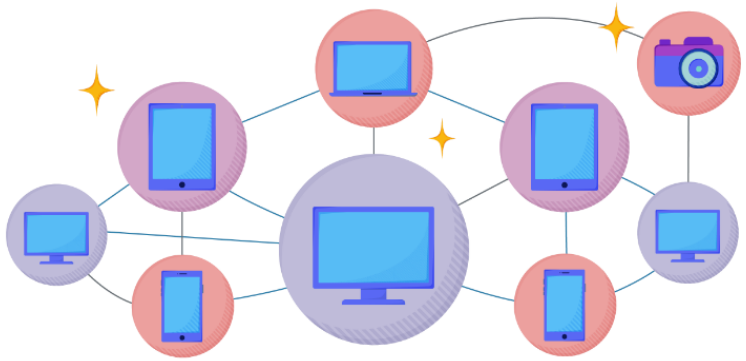


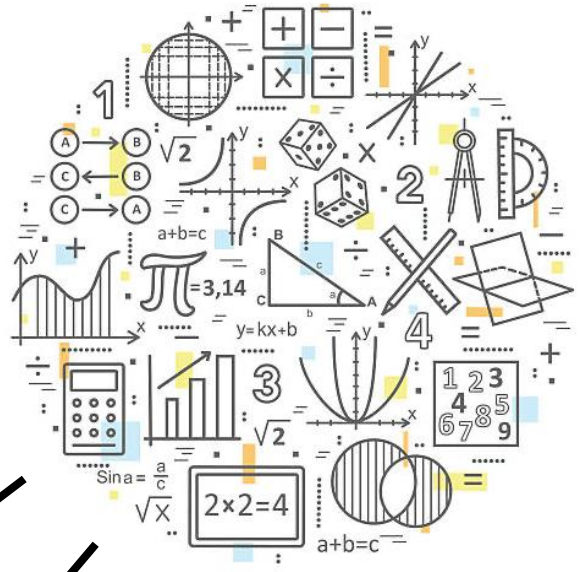
The Dream

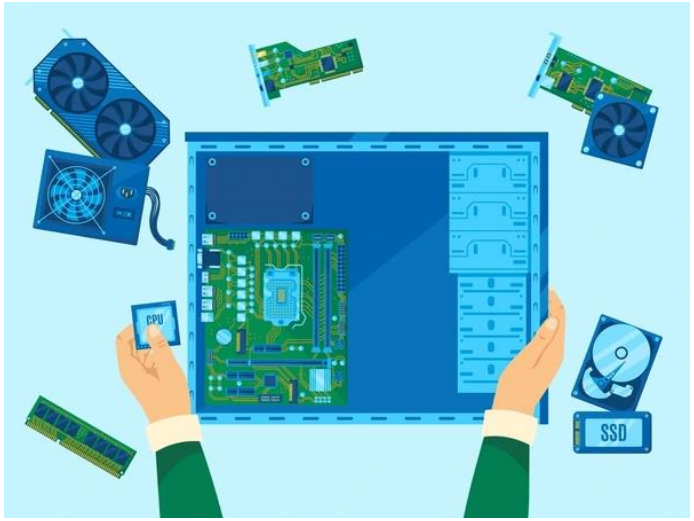
The Dream

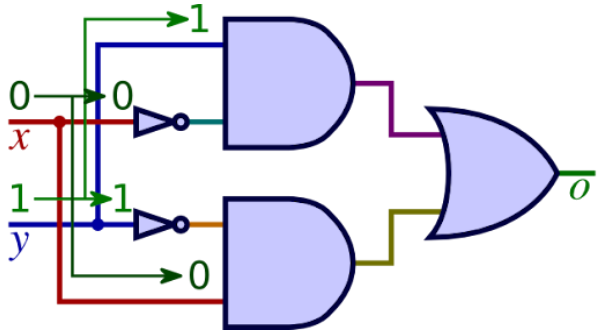


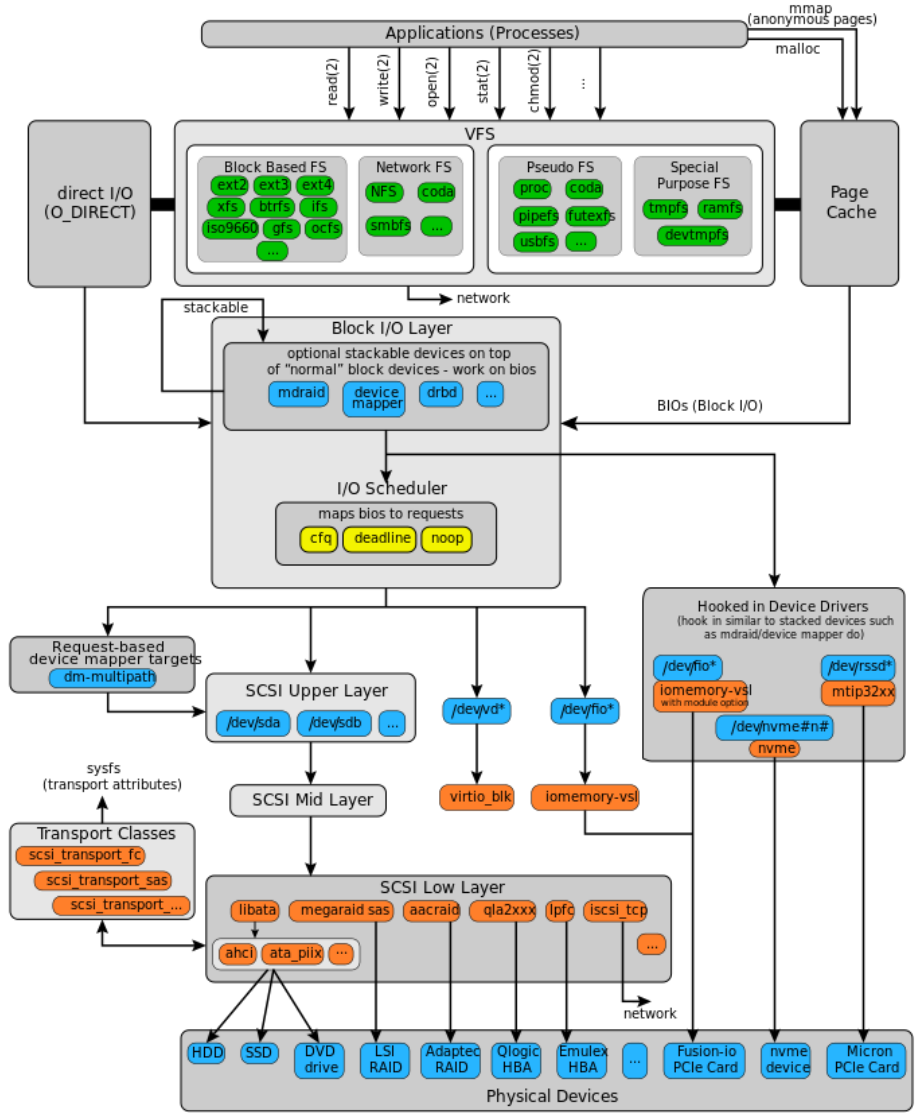
The Dream



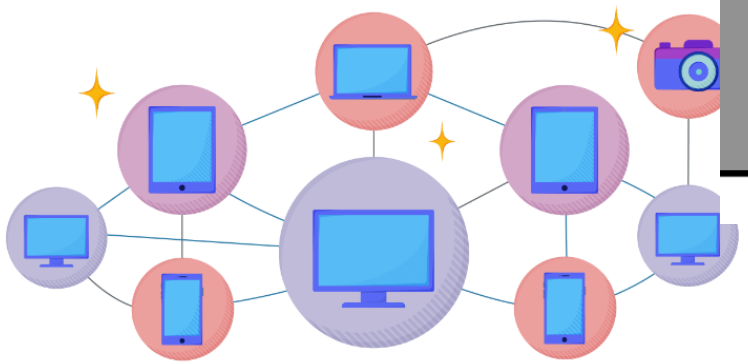

a+b=c
=3,14
y=kx+b
Sin a = a/c
2×2=4
a+b=c

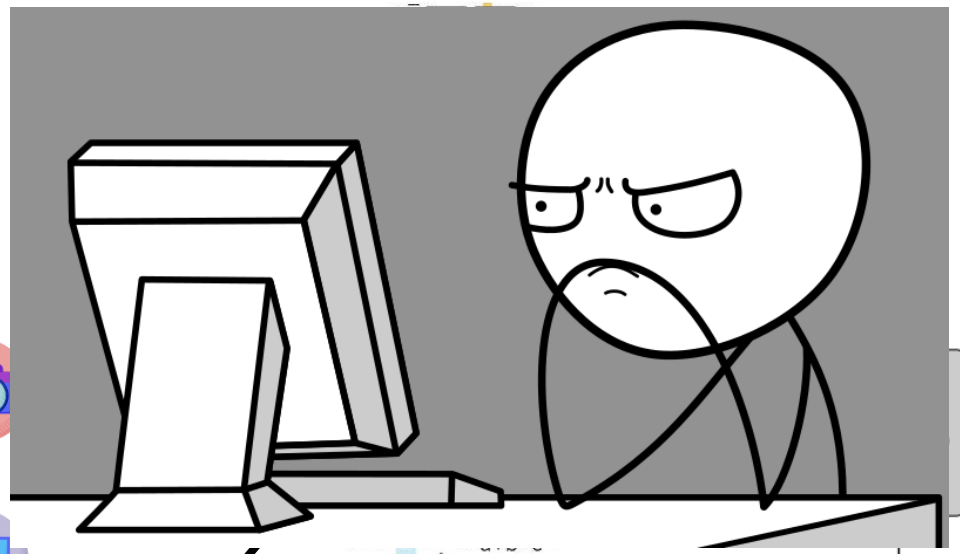

CPU
SSD

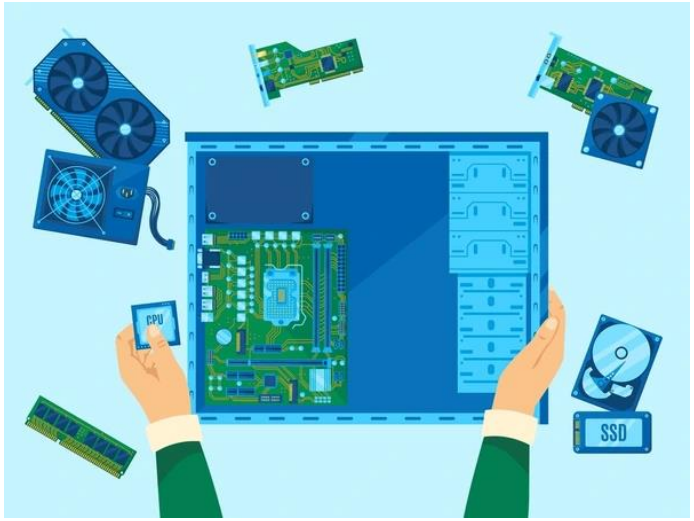

0
x
1
y
1
0
1
0
o

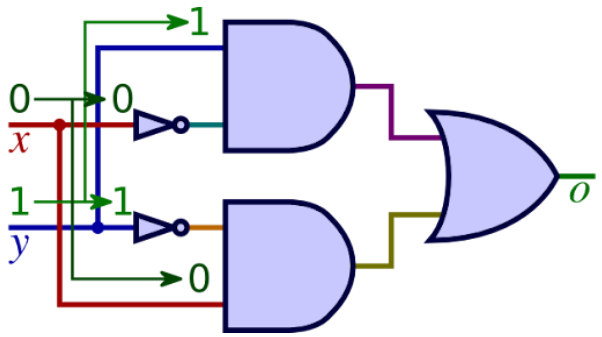

Applications (Processes)
mmap
(anonymous pages)
malloc
read(2)
write(2)
open(2)
stat(2)
chmod(2)
VFS
direct I/O
(O_DIRECT)
Block Based FS
ext2 ext3 ext4
xfs btrfs ifs
iso9660 gfs ocfs
Network FS
NFS coda
smbfs ...
Pseudo FS
proc coda
pipefs futexfs
usbfs ...
Special
Purpose FS
tmpfs ramfs
devtmpfs
Page
Cache
network
stackable
Block I/O Layer
optional stackable devices on top
of "normal" block devices - work on bios
mdraid
device
mapper
drbd ...
BIOs (Block I/O)
I/O Scheduler
maps bios to requests
cfq deadline noop
Request-based
device mapper targets
dm-multipath
SCSI Upper Layer
/dev/sda /dev/sdb ...
/dev/vd*
/dev/fio*
Hooked in Device Drivers
(hook in similar to stacked devices such
as mdraid/device mapper do)
/dev/fio*
/dev/rssd*
iomemory-vsl
with module option
mtip32xx
/dev/nvme#n#
nvme
sysfs
(transport attributes)
SCSI Mid Layer
virtio_blk
iomemory-vsl
Transport Classes
scsi_transport_fc
scsi_transport_sas
scsi_transport_...
SCSI Low Layer
libata
megaraid sas
aacraid
qla2xxx
lpfc
iscsi_tcp
ahci
ata_piix
...
network
HDD
SSD
DVD
drive
LSI
RAID
Adaptec
RAID
Qlogic
HBA
Emulex
HBA
...
Fusion-io
PCIe Card
nvme
device
Micron
PCIe Card
Physical Devices

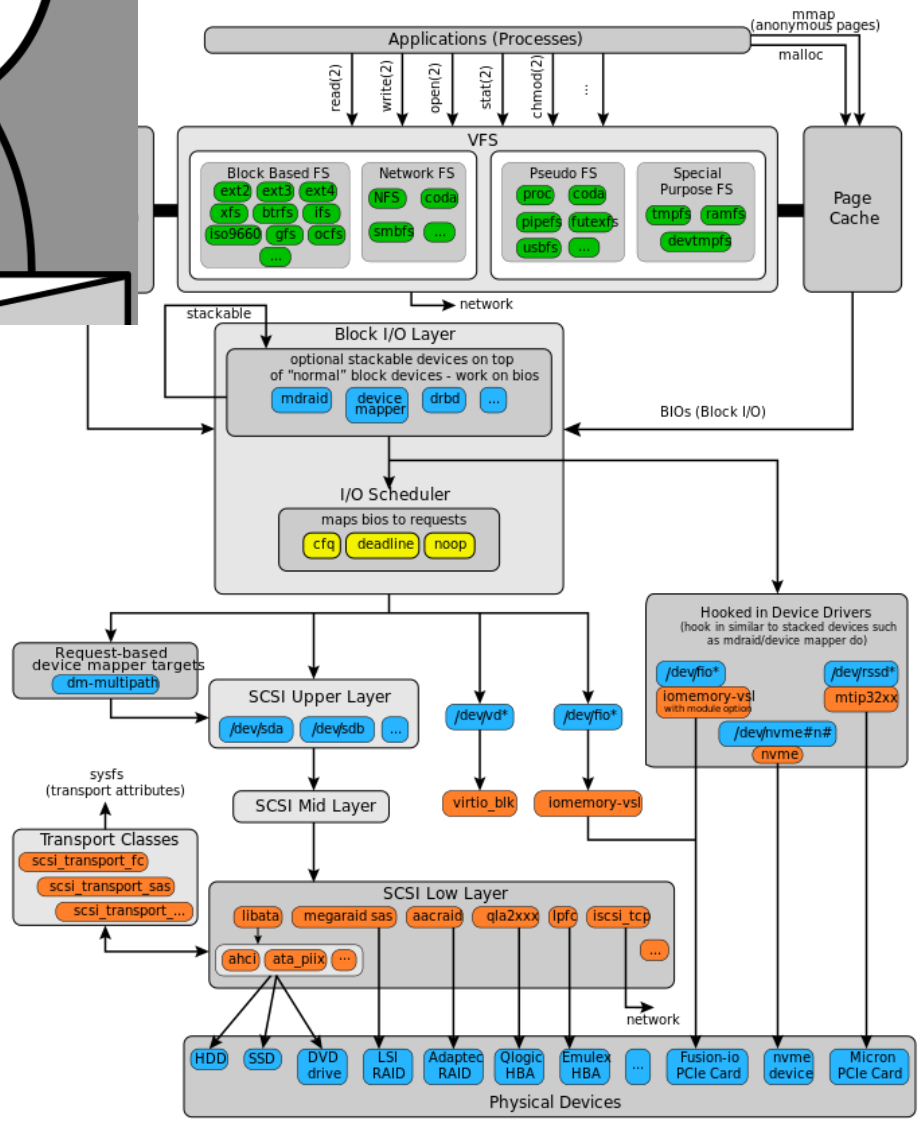
The Dream






CPU
SSD


1
0
0
x
1
1
y
0
o


mmap
(anonymous pages)
Applications (Processes)
malloc
read(2)
write(2)
open(2)
stat(2)
chmod(2)
...
VFS
Block Based FS
ext2 ext3 ext4
xfs btrfs ifs
iso9660 gfs ocfs
...
Network FS
NFS coda
smbfs ...
Pseudo FS
proc coda
pipefs futexfs
usbfs ...
Special Purpose FS
tmpfs ramfs
devtmpfs
Page Cache
network
stackable
Block I/O Layer
optional stackable devices on top of "normal" block devices - work on bios
mdraid
device mapper
drbd
...
BIOs (Block I/O)
I/O Scheduler
maps bios to requests
cfq
deadline
noop
Request-based device mapper targets
dm-multipath
SCSI Upper Layer
/dev/sda
/dev/sdb
...
/dev/vd*
/dev/fio*
Hooked in Device Drivers
(hook in similar to stacked devices such as mdraid/device mapper do)
/dev/fio*
/dev/rssd*
iomemory-vsl
with module option
mtip32xx
/dev/nvme#n#
nvme
sysfs
(transport attributes)
SCSI Mid Layer
virtio_blk
iomemory-vsl
Transport Classes
scsi_transport_fc
scsi_transport_sas
scsi_transport_...
SCSI Low Layer
libata
megaraid sas
aacraid
qla2xxx
lpfc
iscsi_tcp
...
ahci
ata_piix
...
network
HDD
SSD
DVD drive
LSI RAID
Adaptec RAID
Qlogic HBA
Emulex HBA
...
Fusion-io PCIe Card
nvme device
Micron PCIe Card
Physical Devices

The Plan

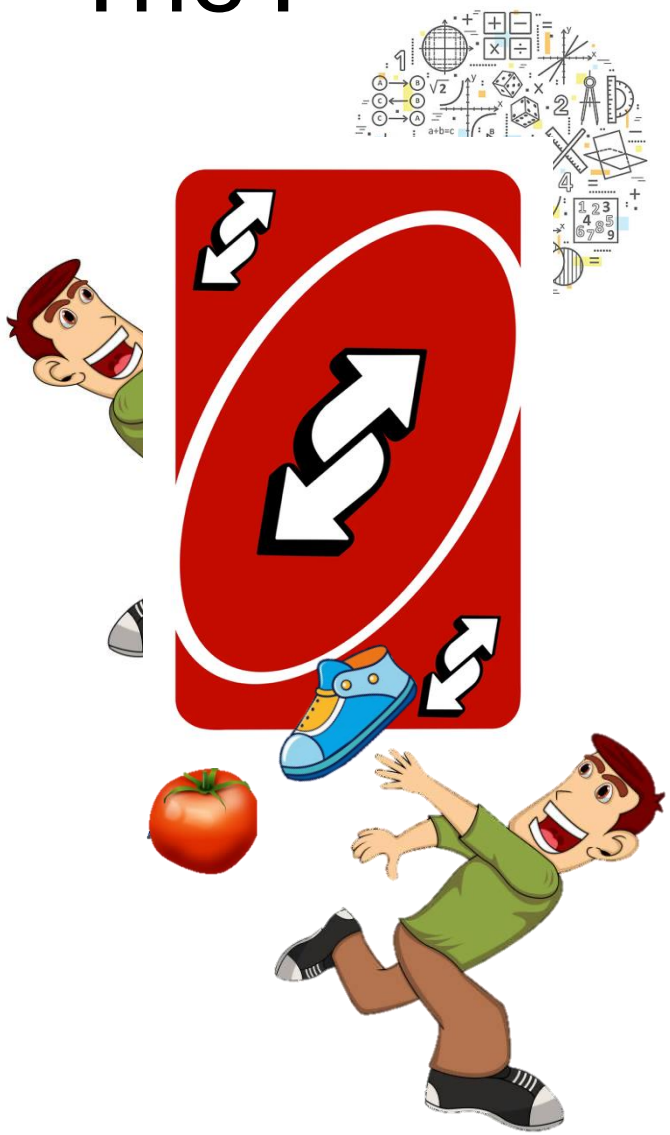
The Plan



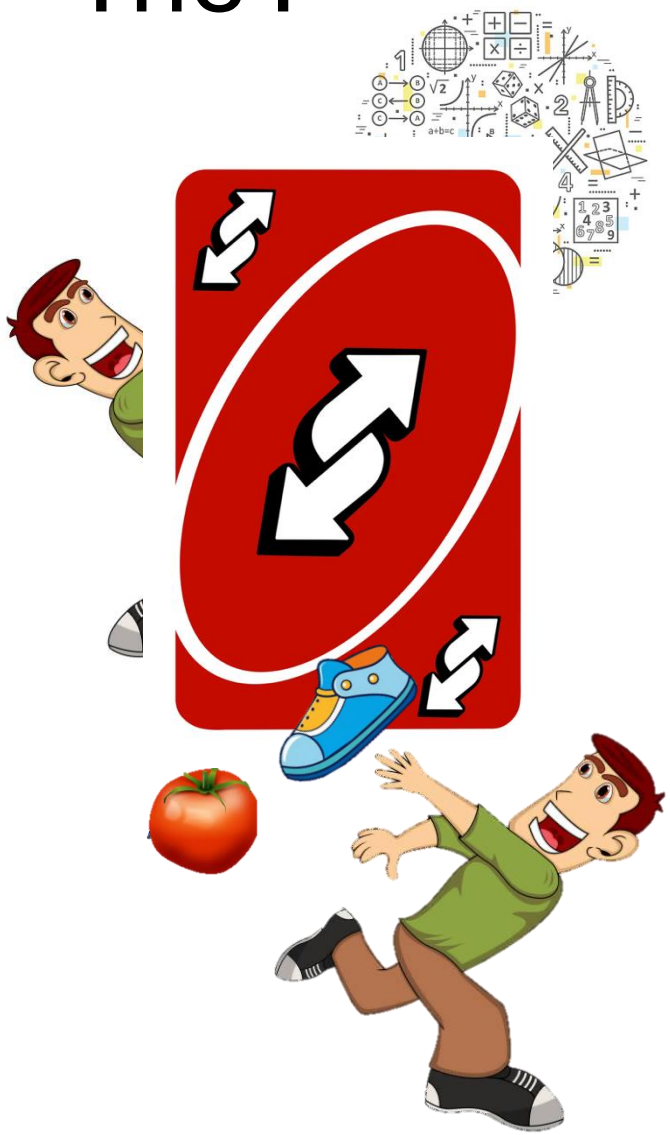
The Plan



The Plan

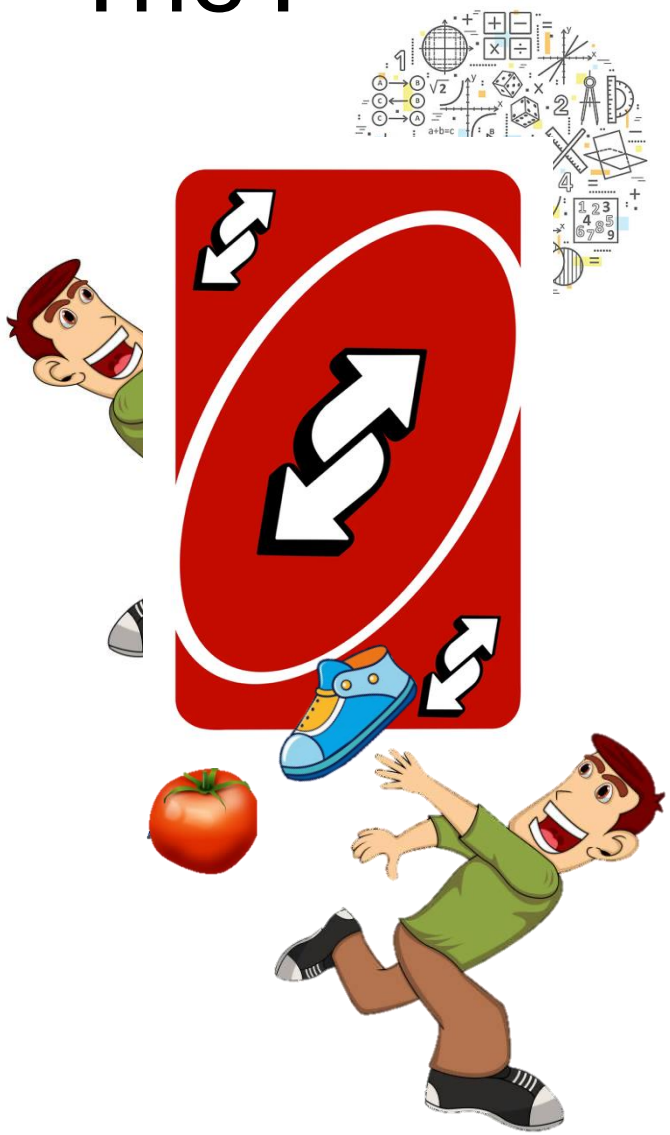


The Plan





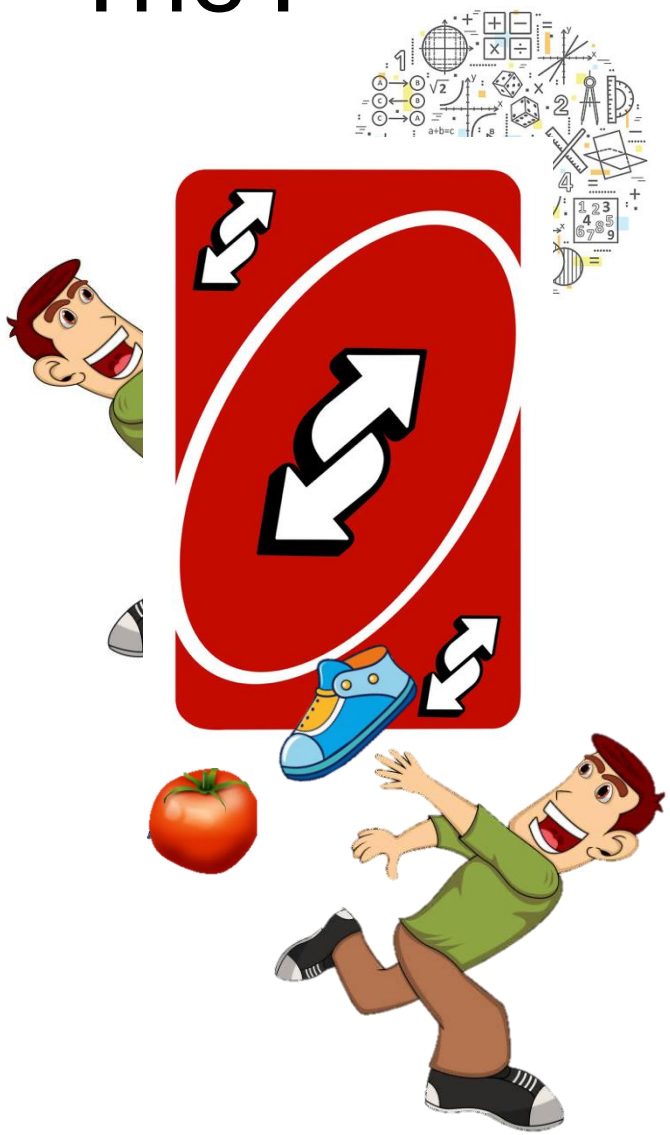
The Plan





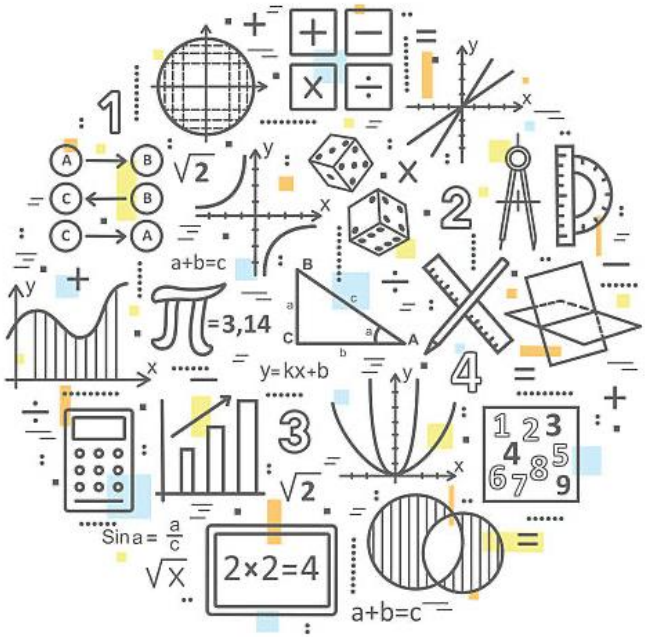


The Plan



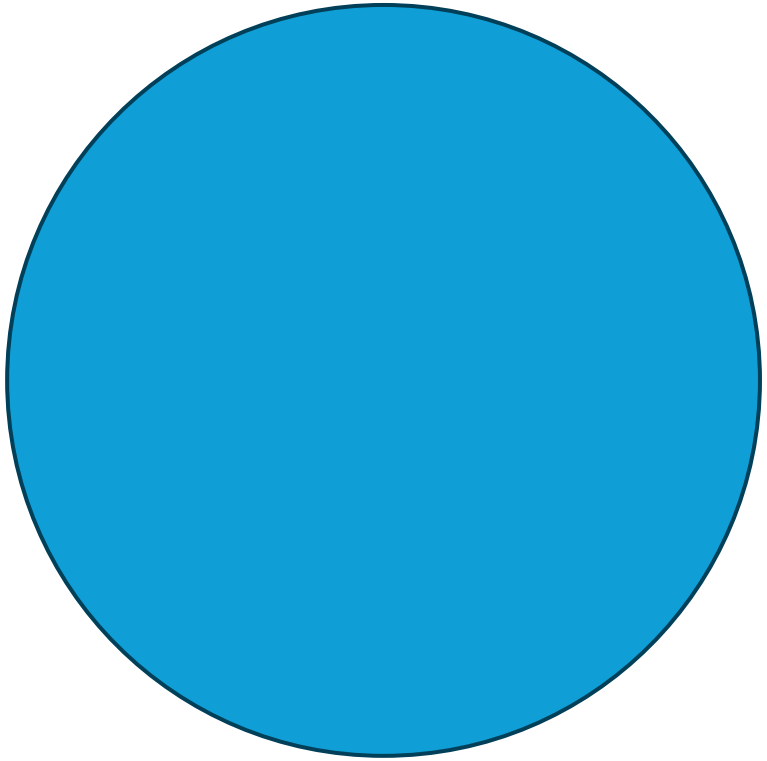




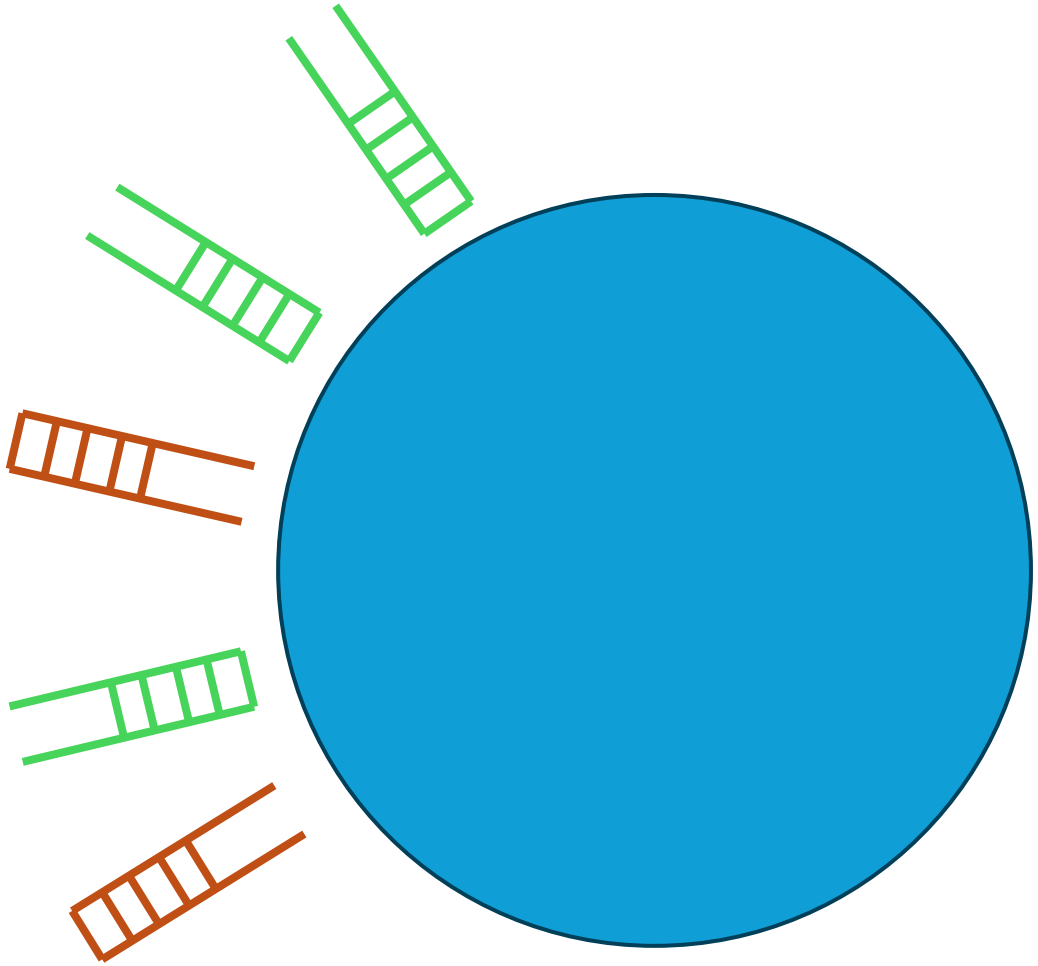


A Simple System Model

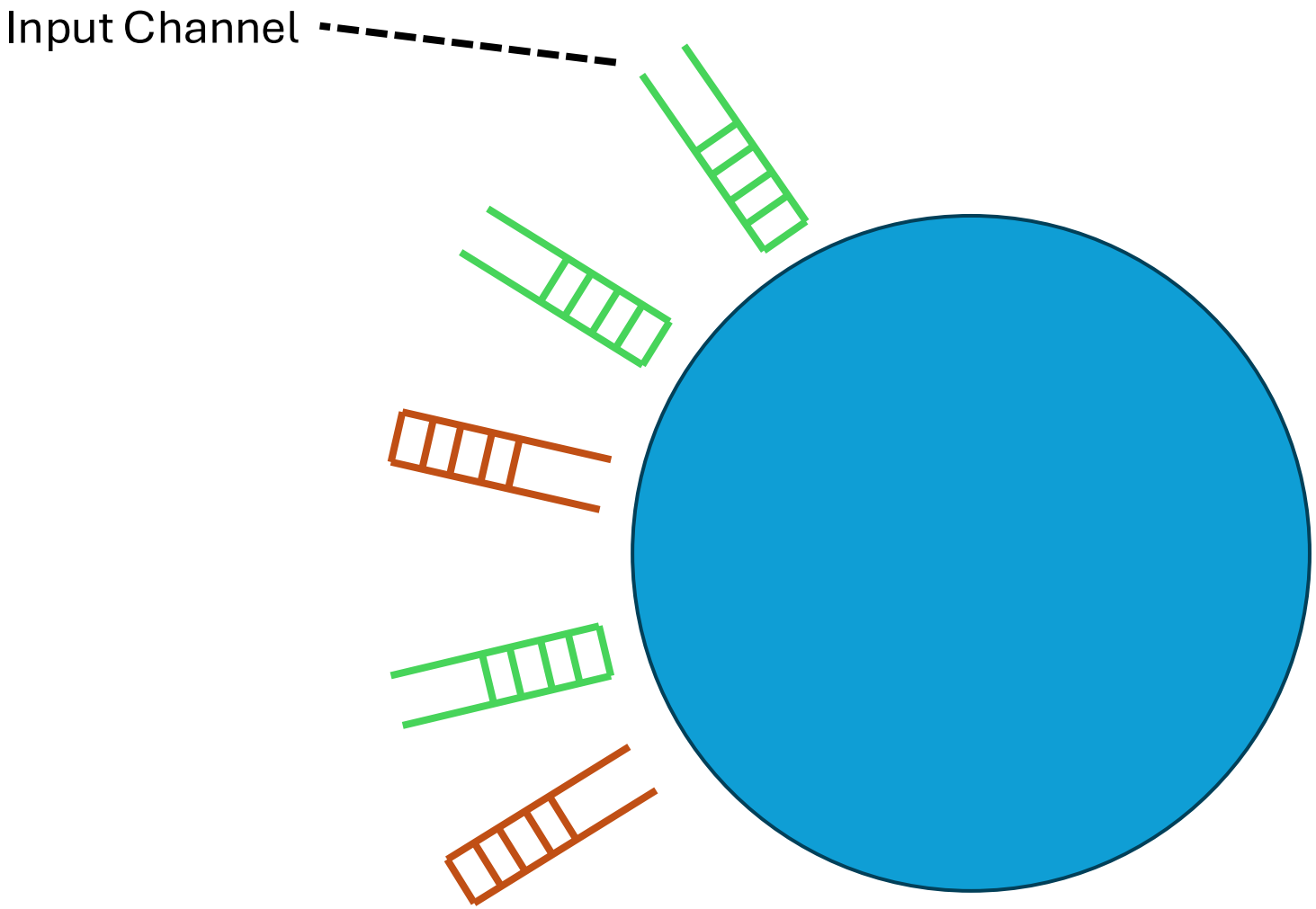
A Simple System Model



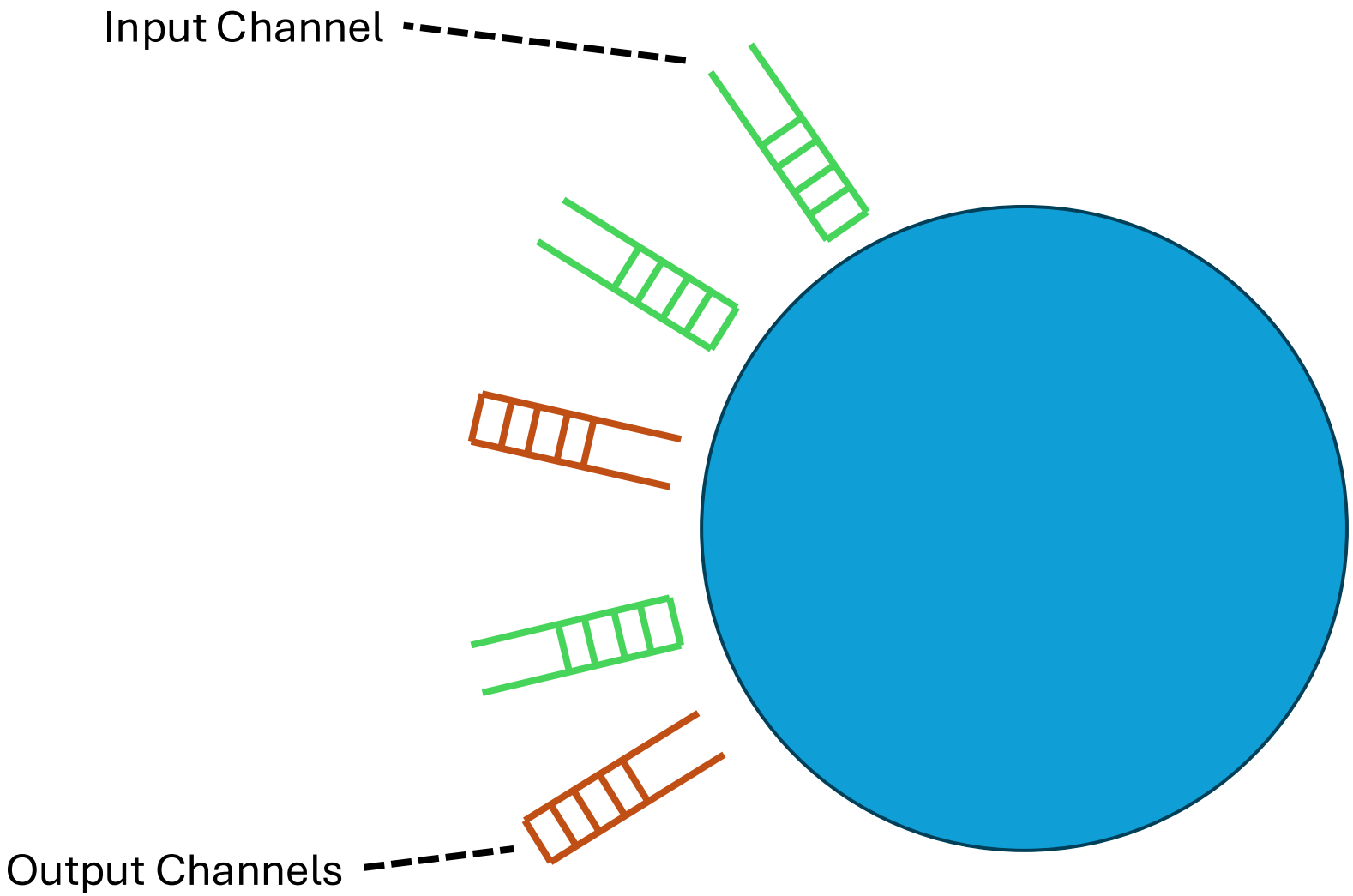
A Simple System Model



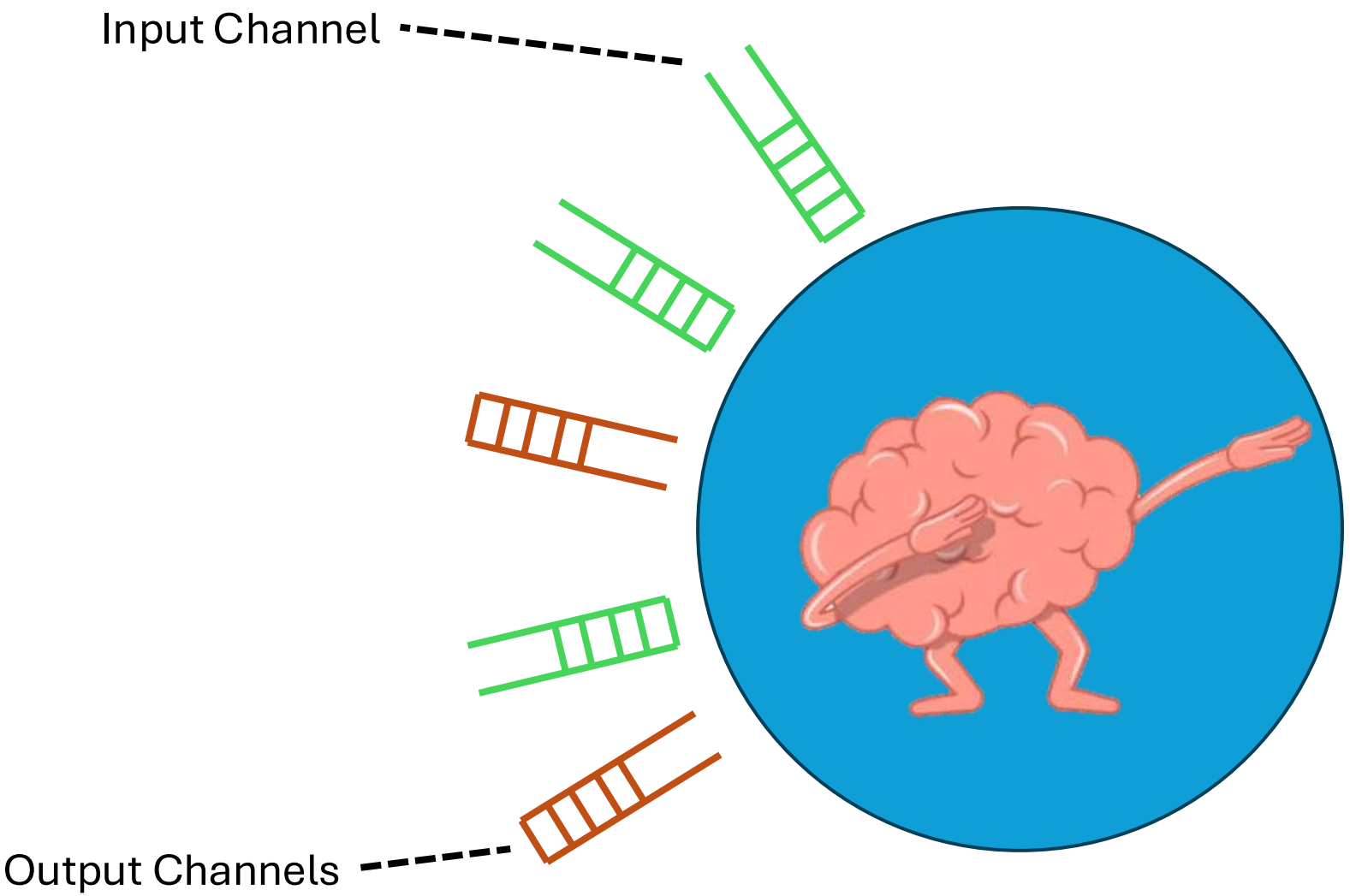
A Simple System Model



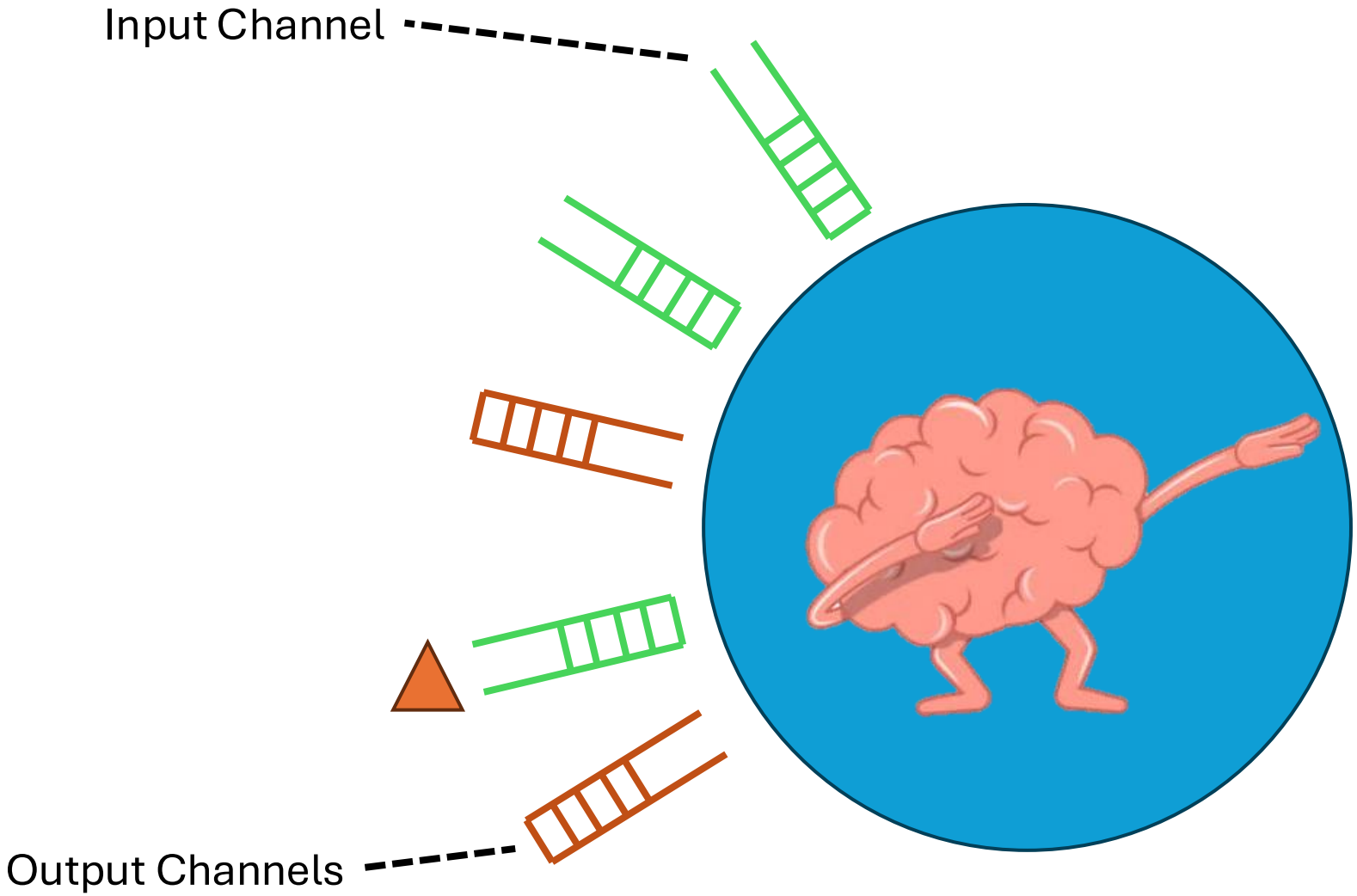
A Simple System Model



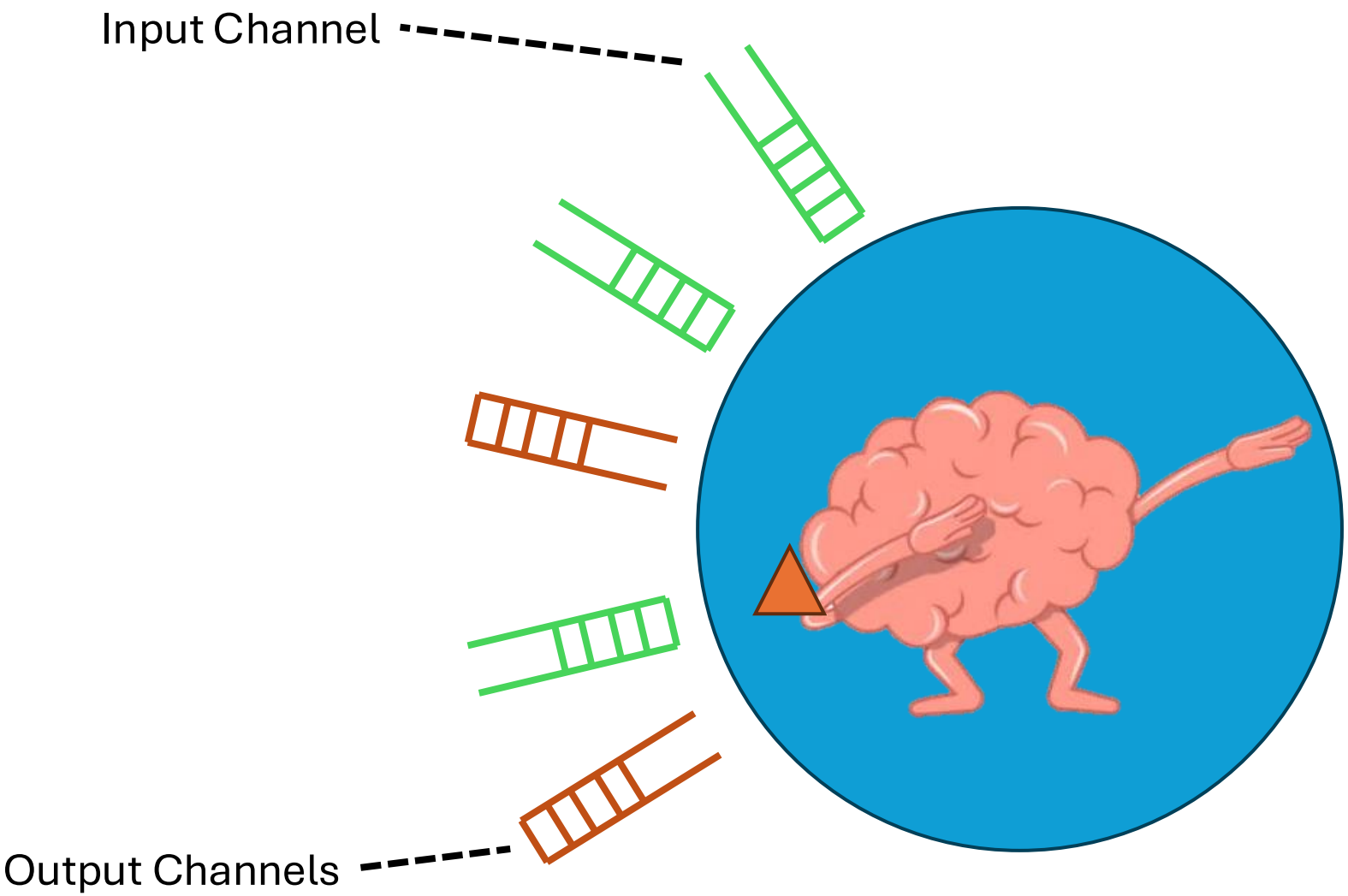
A Simple System Model



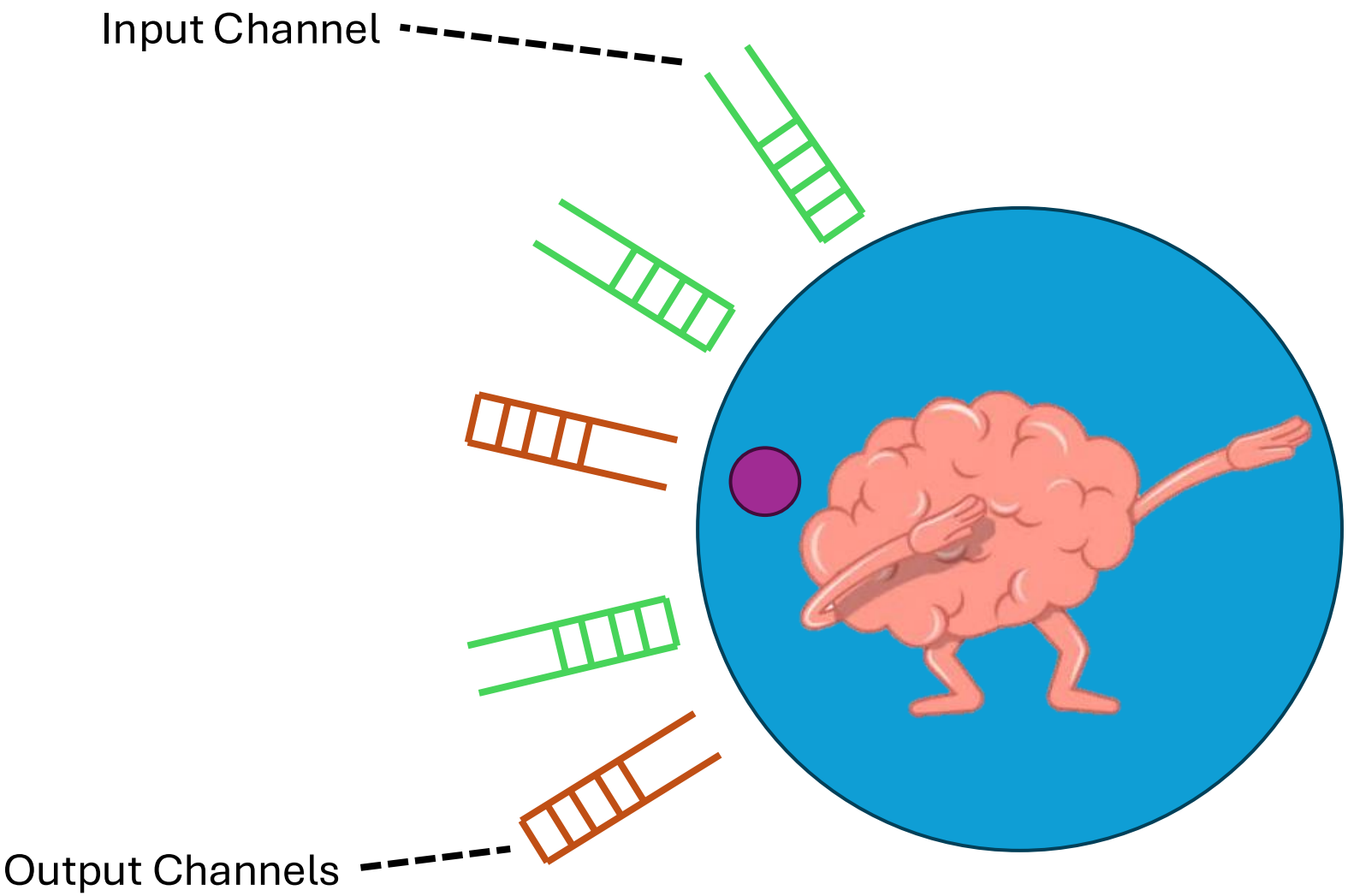
A Simple System Model



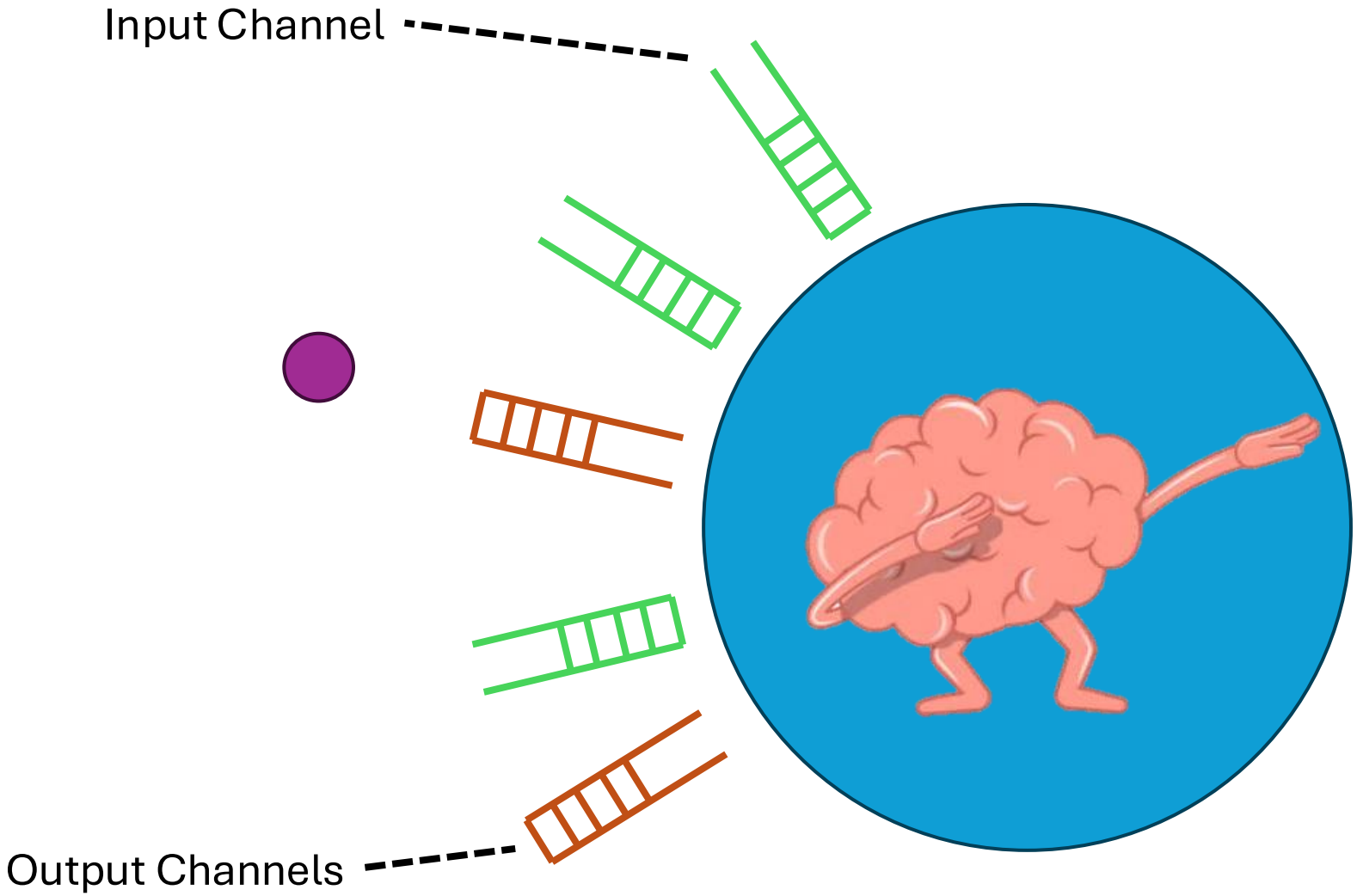
A Simple System Model



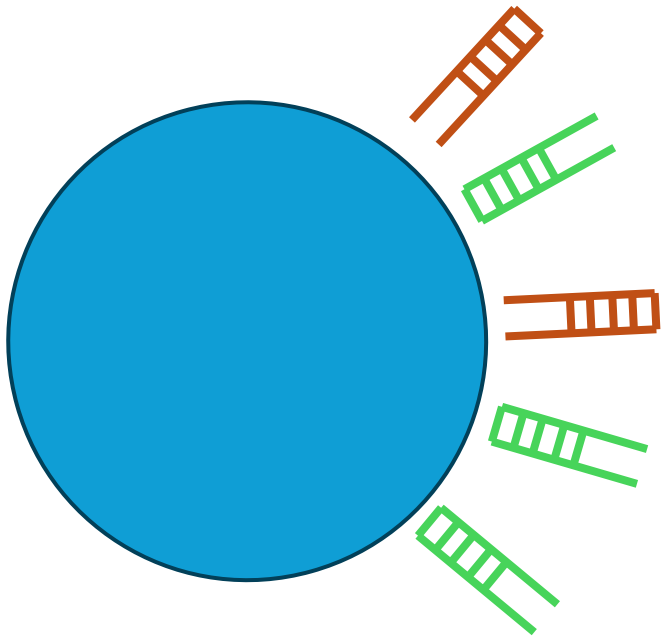
A Simple System Model



A Simple System Model

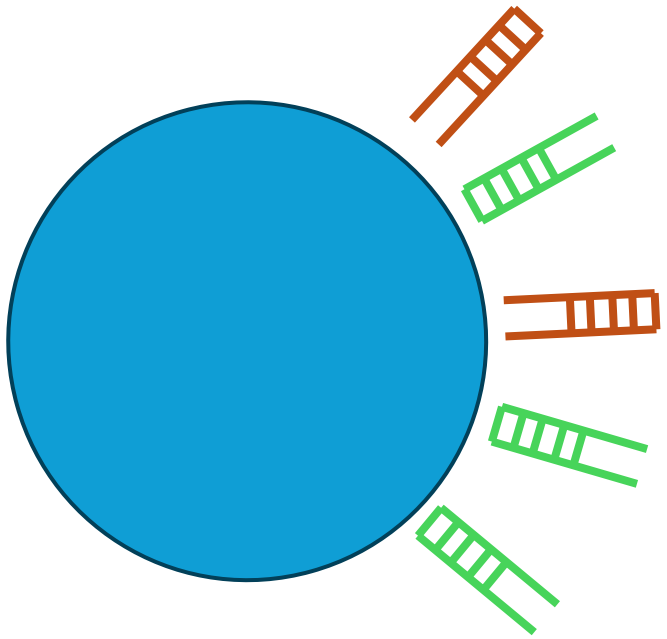


More Precisely…



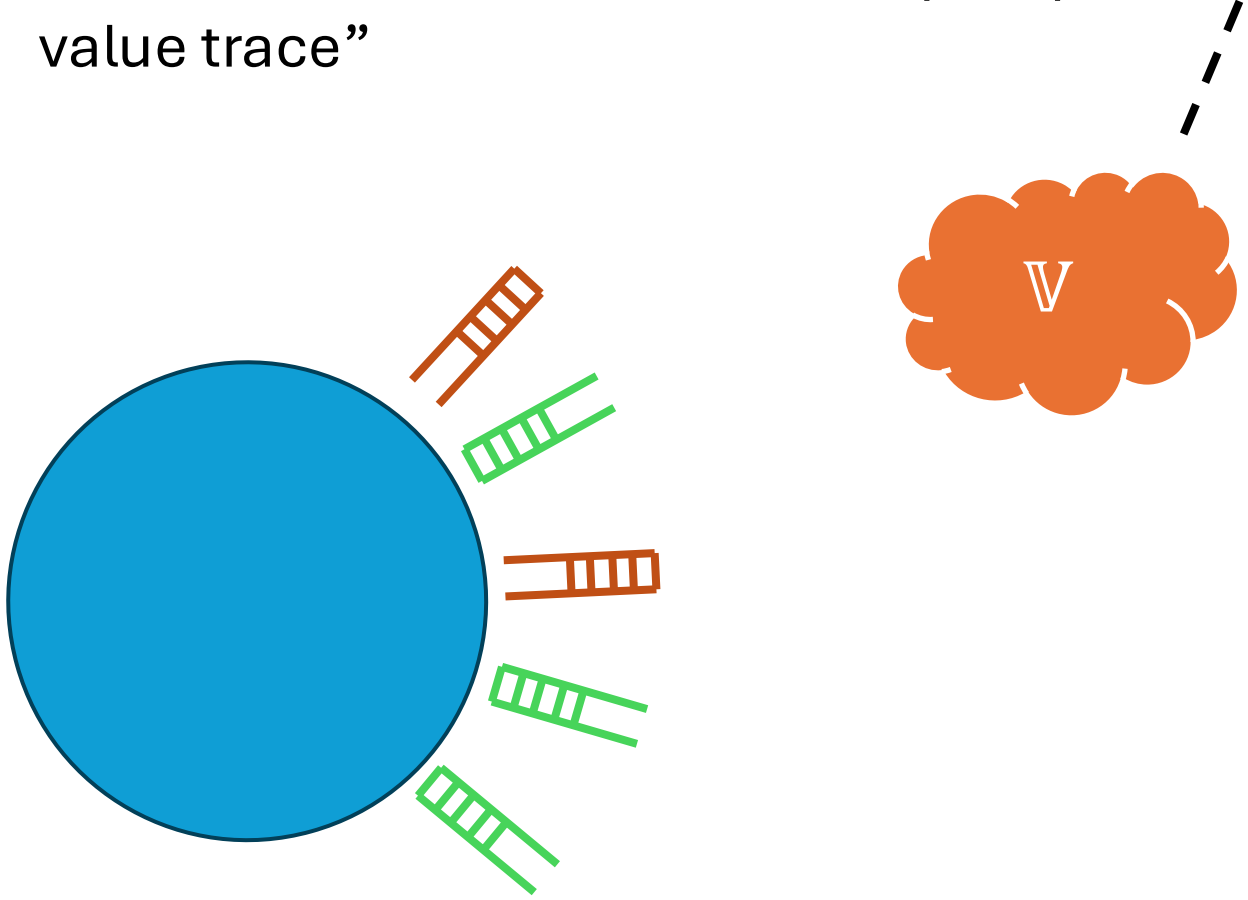
More Precisely…

“Mathematical function that maps input value history at channels to output value trace”



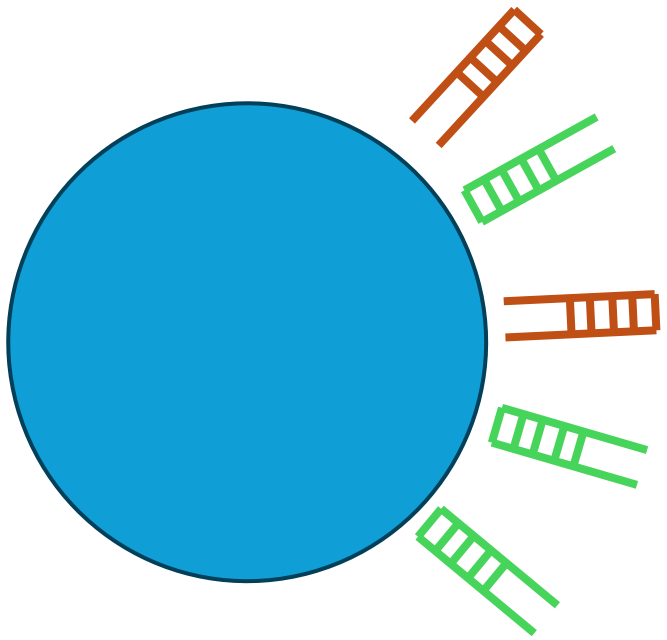
More Precisely…

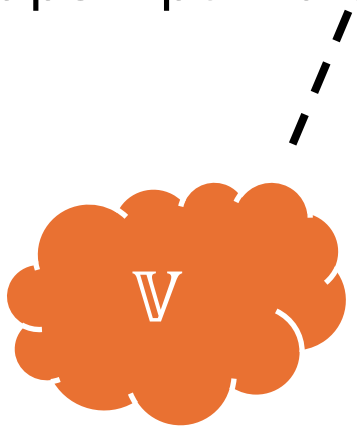
“Mathematical function that maps input value history at channels to output value trace”

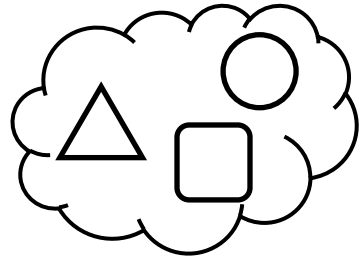


More Precisely…

“Mathematical function that maps input value history at channels to output value trace”

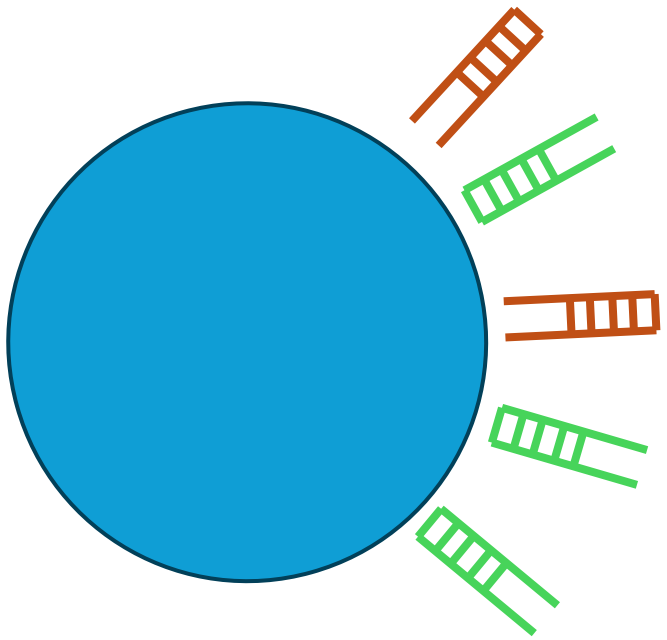


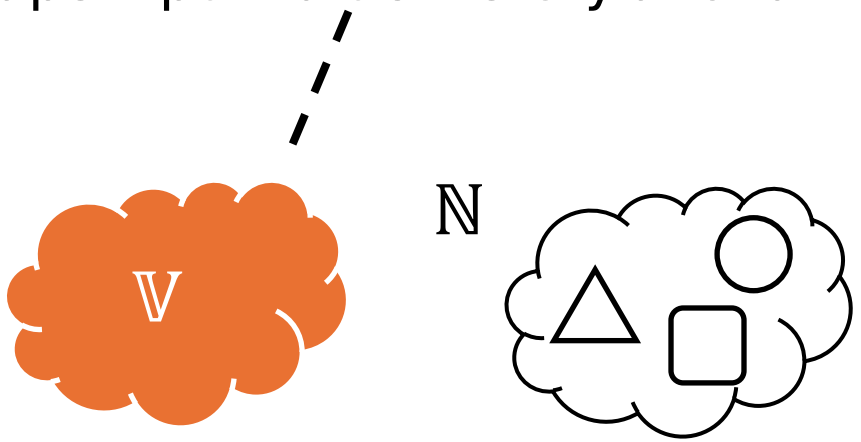




More Precisely…

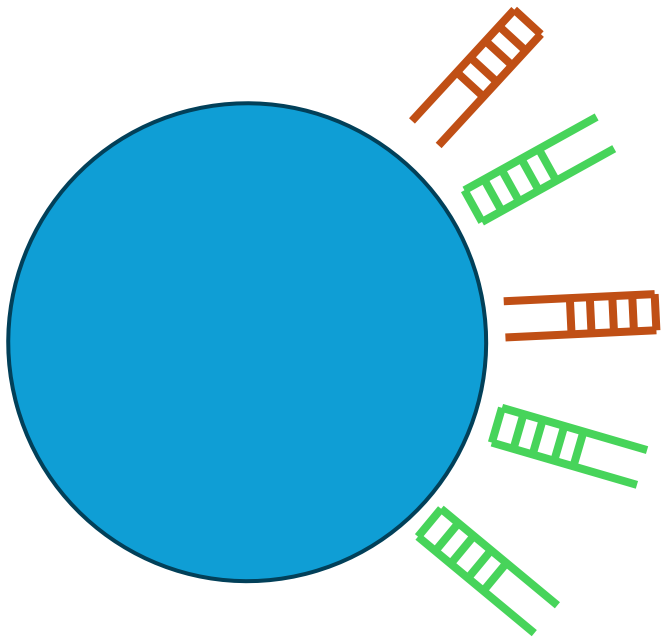
“Mathematical function that maps input value history at channels to output value trace”

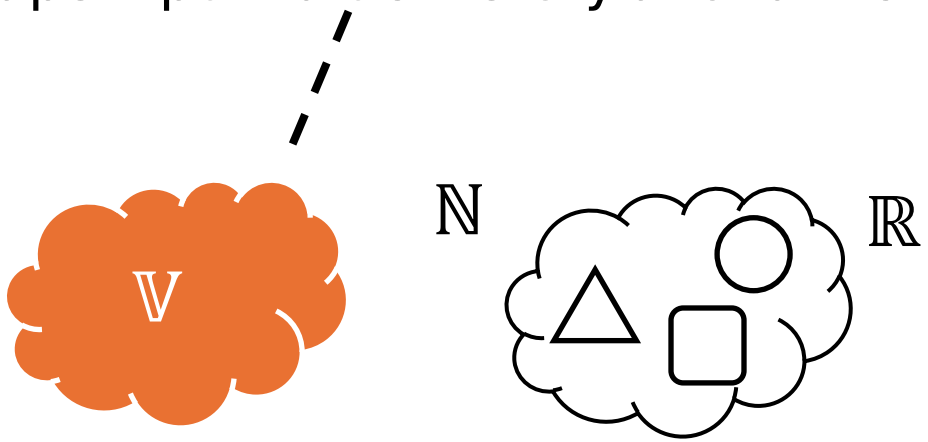




More Precisely…

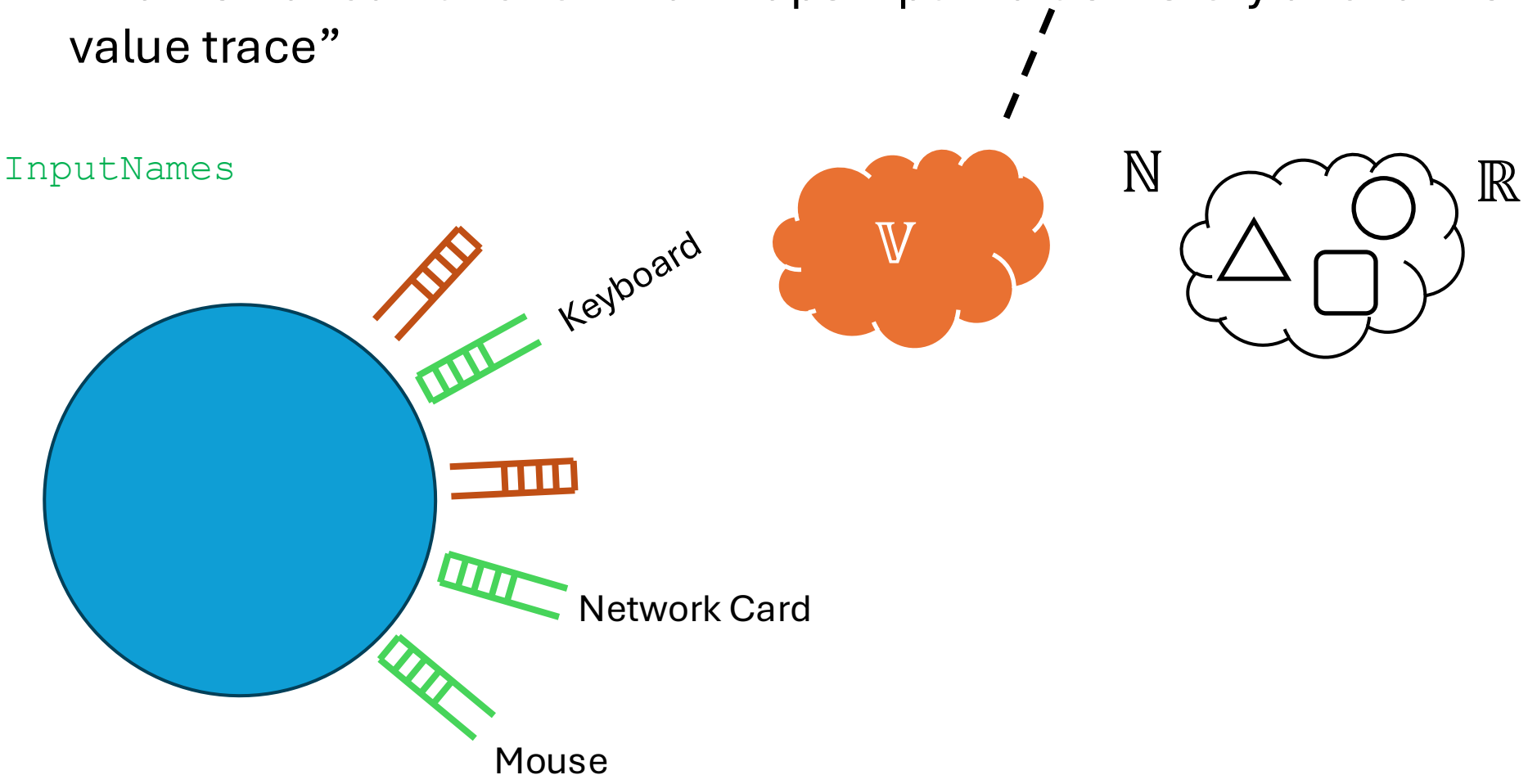
“Mathematical function that maps input value history at channels to output value trace”





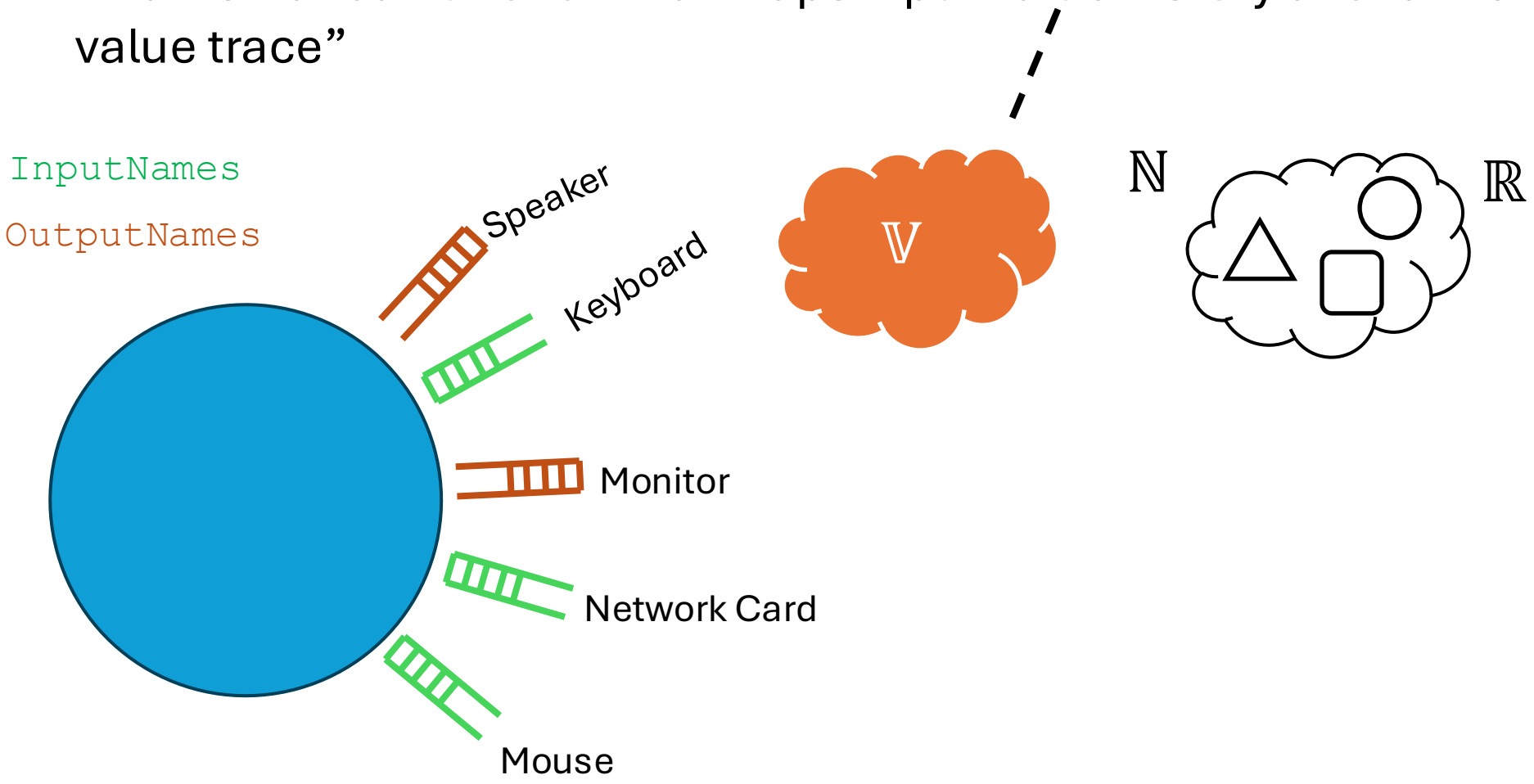
More Precisely…

"Mathematical function that maps input value history at channels to output value trace"



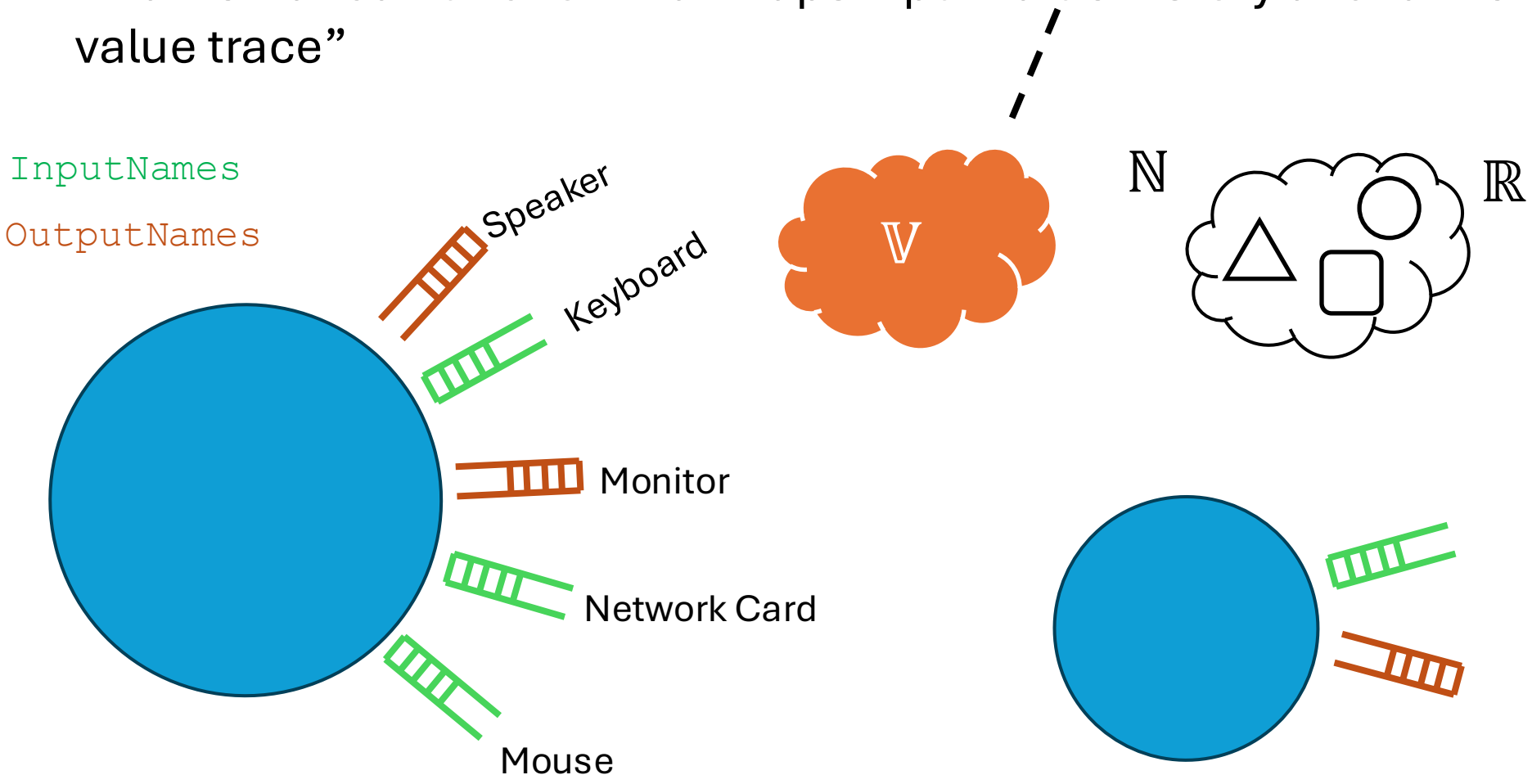
More Precisely…

"Mathematical function that maps input value history at channels to output value trace"



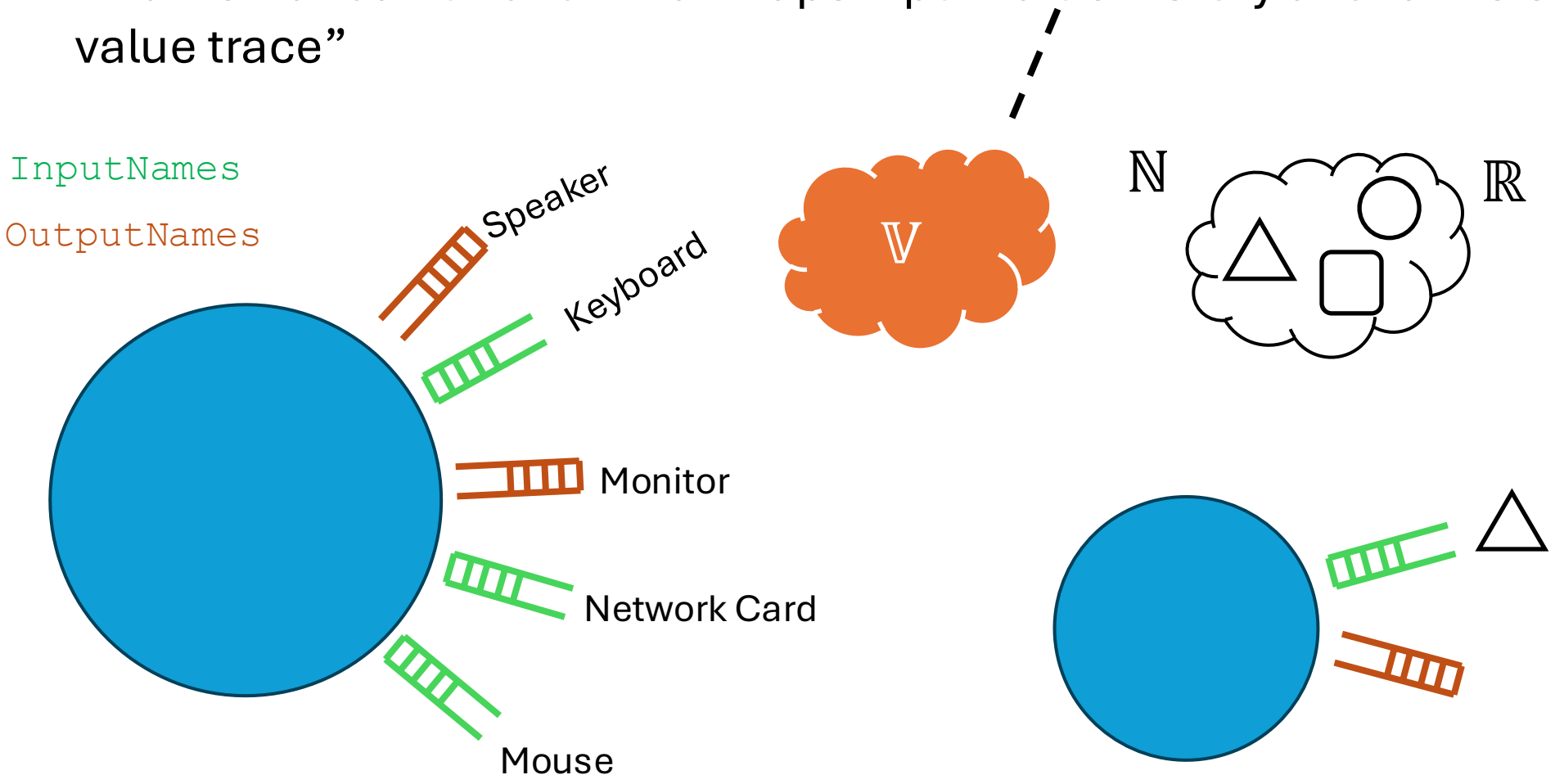
More Precisely…

“Mathematical function that maps input value history at channels to output value trace”



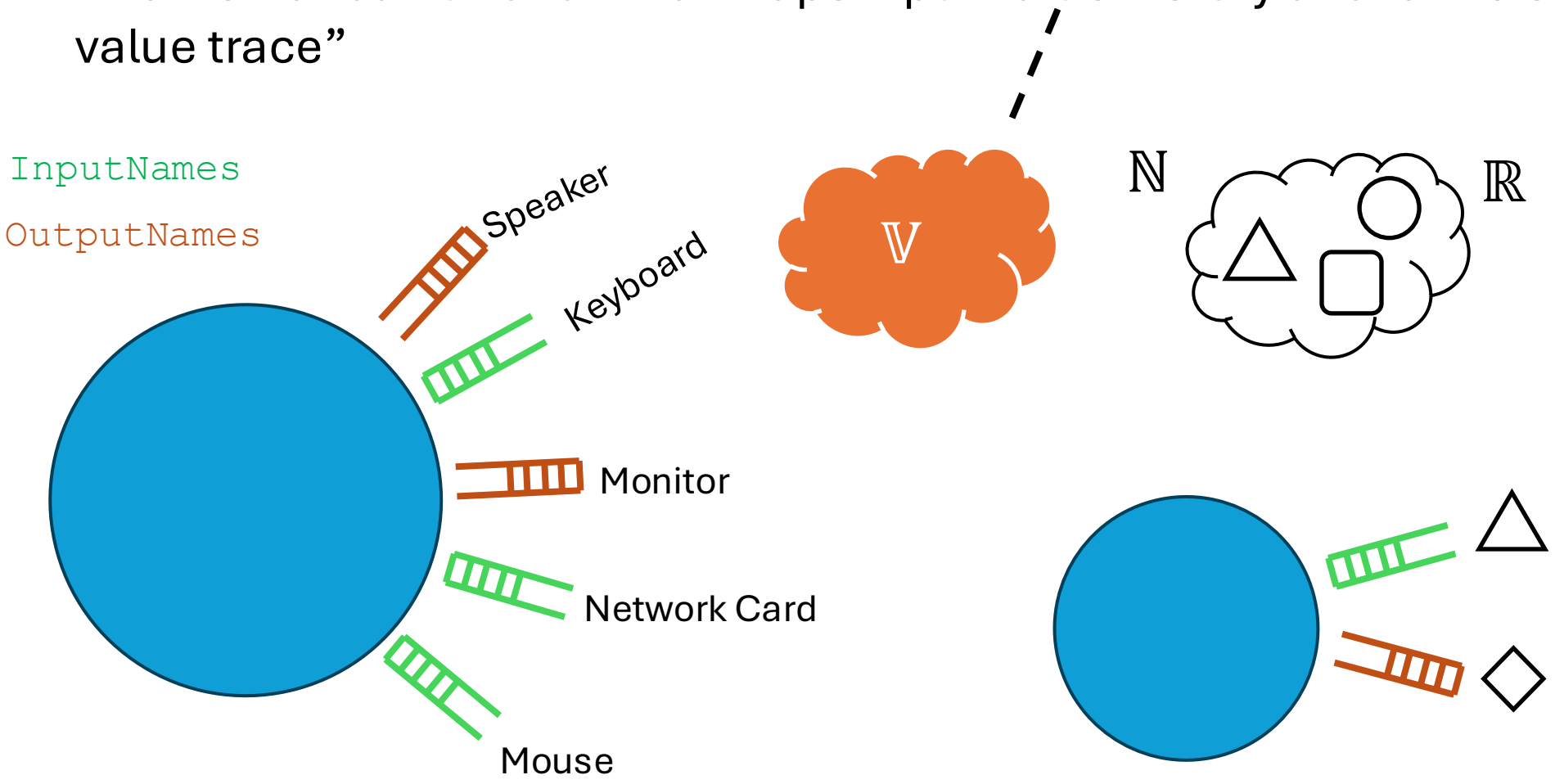
More Precisely…

"Mathematical function that maps input value history at channels to output value trace"



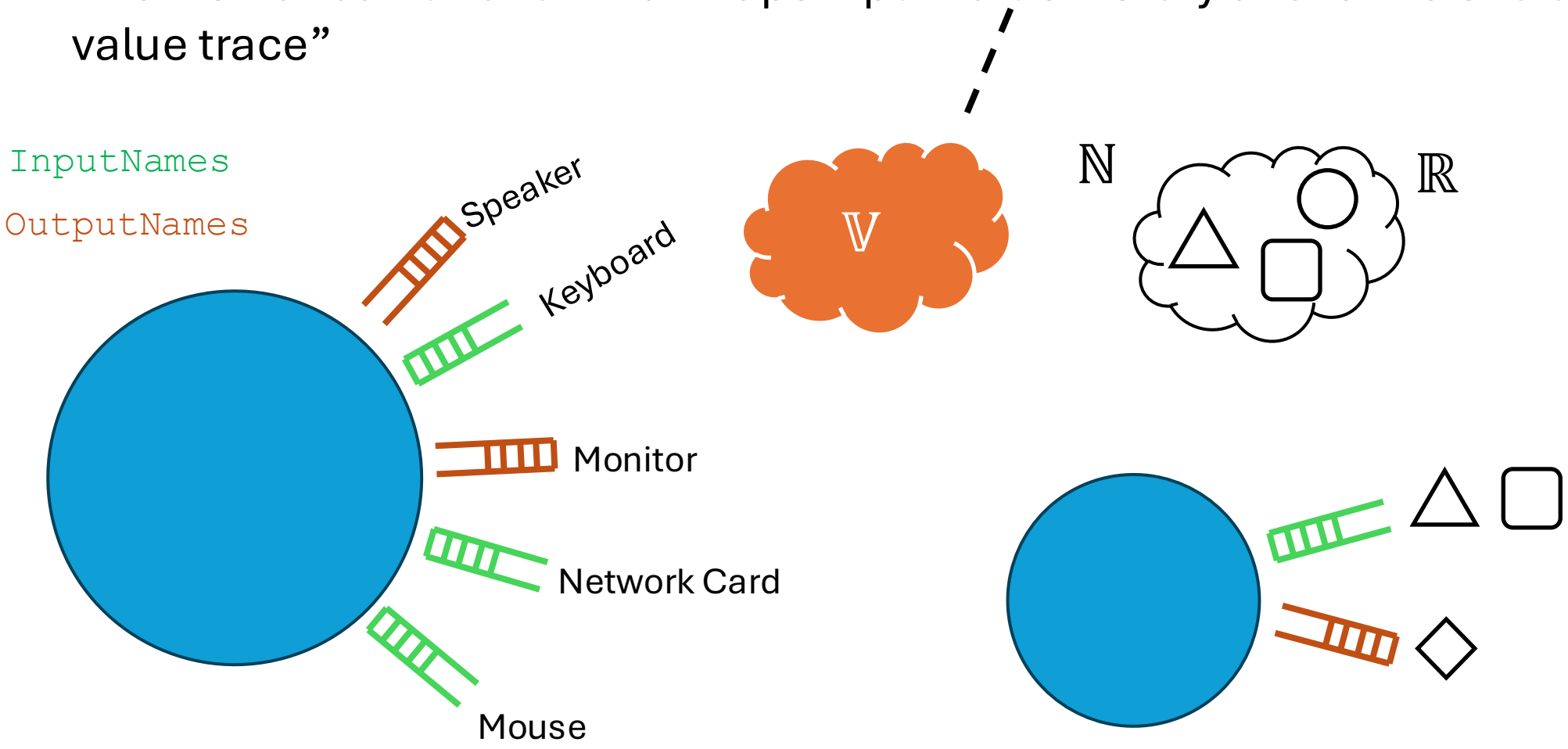
More Precisely…

"Mathematical function that maps input value history at channels to output value trace"



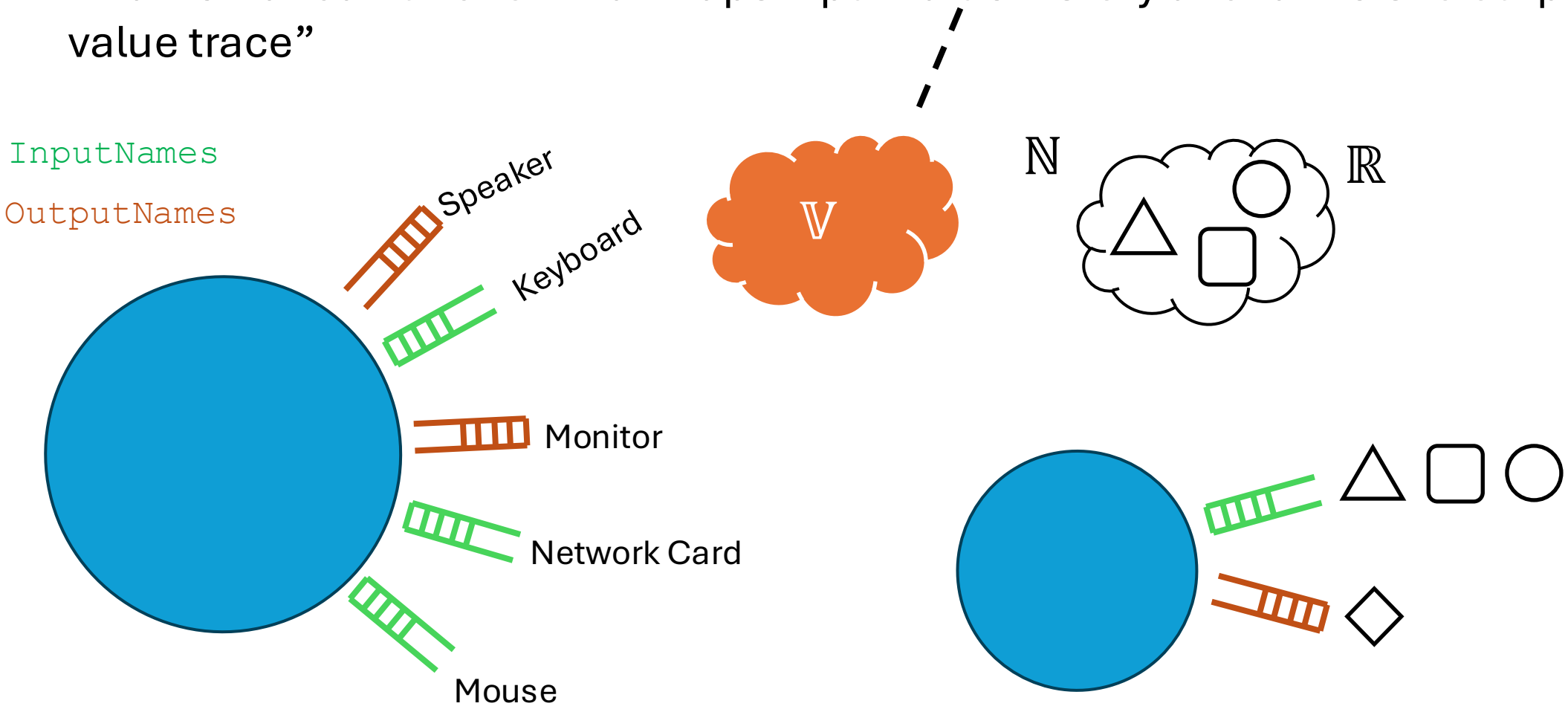
More Precisely…

“Mathematical function that maps input value history at channels to output value trace”



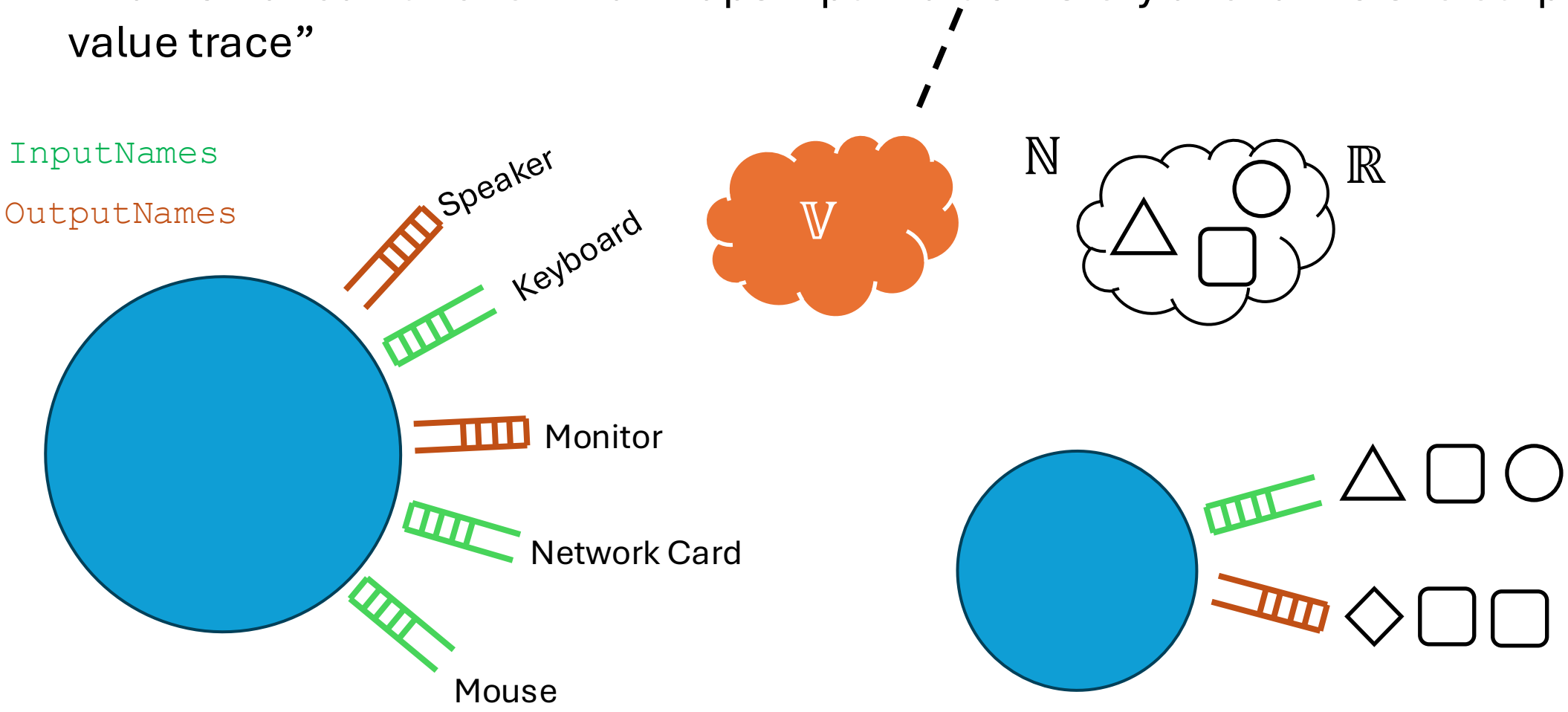
More Precisely…

"Mathematical function that maps input value history at channels to output value trace"



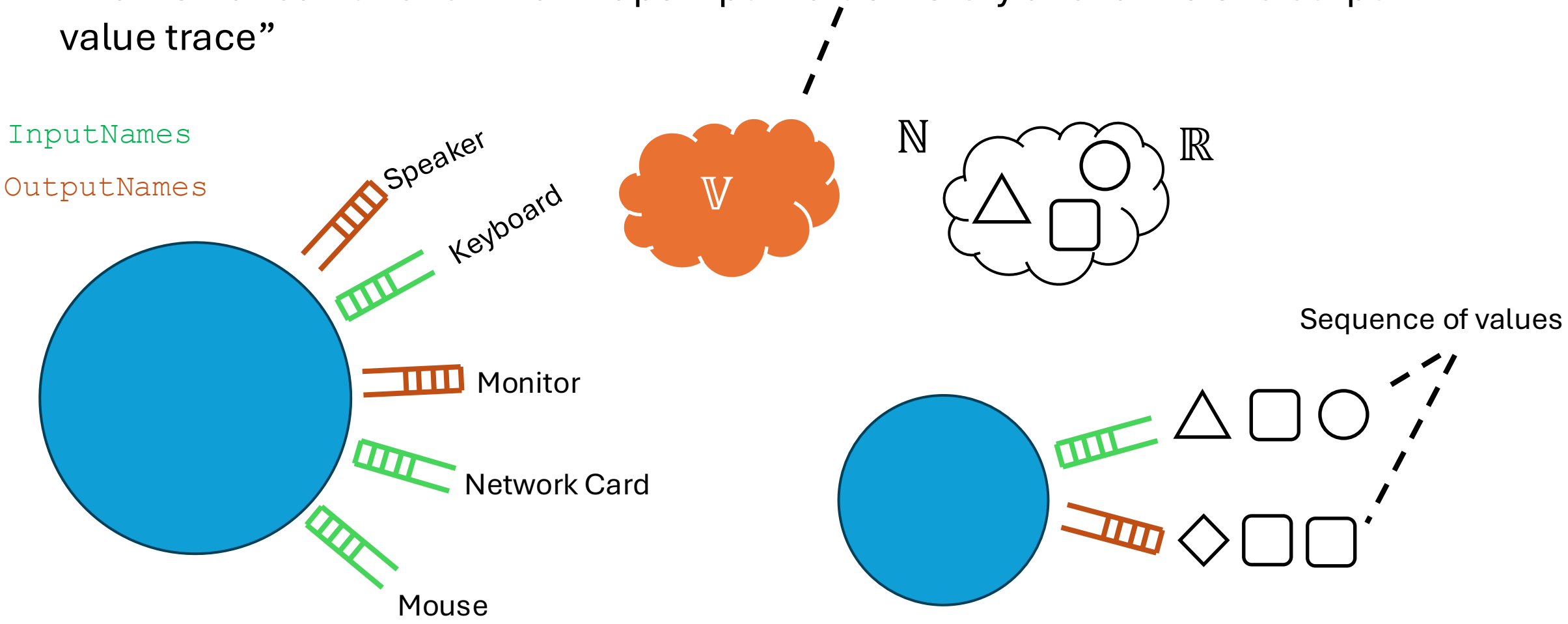
More Precisely…

“Mathematical function that maps input value history at channels to output value trace”



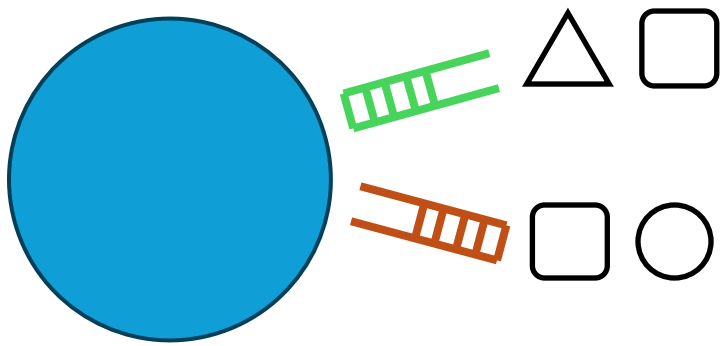
More Precisely…

"Mathematical function that maps input value history at channels to output value trace"

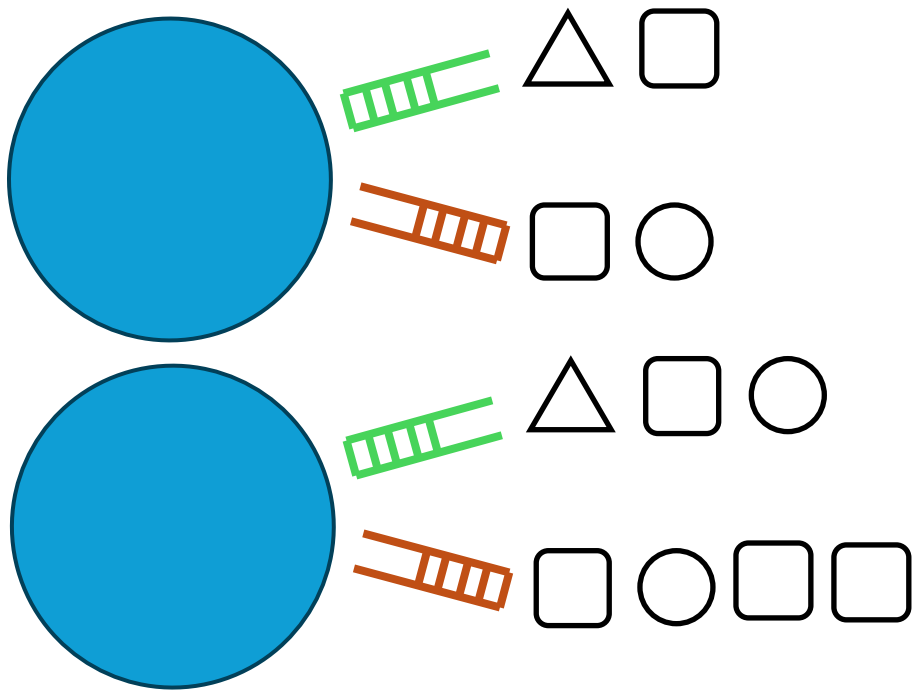


Not any function...

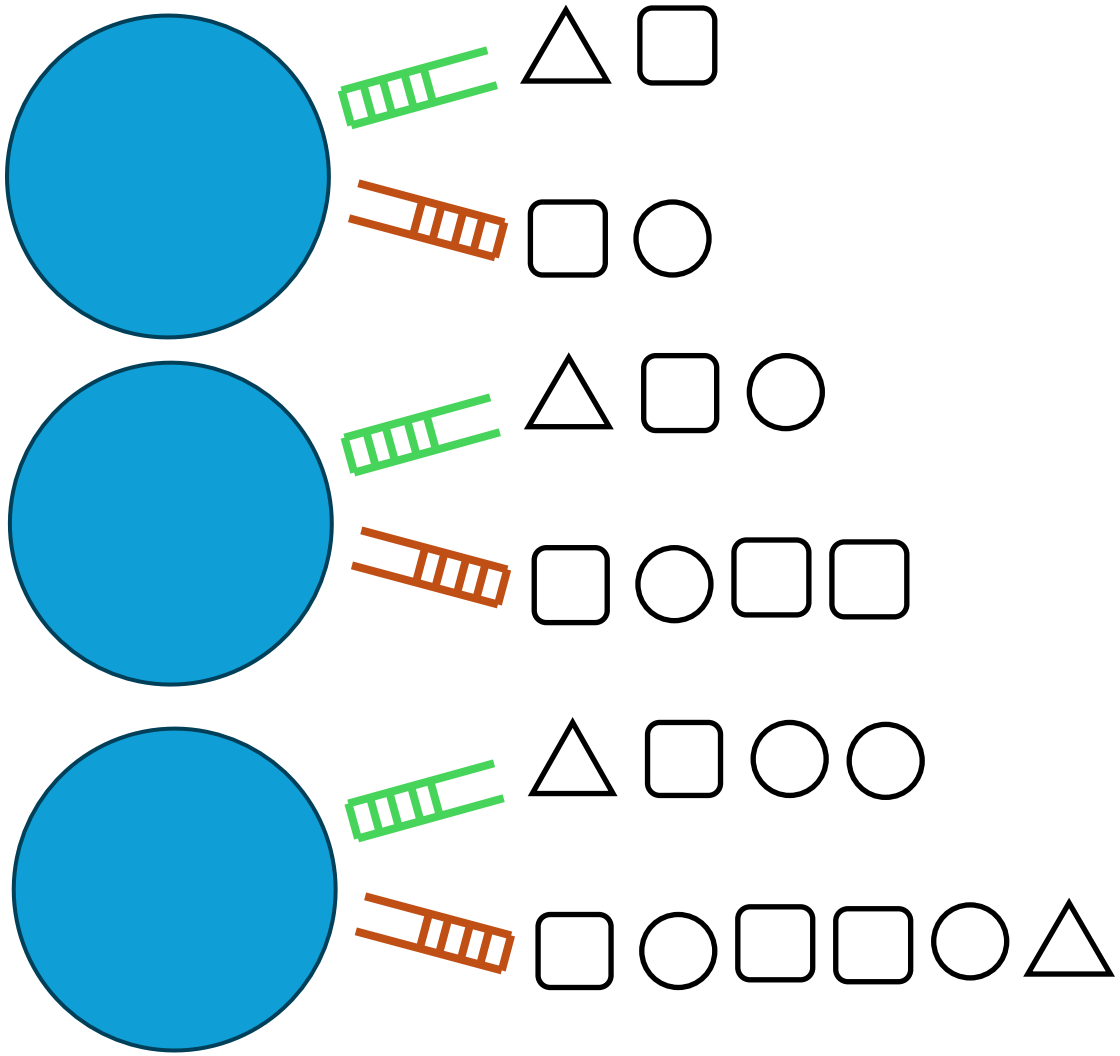
Not any function...



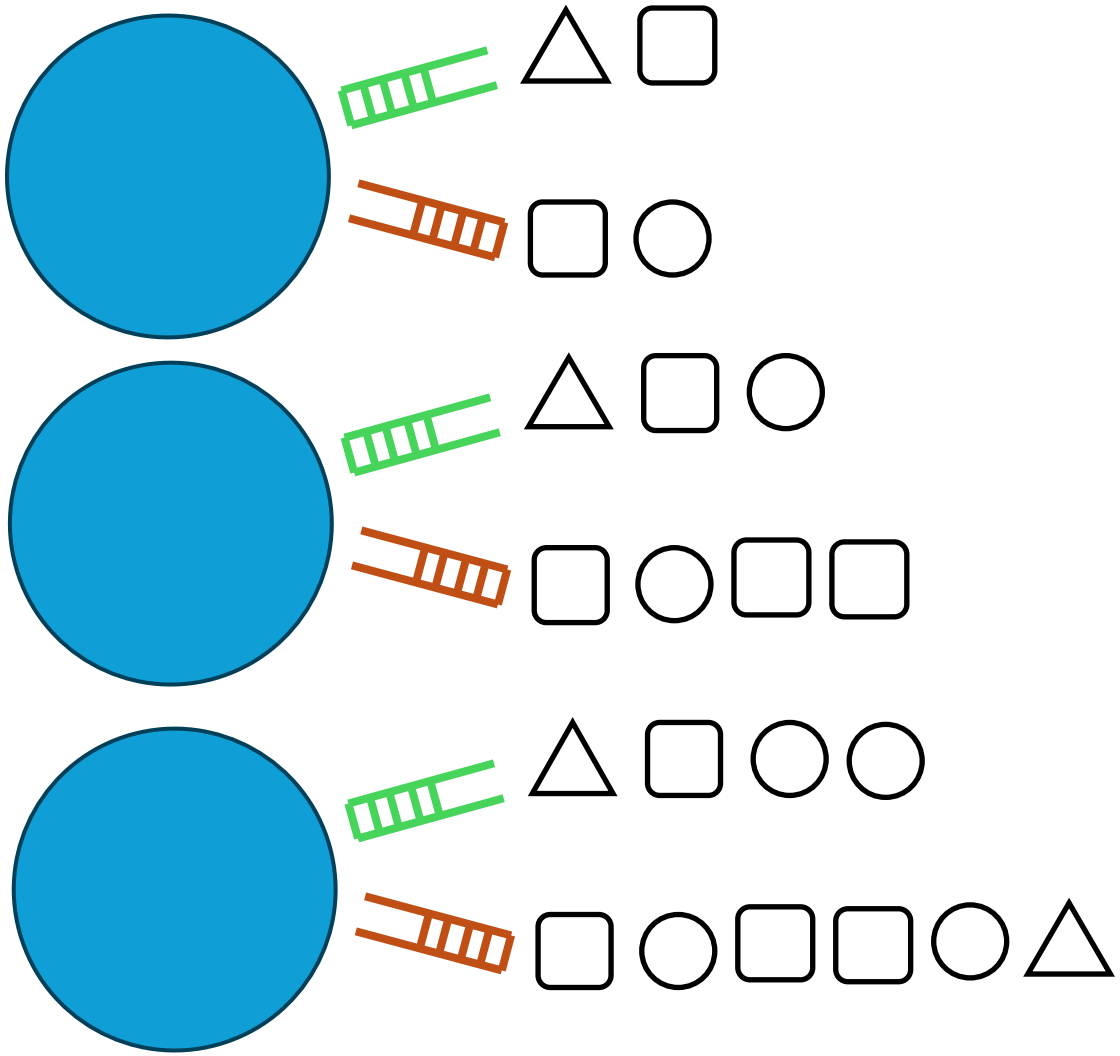
Not any function...

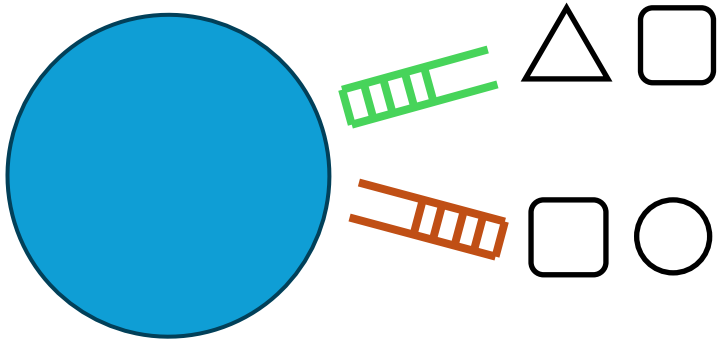


Not any function...

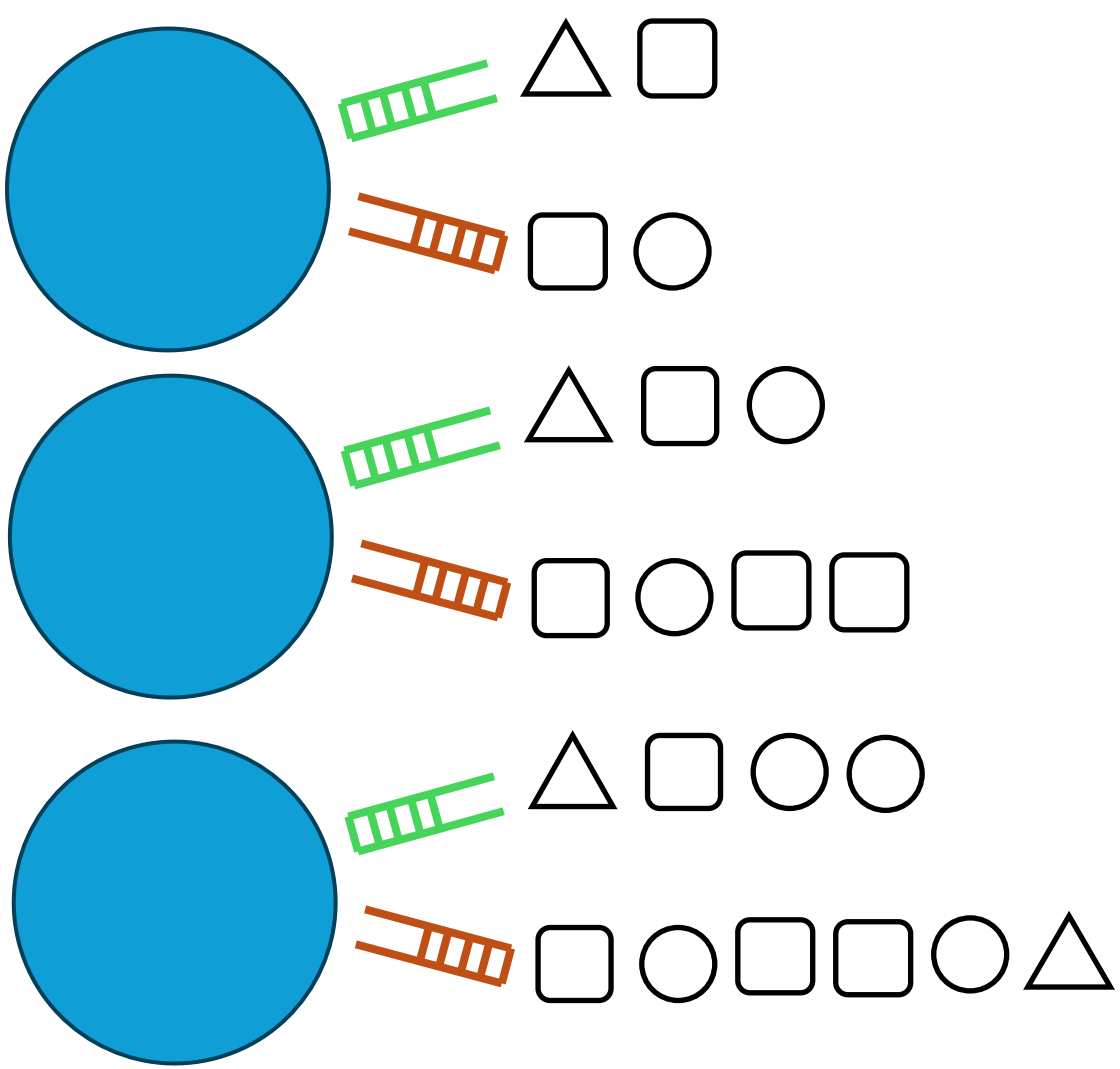


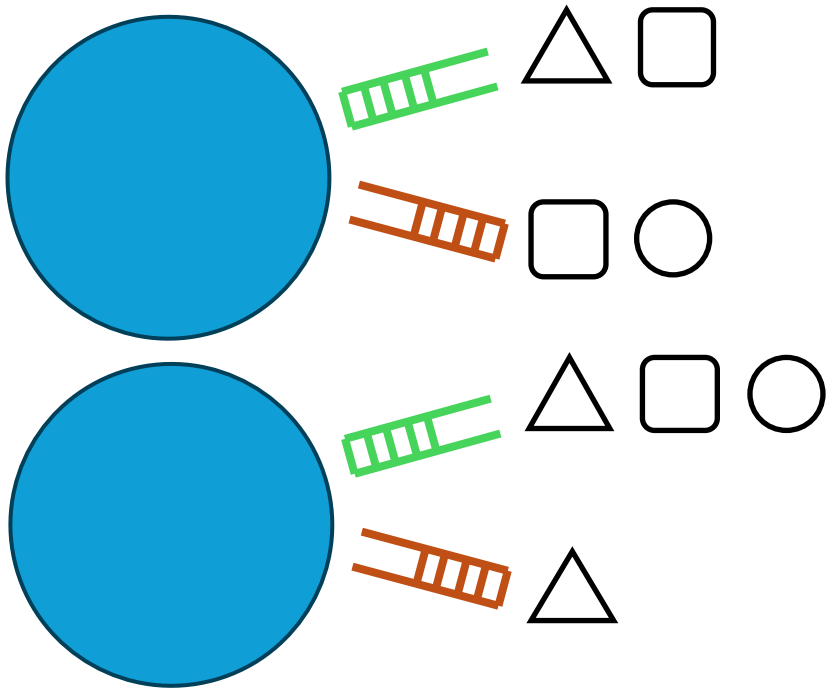
Not any function...



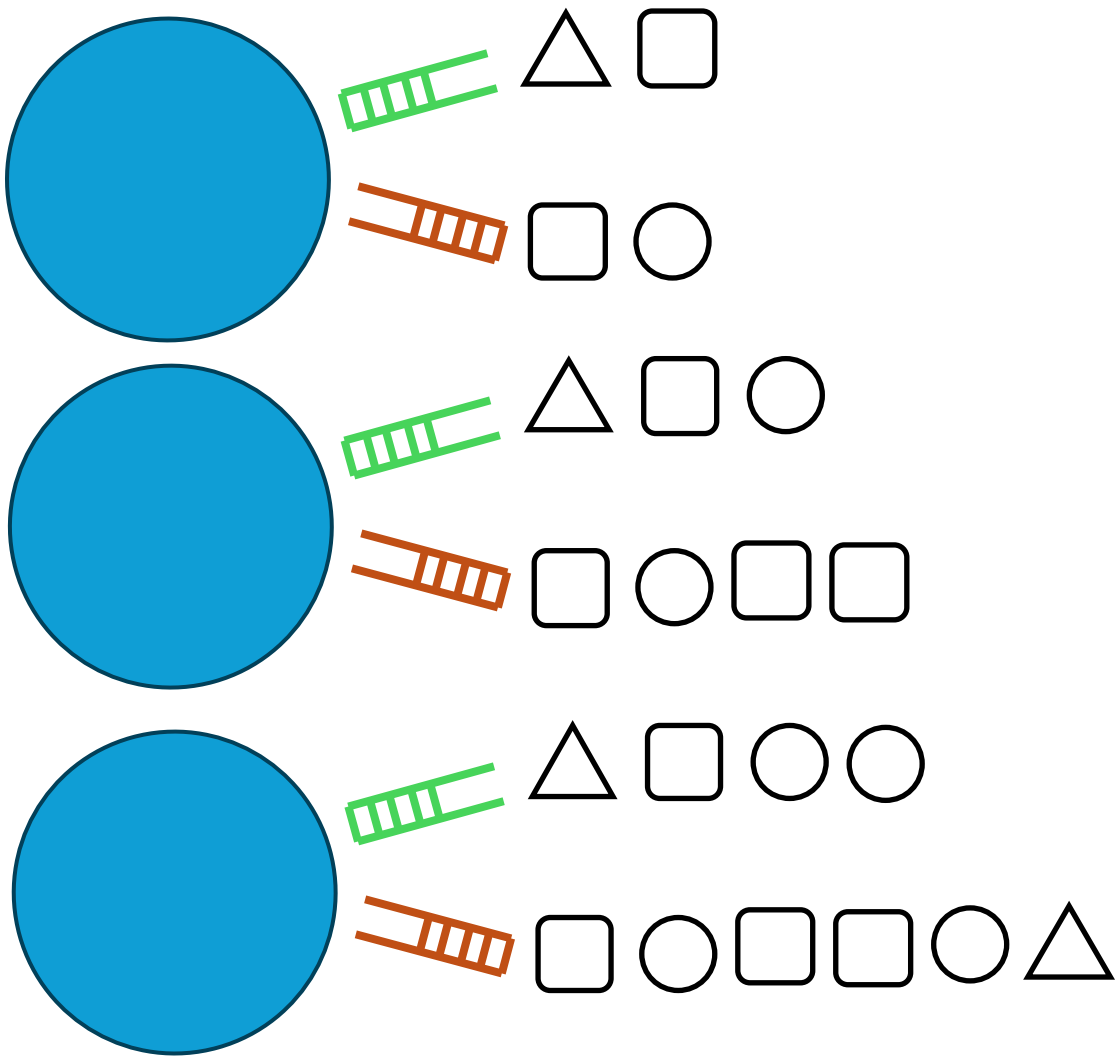


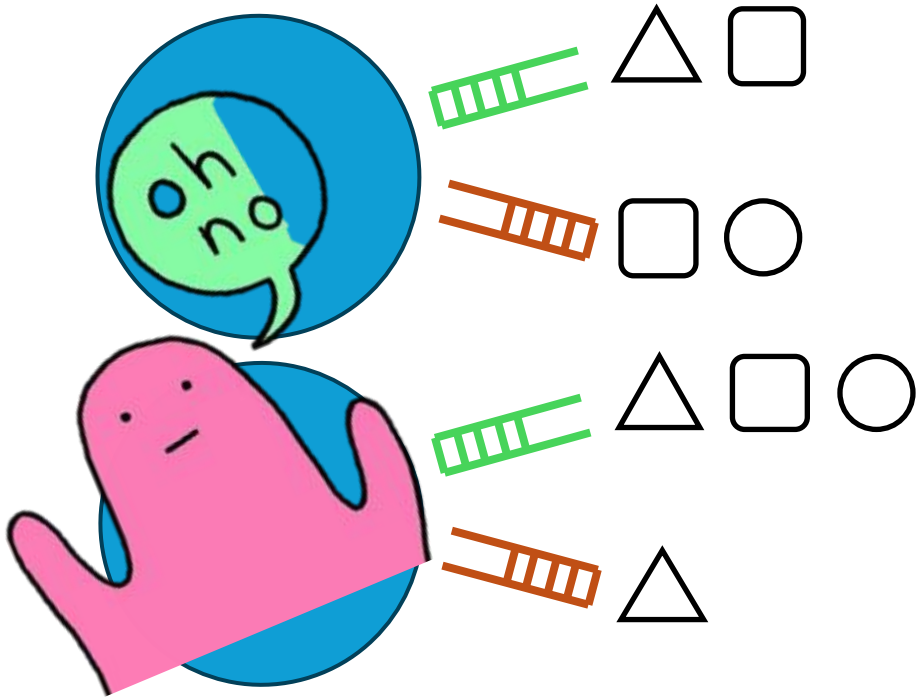
Not any function...



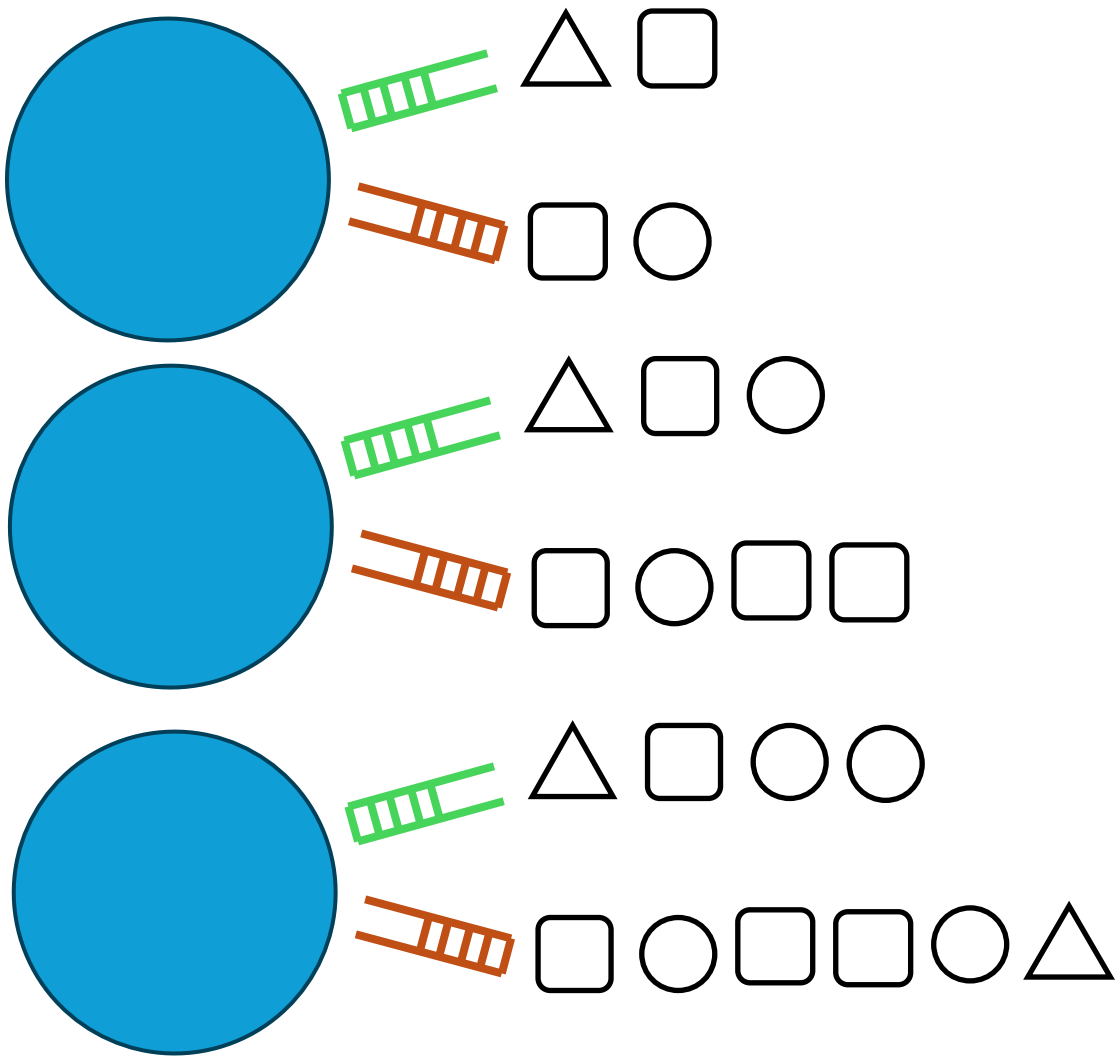


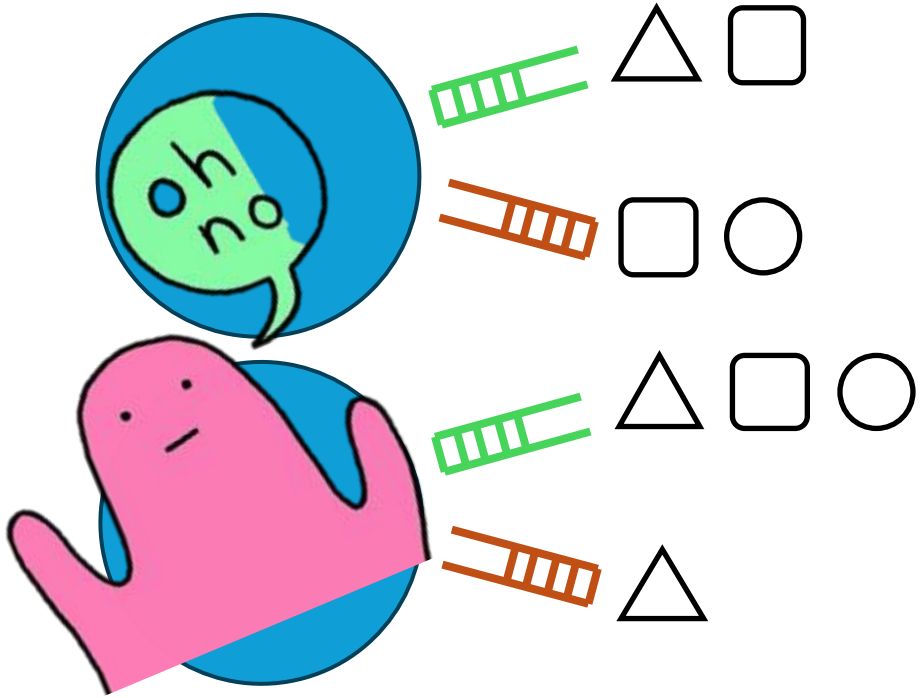
Not any function...





Not any function...





The function should be **monotone**

A Succinct Definition

A Succinct Definition

$\mathbb{V} : Set$

A Succinct Definition

$\mathbb{V} : Set$ $\quad\quad$ $InputNames : FiniteSet$

A Succinct Definition

$\mathbb{V} : Set$ $\quad$ $InputNames : FiniteSet$ $\quad$ $OutputNames : FiniteSet$

A Succinct Definition

$\mathbb{V} : Set$ $\quad$ $InputNames : FiniteSet$ $\quad$ $OutputNames : FiniteSet$

$ValueSequence \triangleq Seq\ \mathbb{V}$

A Succinct Definition

$\mathbb{V} : Set$ $\quad$ $InputNames : FiniteSet$ $\quad$ $OutputNames : FiniteSet$

$ValueSequence \triangleq Seq\ \mathbb{V}$

$InputHistory \triangleq InputNames \to ValueSequence$

A Succinct Definition

$\mathbb{V} : Set$ $\quad$ $InputNames : FiniteSet$ $\quad$ $OutputNames : FiniteSet$

$ValueSequence \triangleq Seq\ \mathbb{V}$

$InputHistory \triangleq InputNames \to ValueSequence$
$OutputTrace \triangleq OutputNames \to ValueSequence$

A Succinct Definition

$\mathbb{V} : Set \qquad InputNames : FiniteSet \qquad OutputNames : FiniteSet$

$ValueSequence \triangleq Seq\ \mathbb{V}$

$InputHistory \triangleq InputNames \to ValueSequence$
$OutputTrace \triangleq OutputNames \to ValueSequence$

$System \triangleq \{f : InputHistory \to OutputTrace \mid f\ monotone\}$

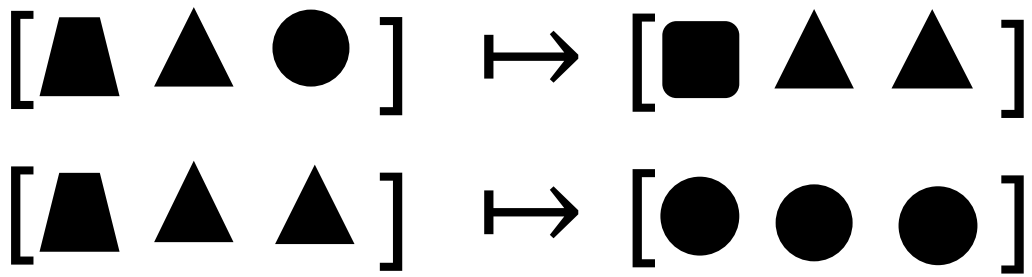
Simple, Extensional and Expressive

Simple, Extensional and Expressive



Simple, Extensional and Expressive





Simple, Extensional and Expressive



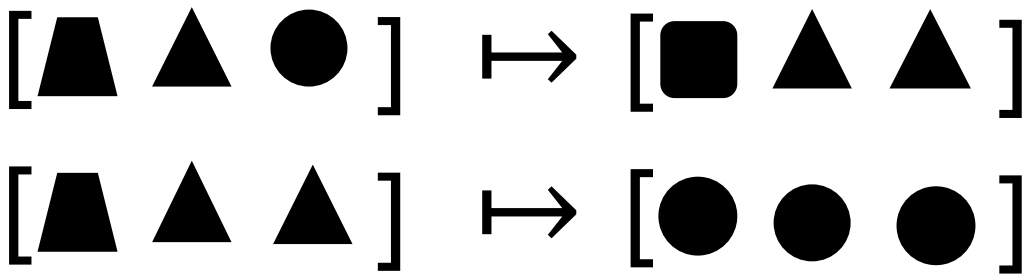
[▼ ▲ ●] ↦ [■ ▲ ▲]

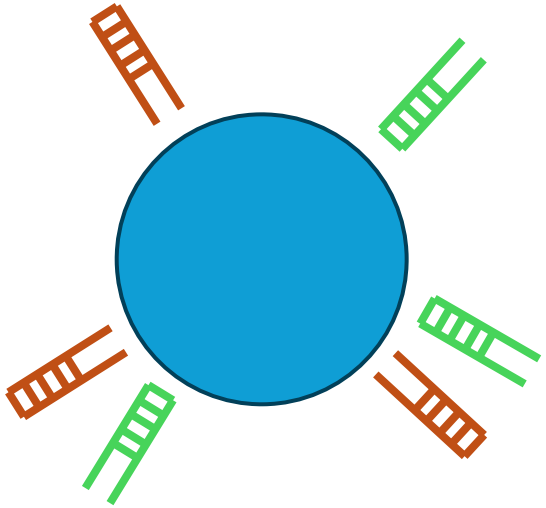
[▼ ▲ ▲] ↦ [● ● ●]



Simple, Extensional and Expressive



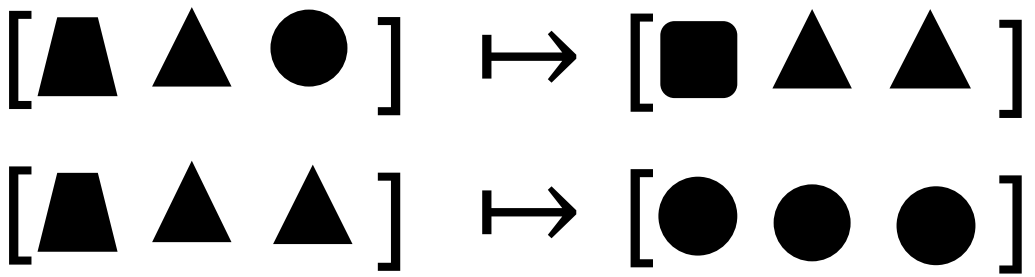


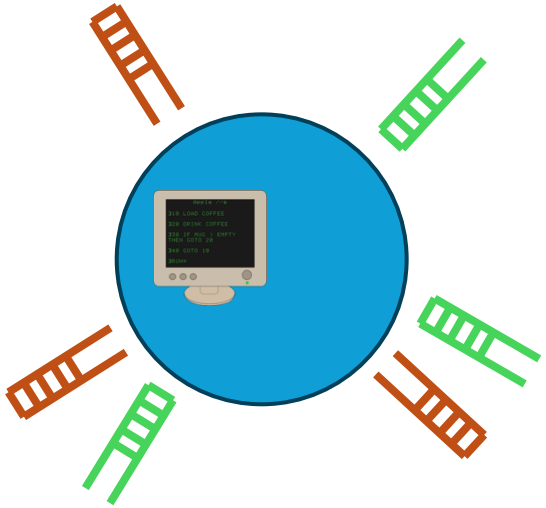




Simple, Extensional and Expressive



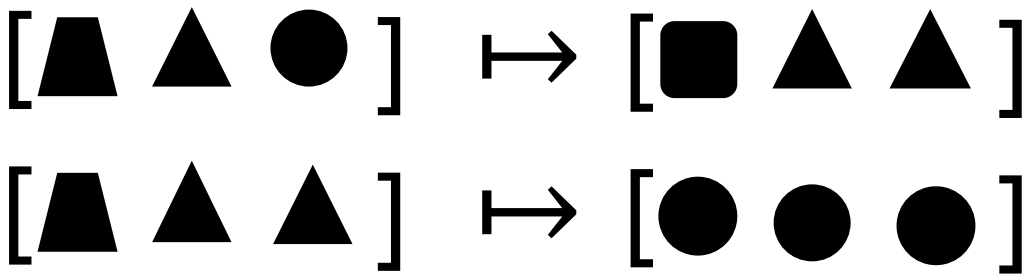


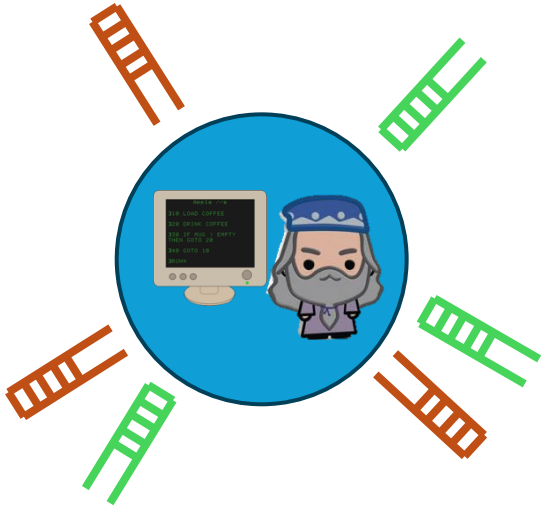




Simple, Extensional and Expressive



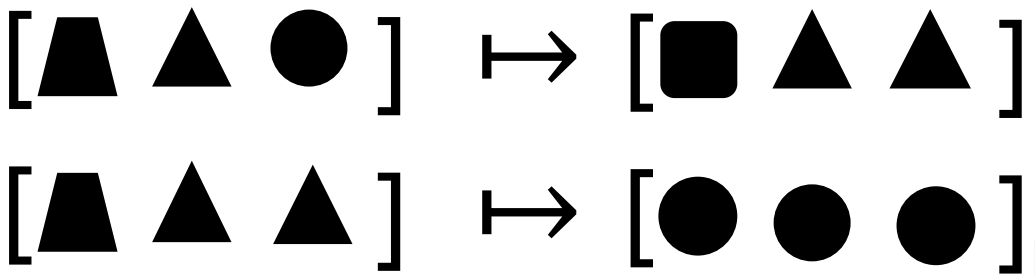


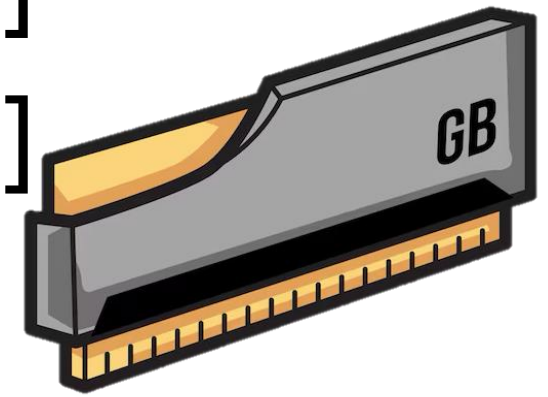


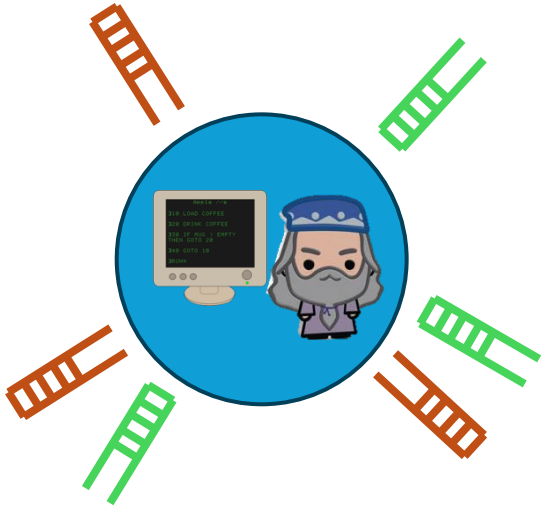


Simple, Extensional and Expressive





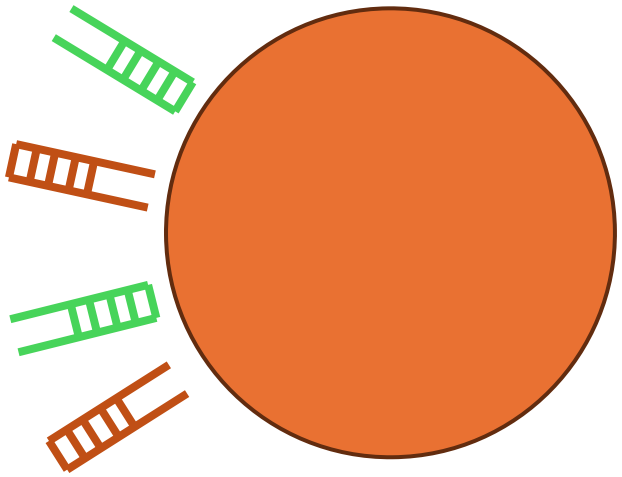




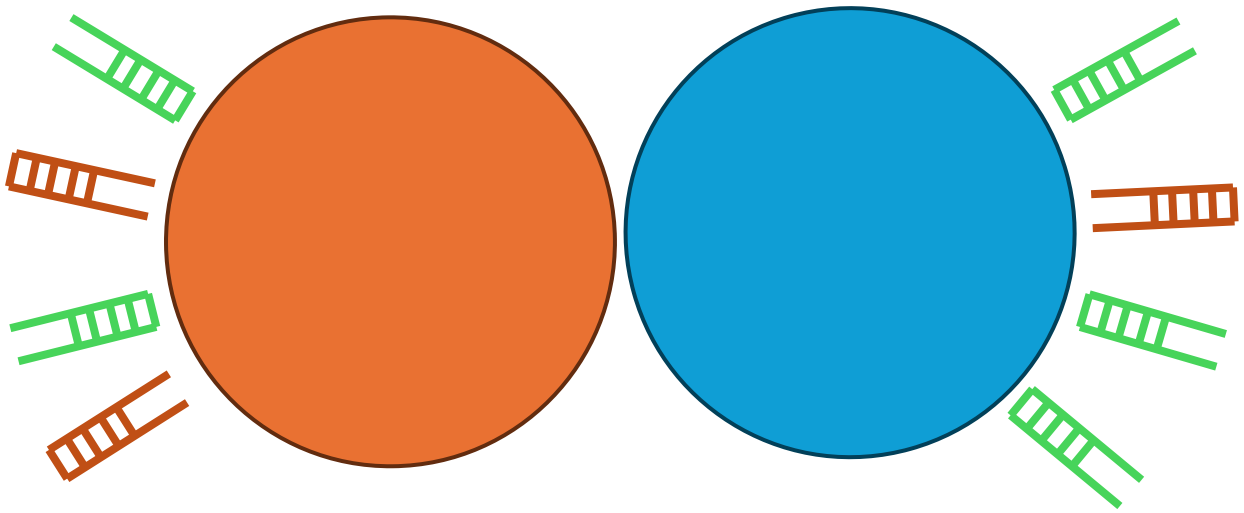


Parallel Composition

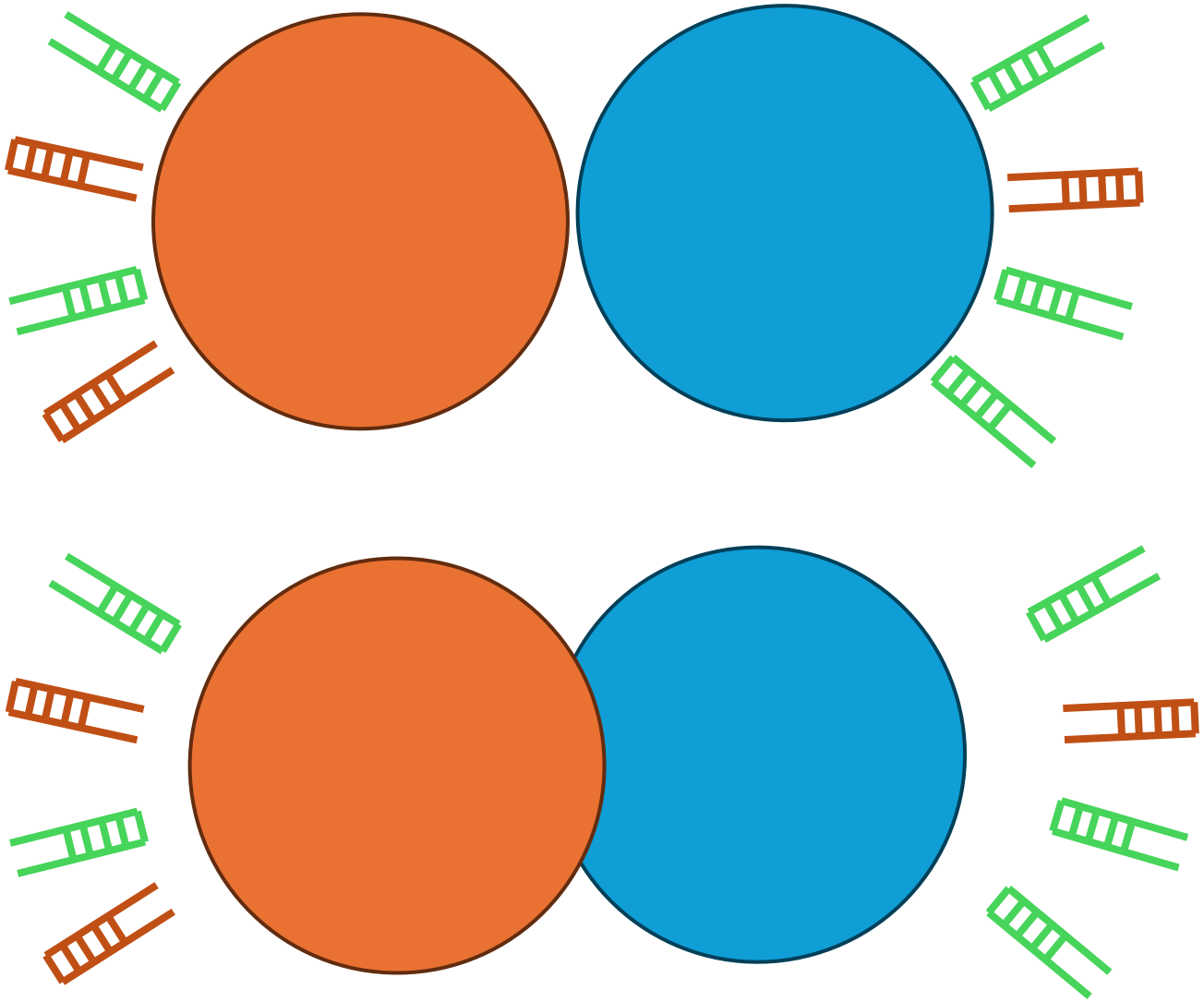
Parallel Composition



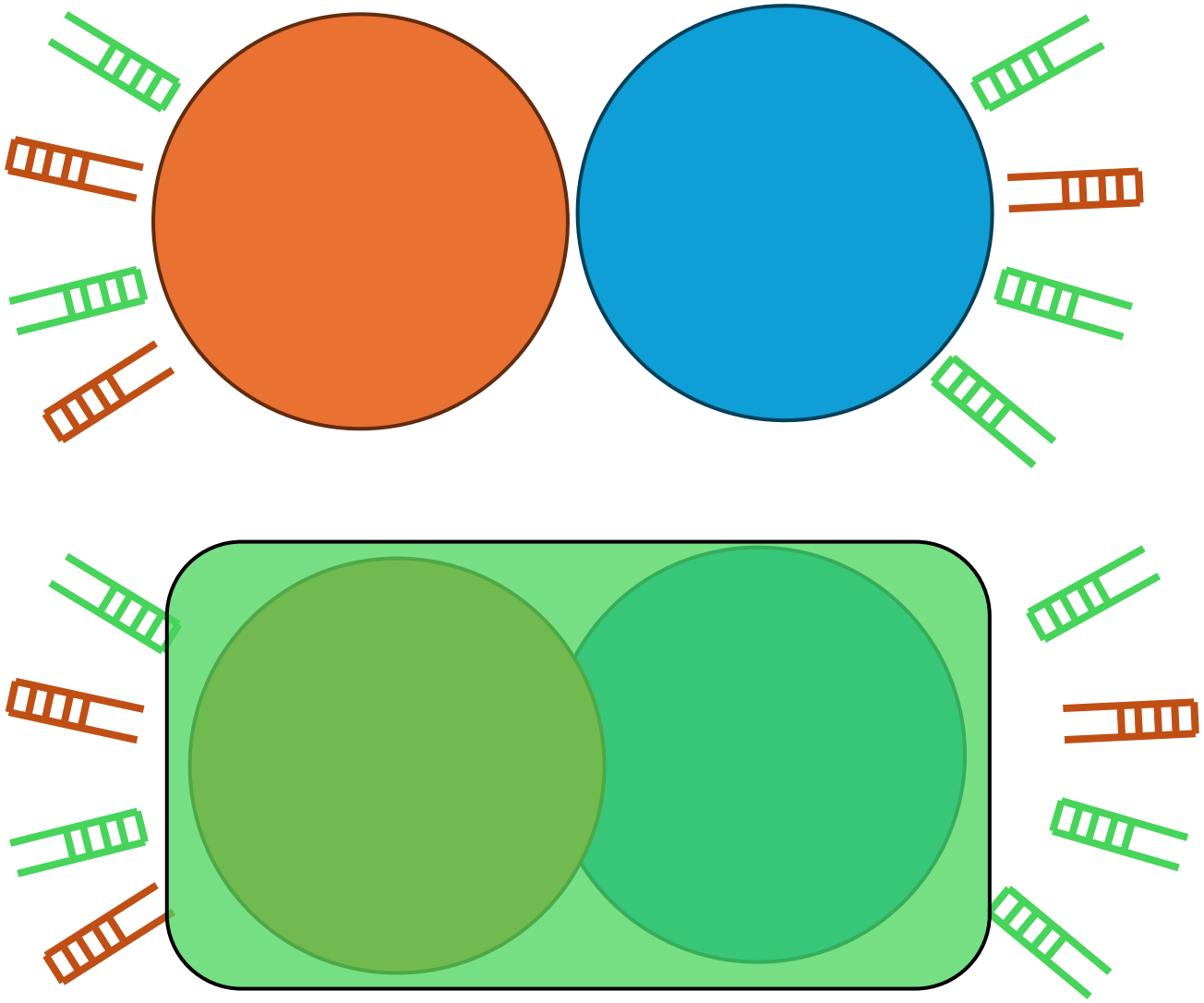
Parallel Composition



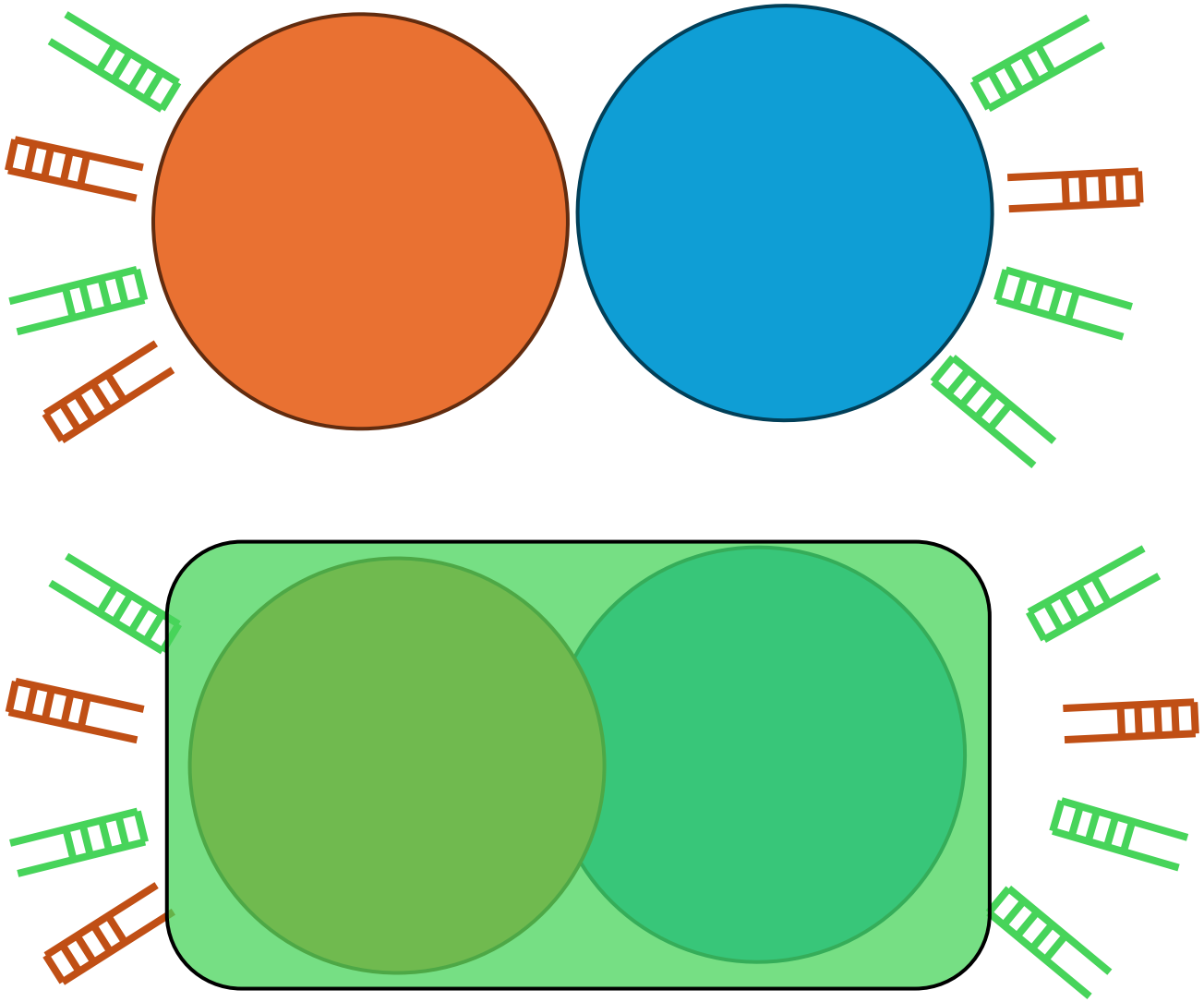
Parallel Composition



Parallel Composition



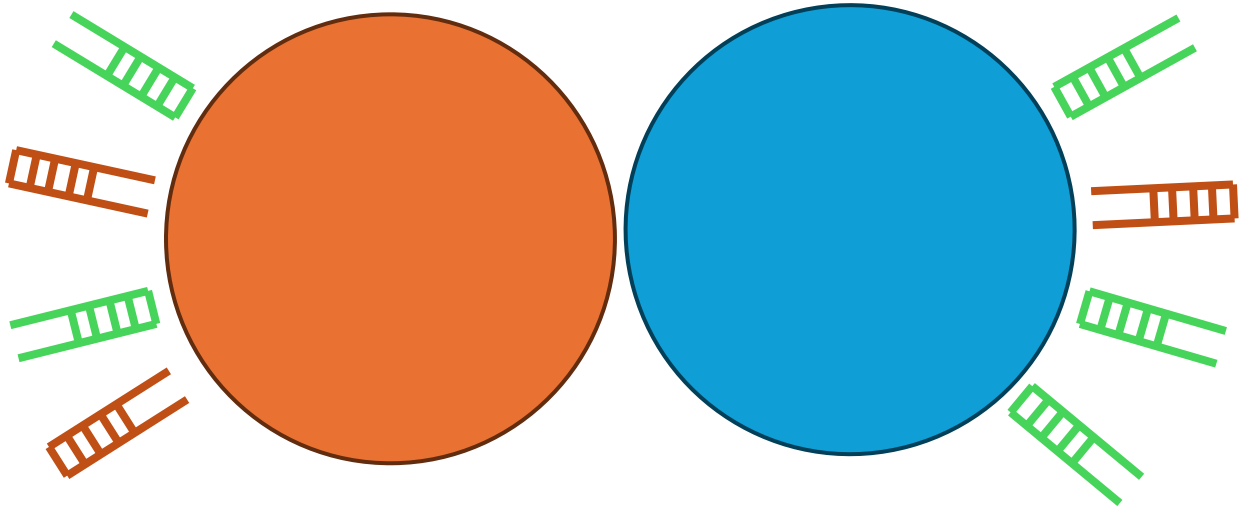
Parallel Composition



S1 , S2 ∈

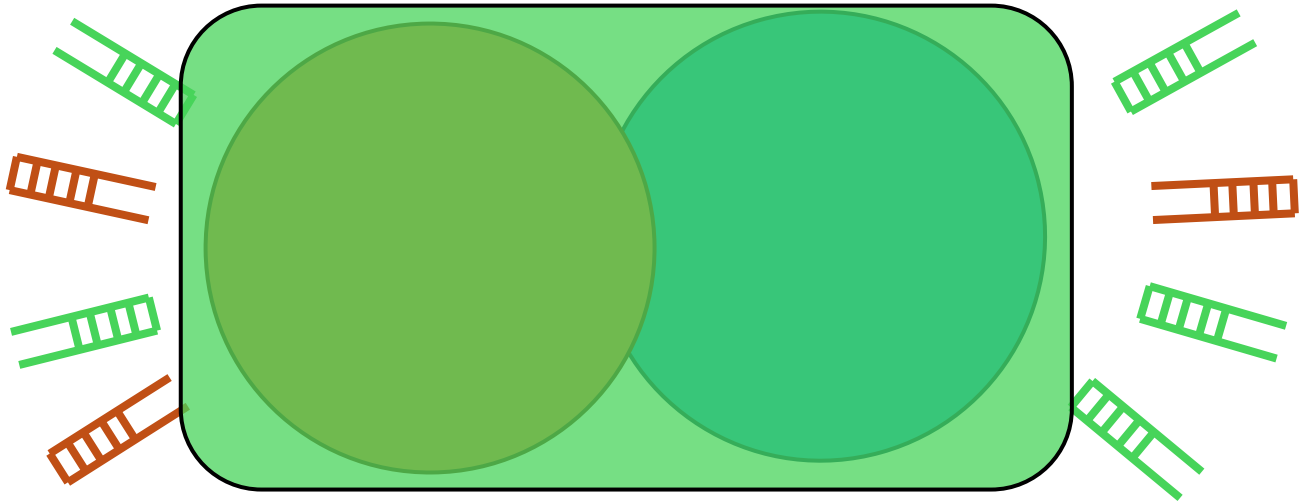
$\{f : InputHistory \rightarrow OutputTrace \mid f\ monotone\}$

Parallel Composition



S1 , S2 ∈

$\{f : InputHistory \rightarrow OutputTrace \mid f\ monotone\}$



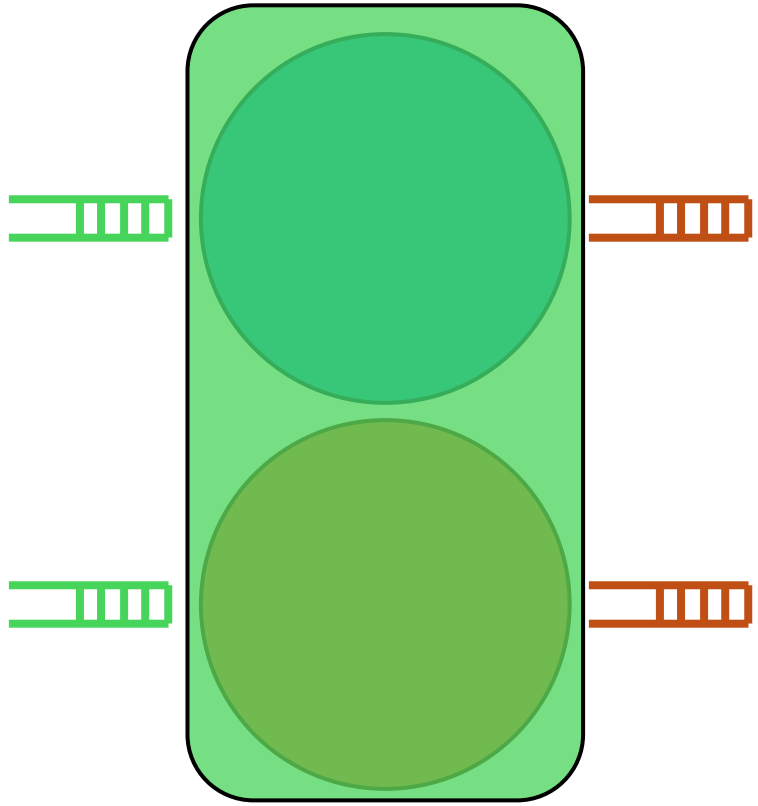
(S1 || S2) ∈

$\{f : InputHistory \rightarrow OutputTrace \mid f\ monotone\}$

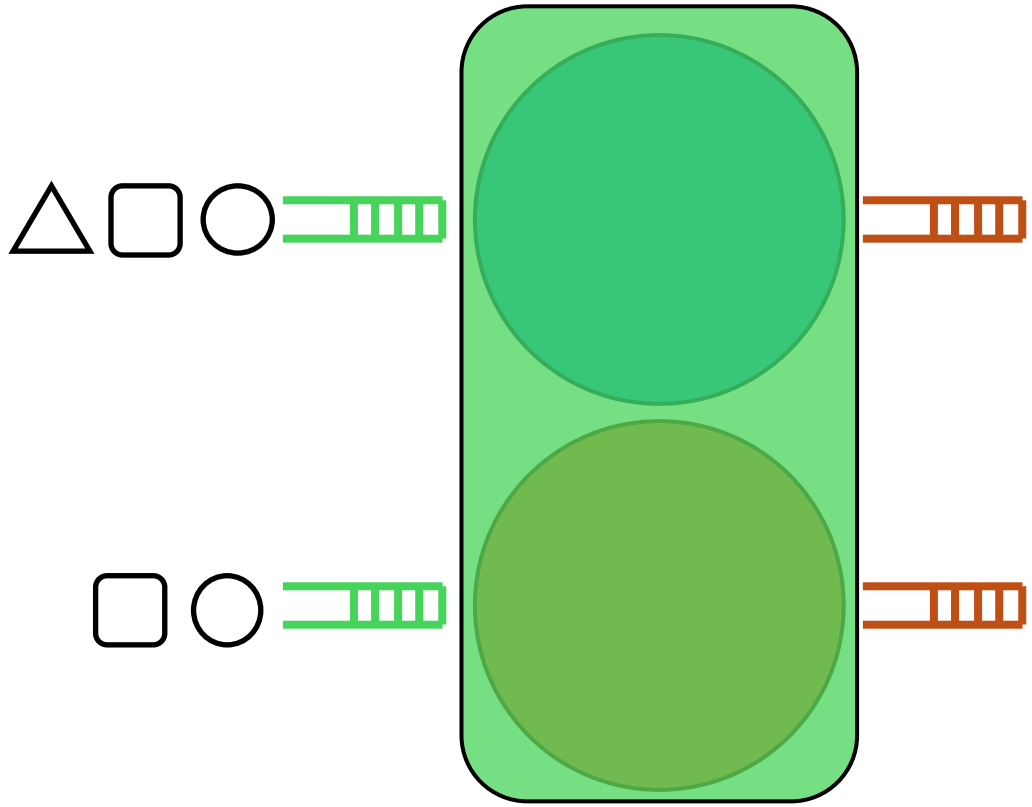
??

Parallel Composition

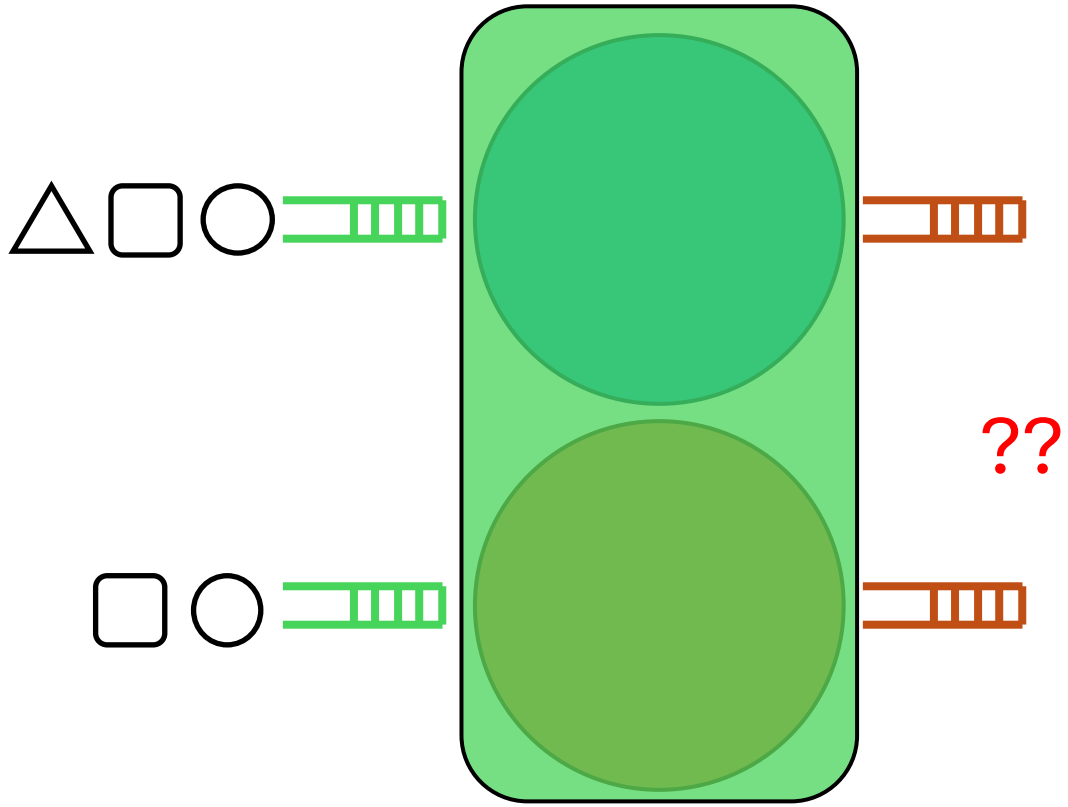
Parallel Composition



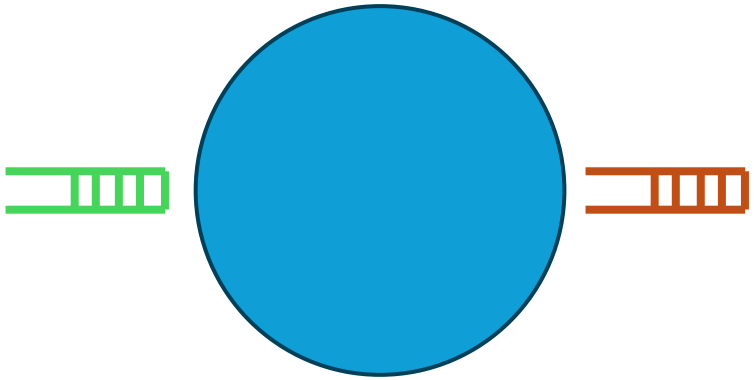
Parallel Composition

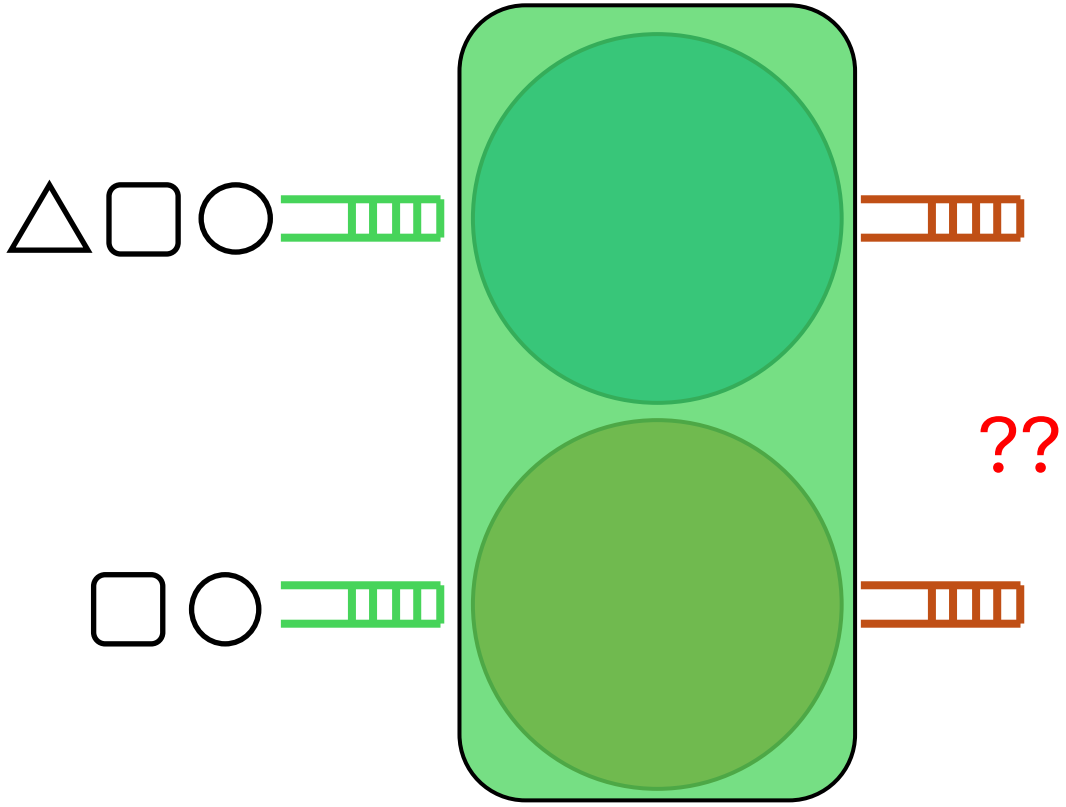


Parallel Composition

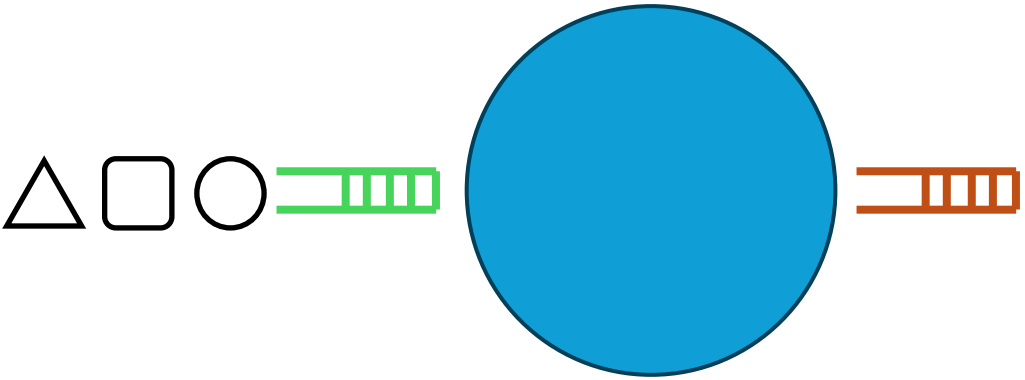


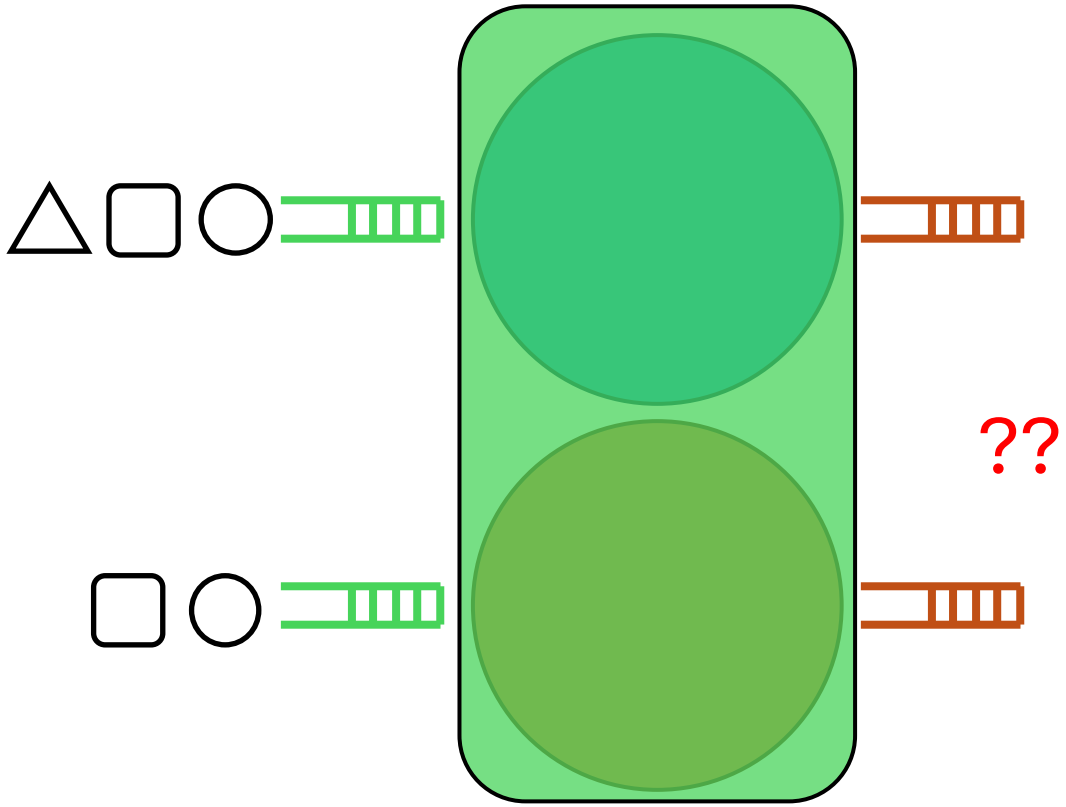
Parallel Composition



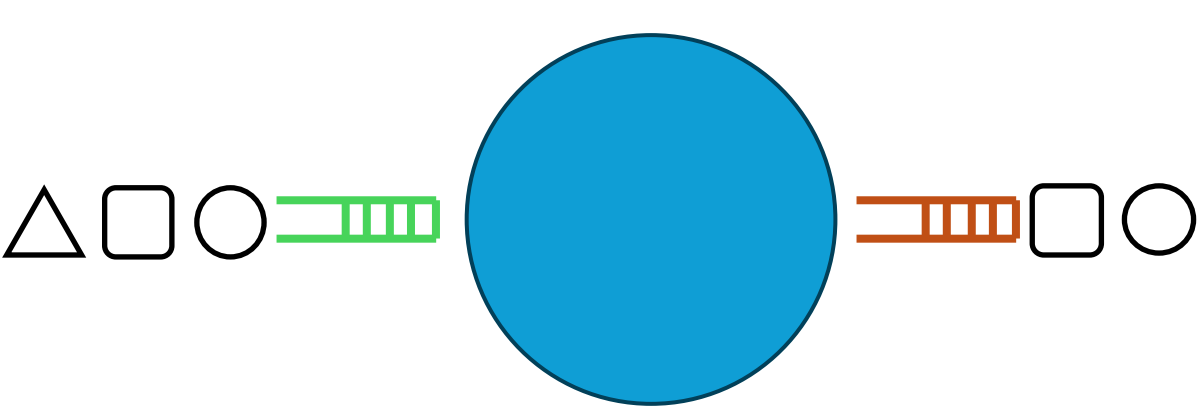


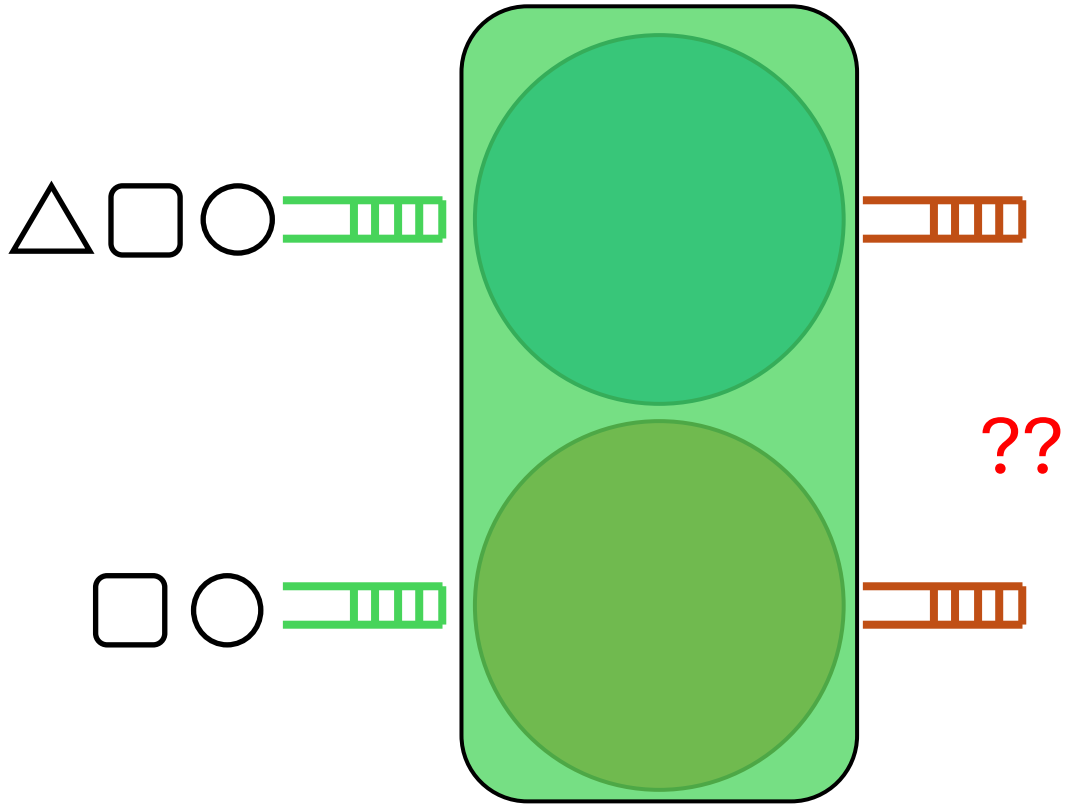
Parallel Composition



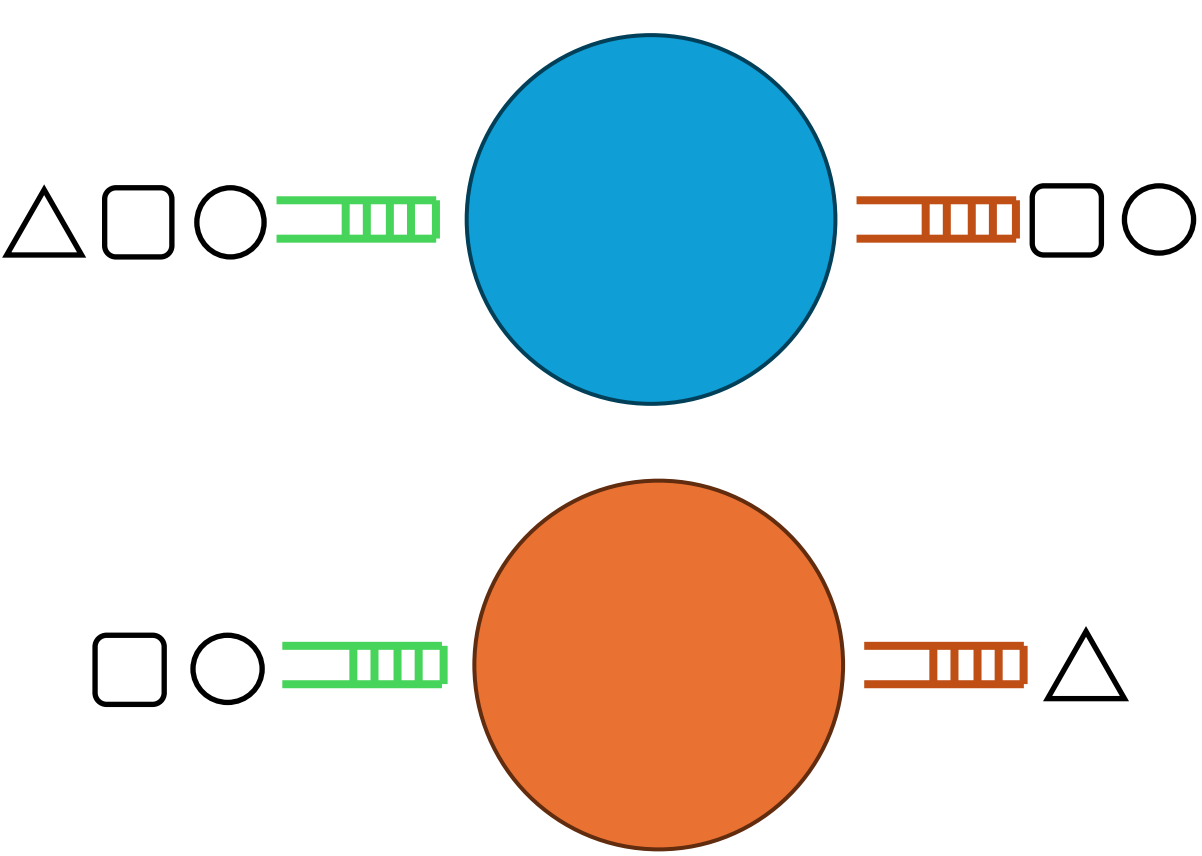


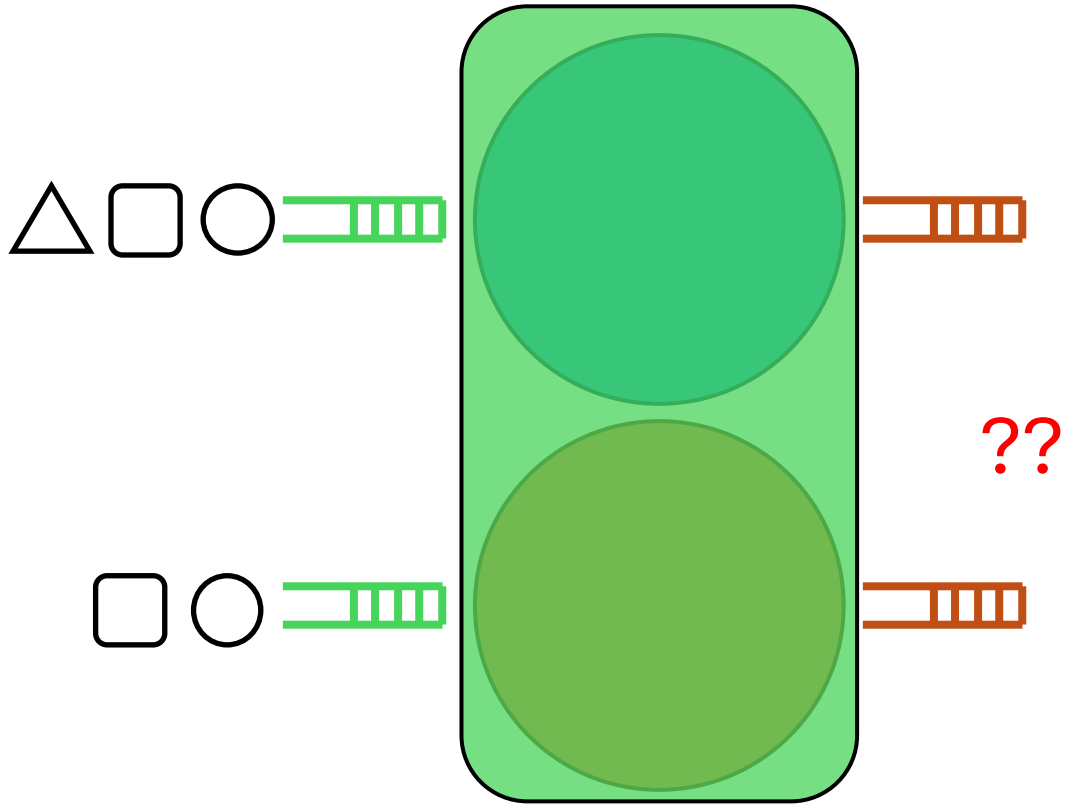
Parallel Composition



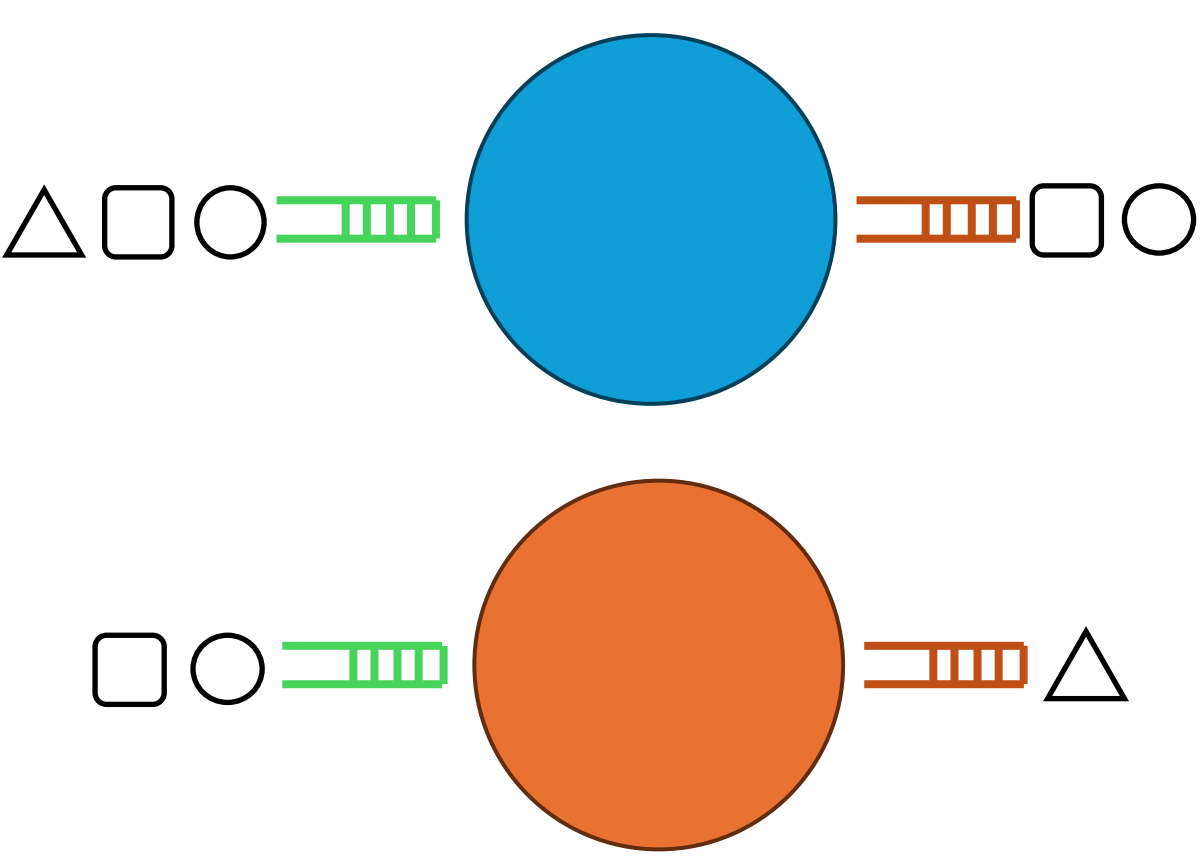


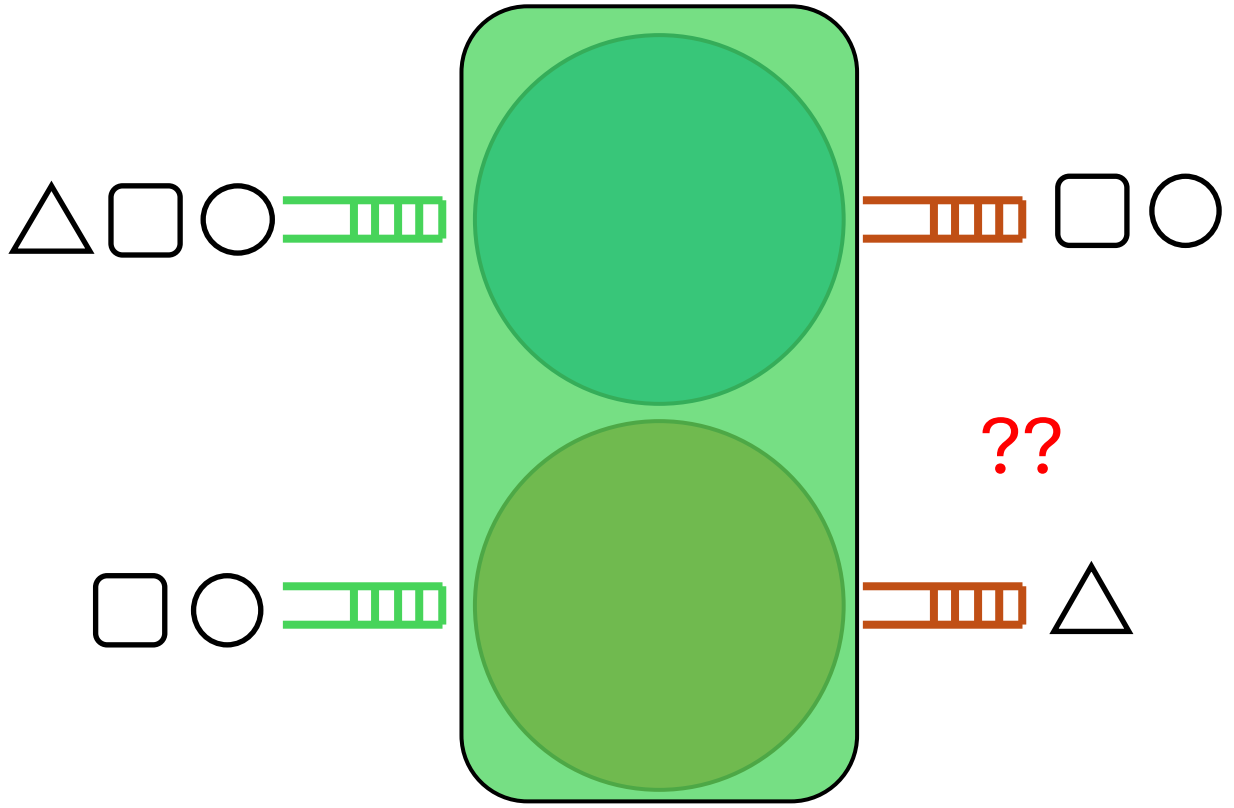
Parallel Composition



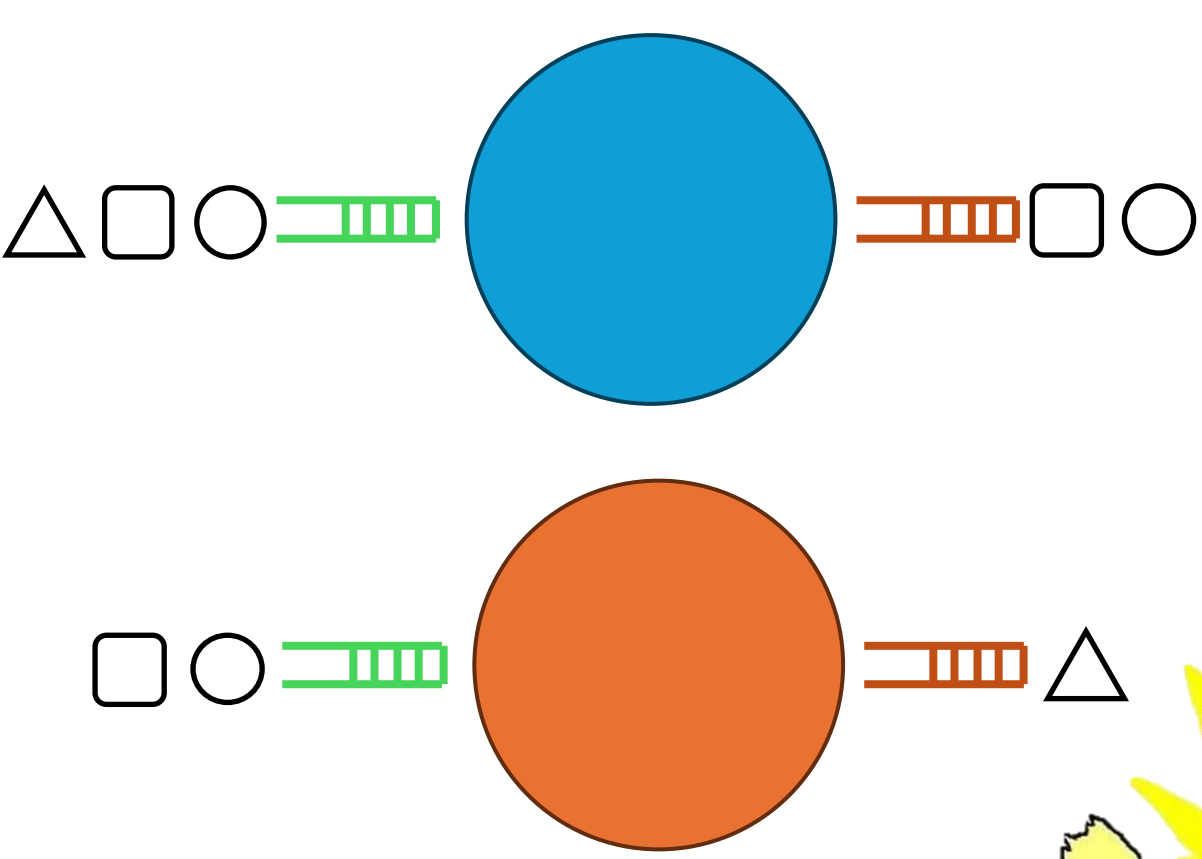


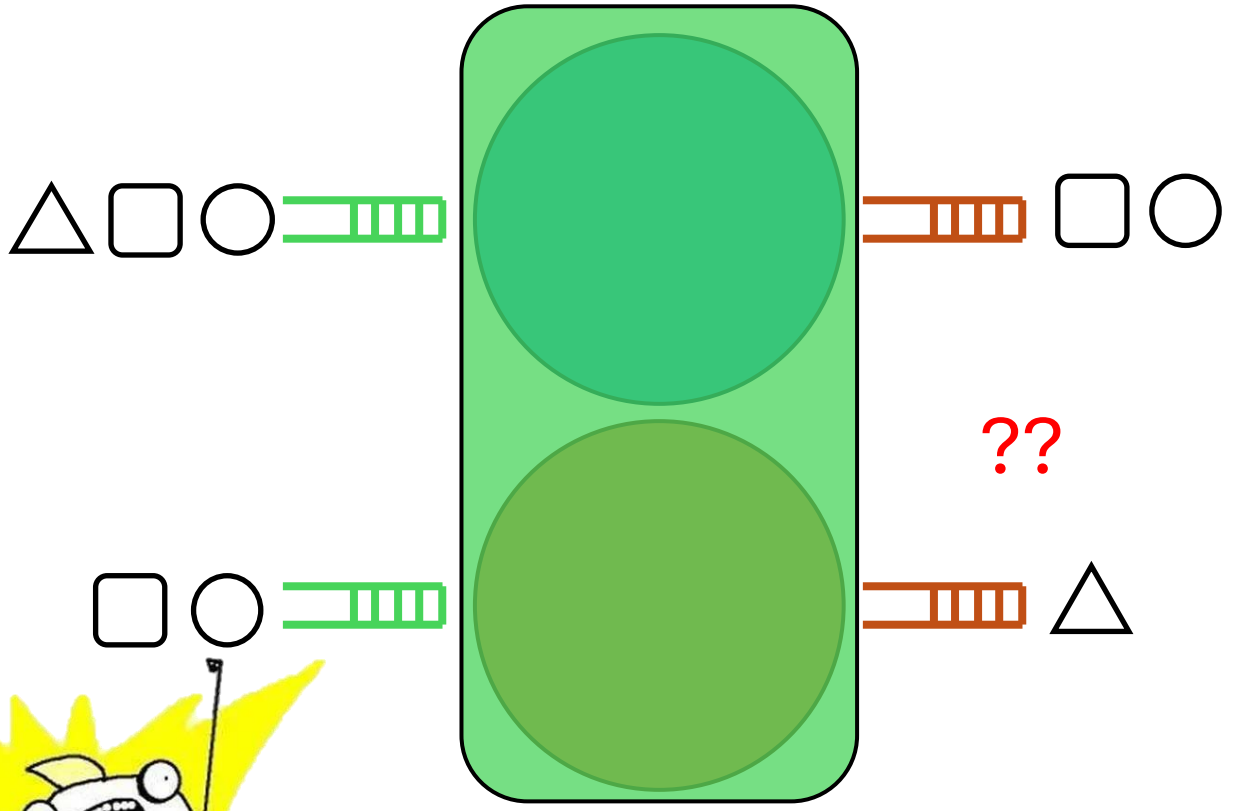
Parallel Composition





Parallel Composition

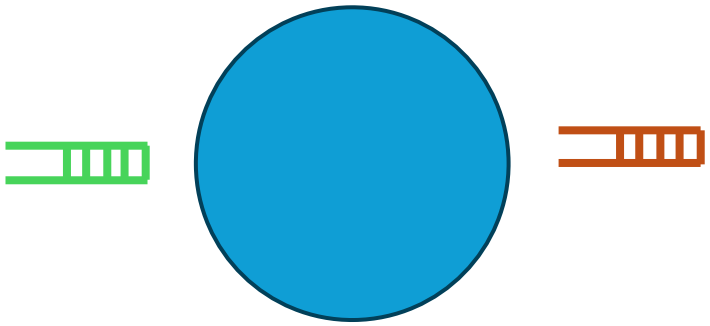




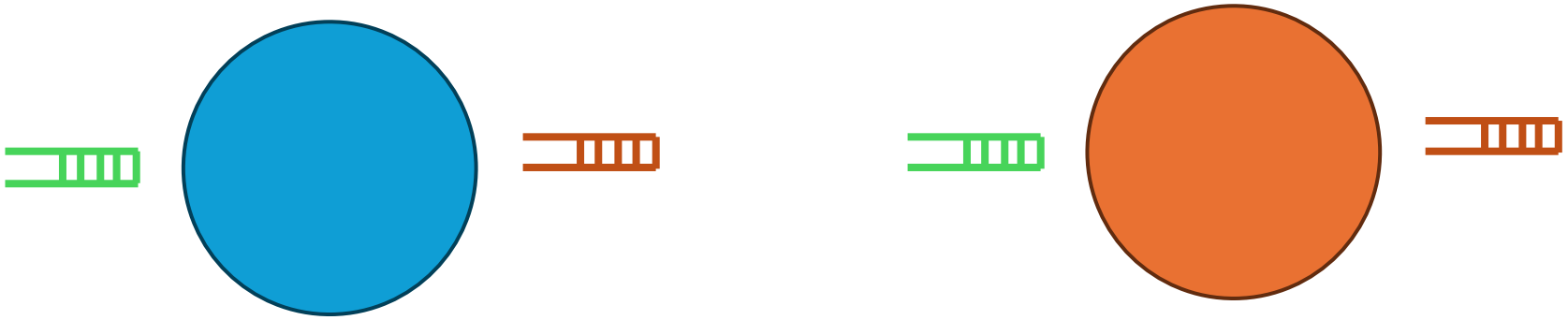


Sequential Composition

Sequential Composition



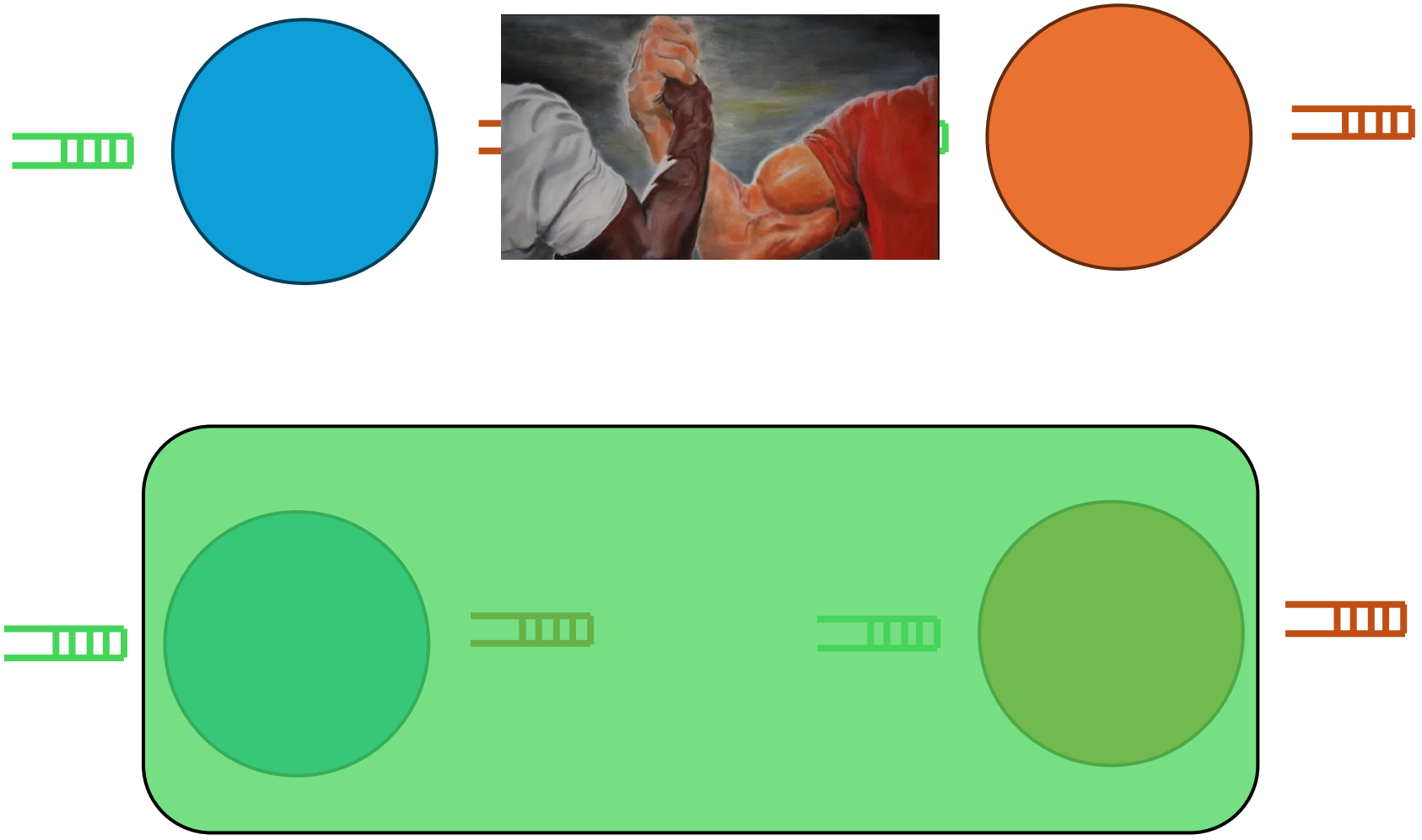
Sequential Composition



Sequential Composition



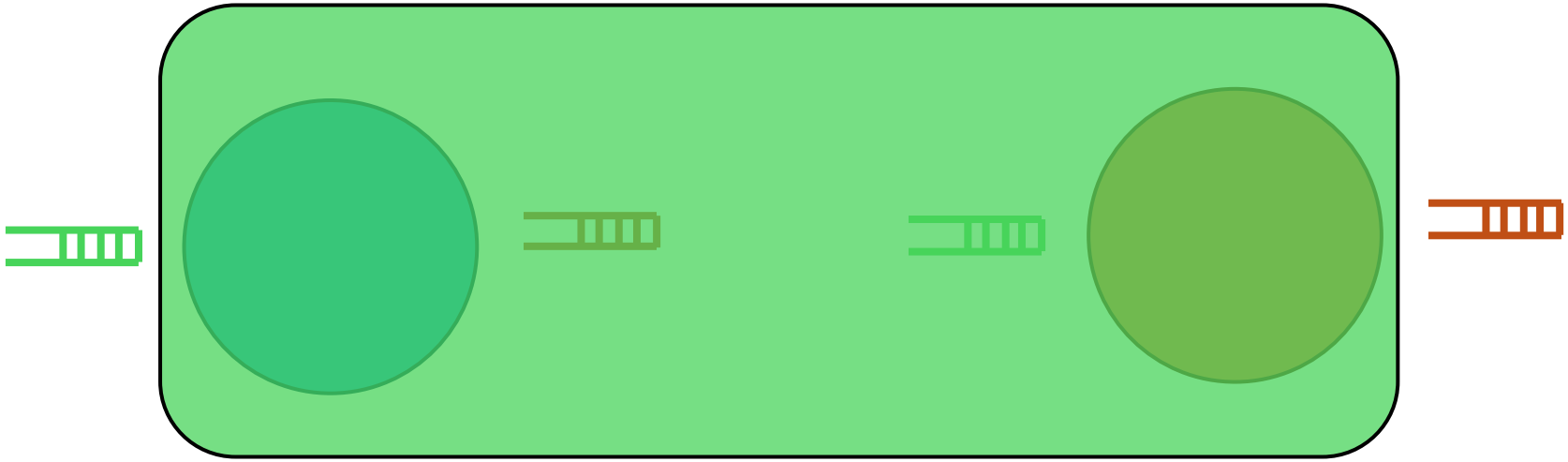
Sequential Composition



Sequential Composition

S1 , S2 ∈
$\{f : InputHistory \rightarrow OutputTrace \mid f\ monotone\}$





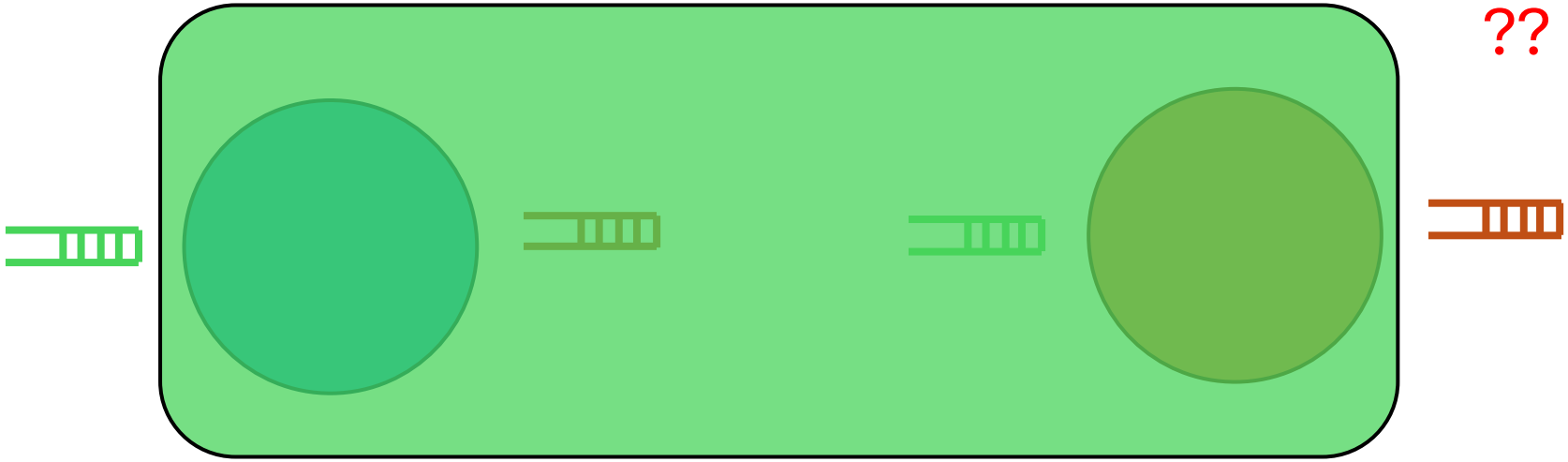
Sequential Composition

S1 , S2 ∈
$\{f : InputHistory \rightarrow OutputTrace \mid f\ monotone\}$



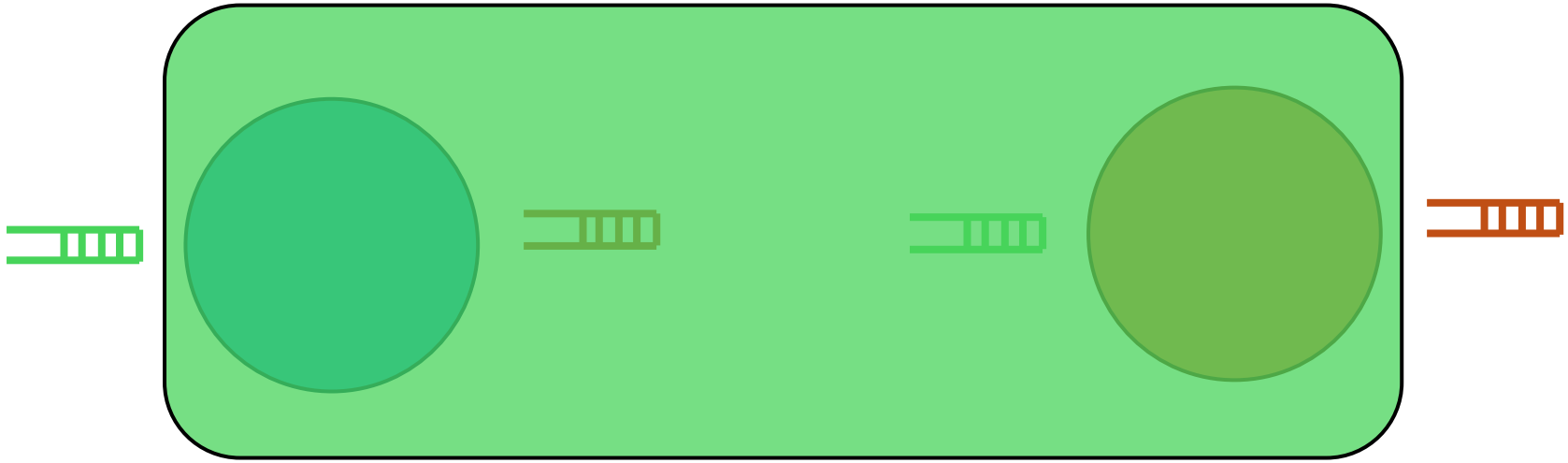
S1 - S2 ∈
$\{f : InputHistory \rightarrow OutputTrace \mid f\ monotone\}$

??

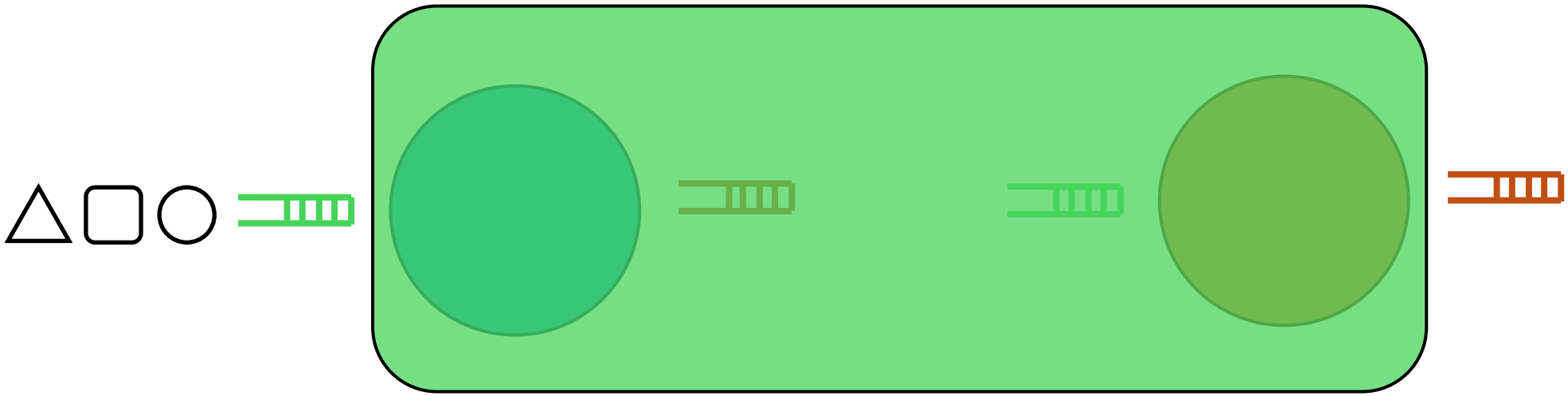


Sequential Composition

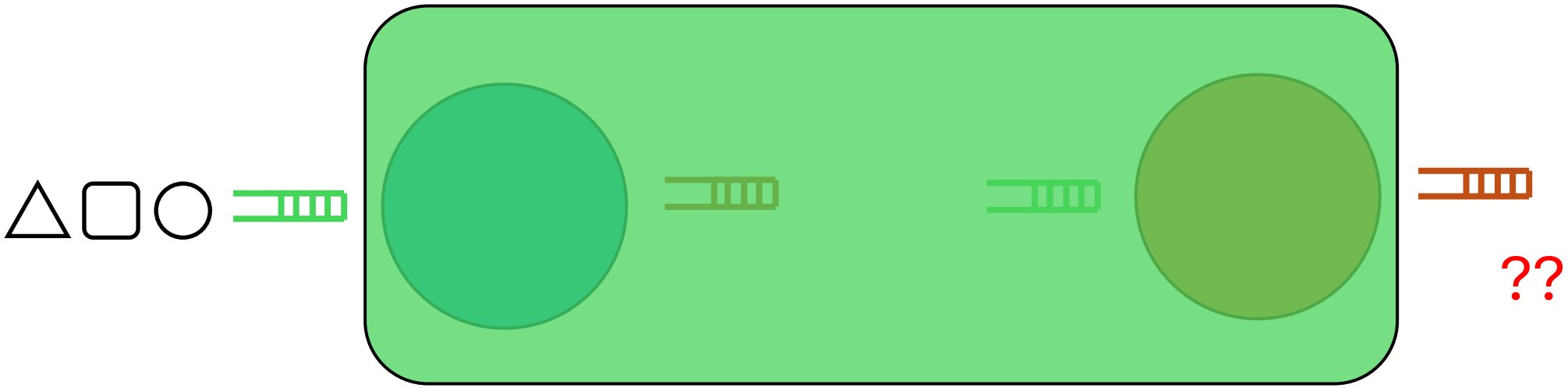
Sequential Composition



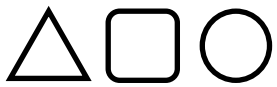
Sequential Composition

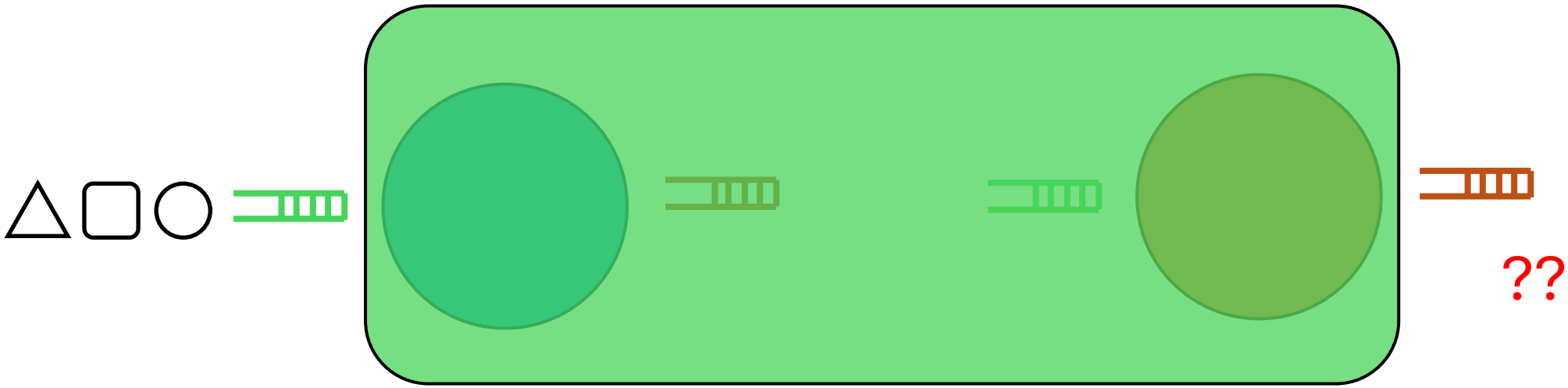


Sequential Composition

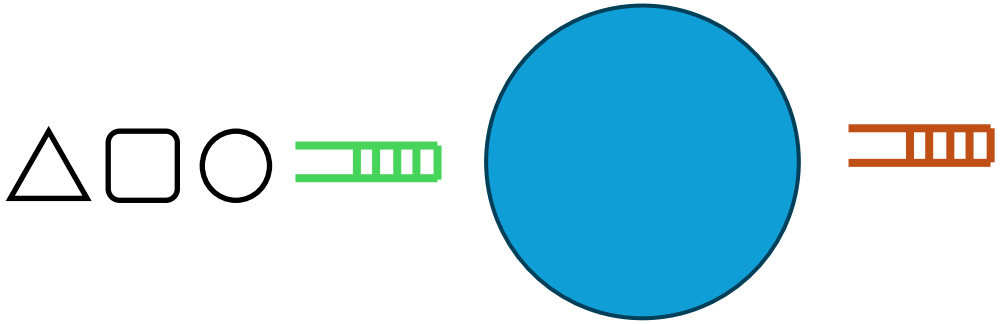


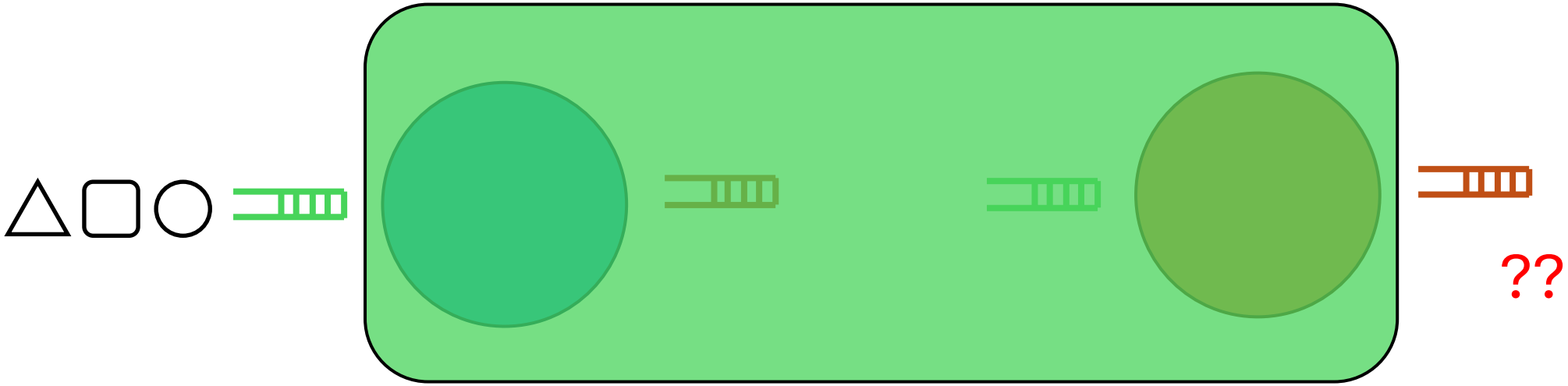
Sequential Composition



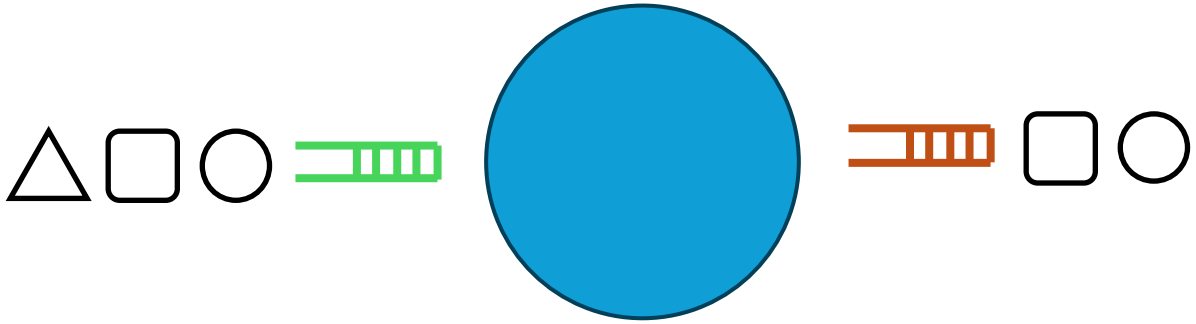


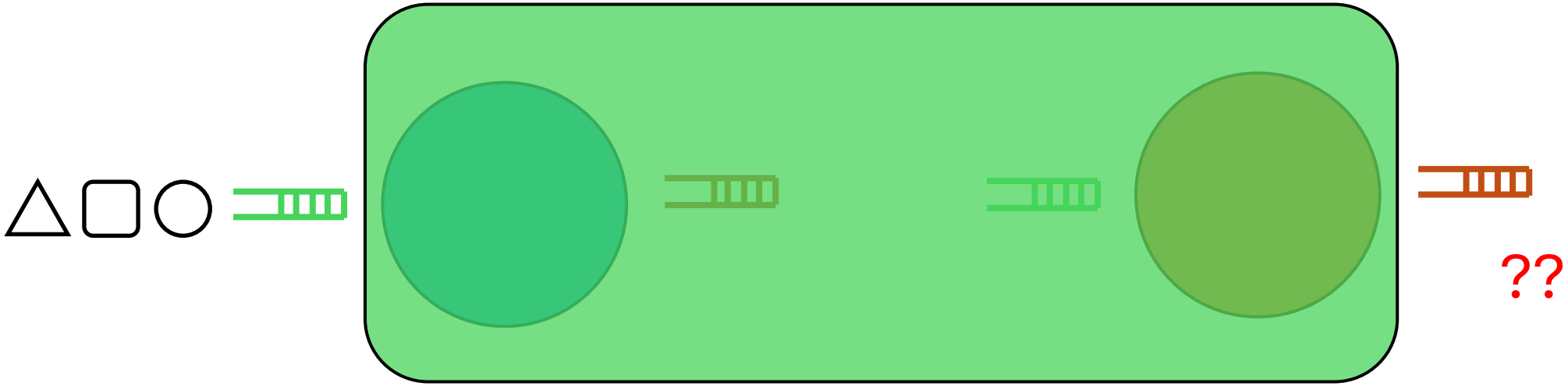
Sequential Composition



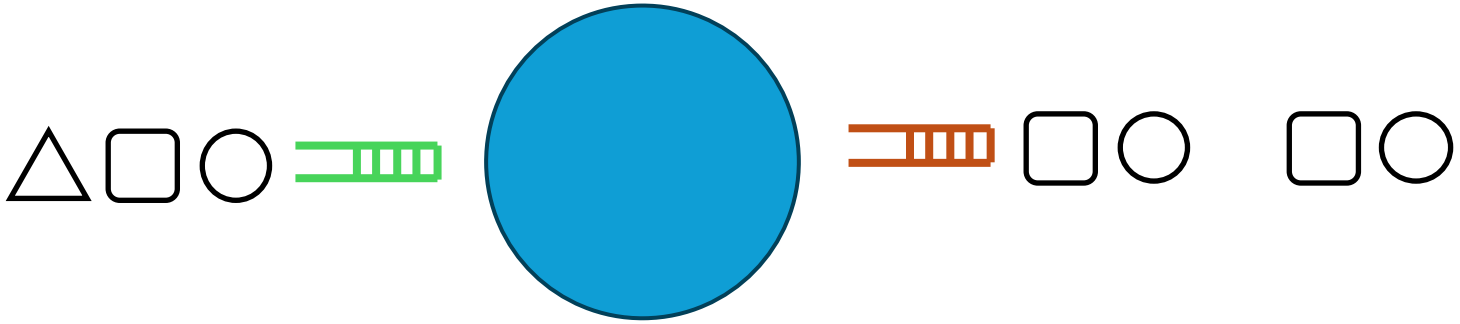


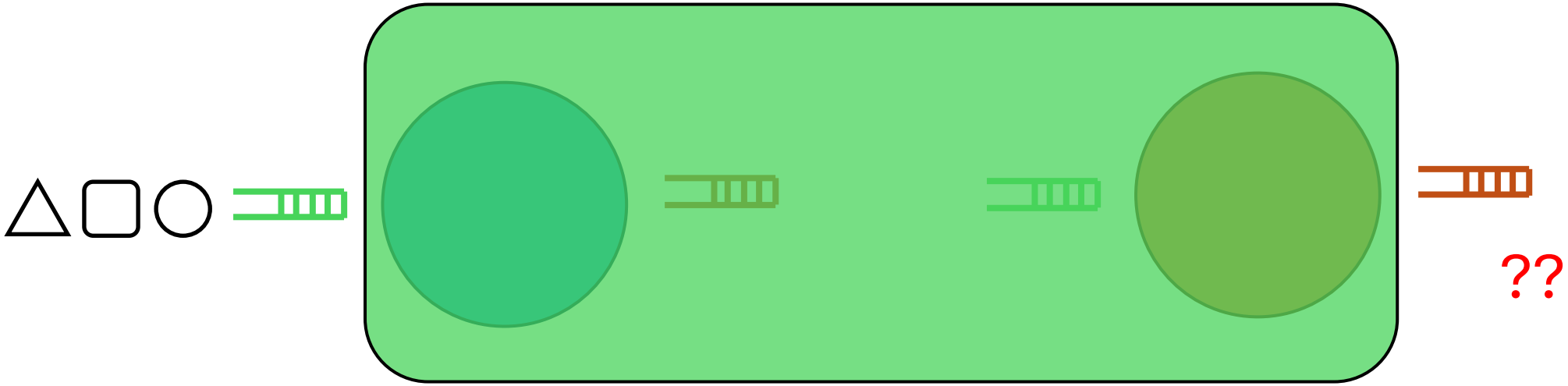
Sequential Composition



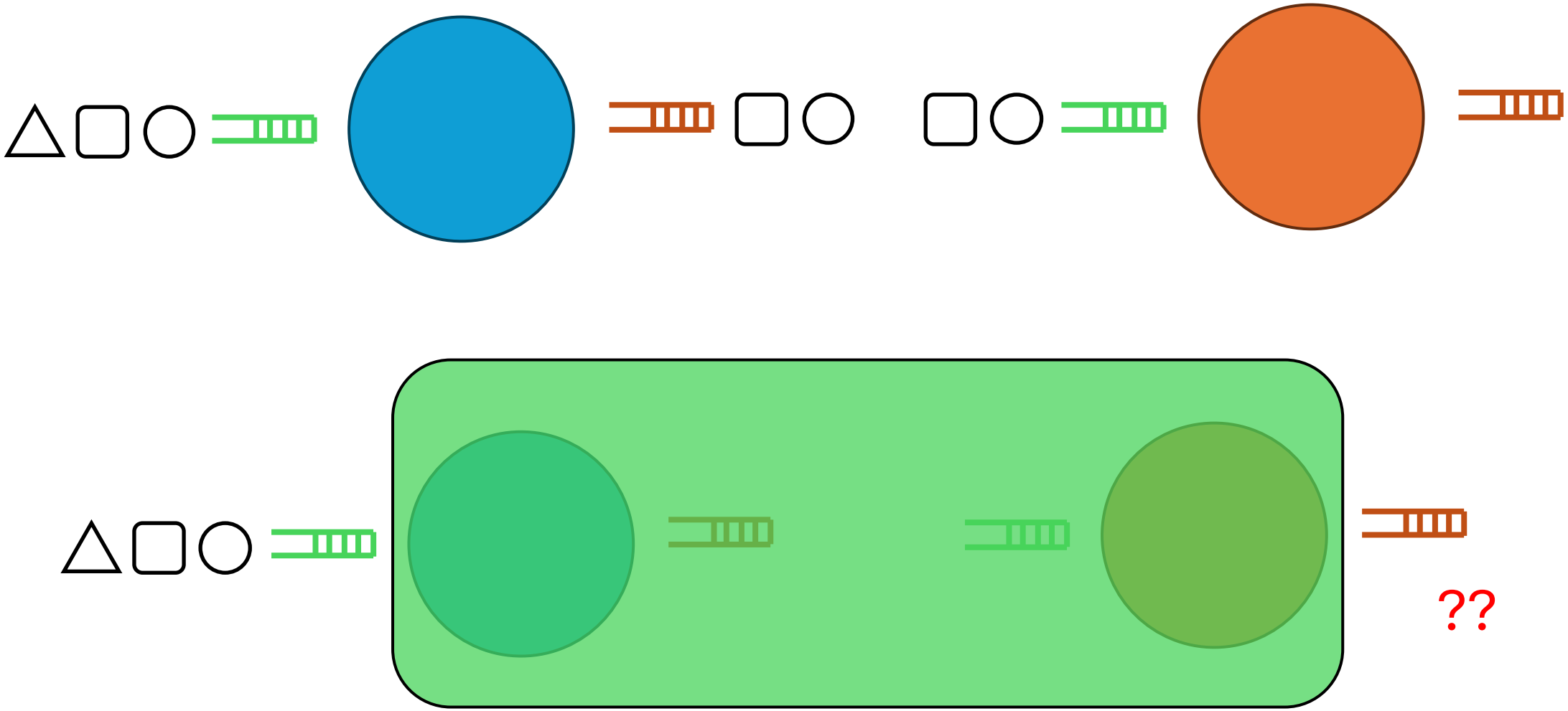


Sequential Composition

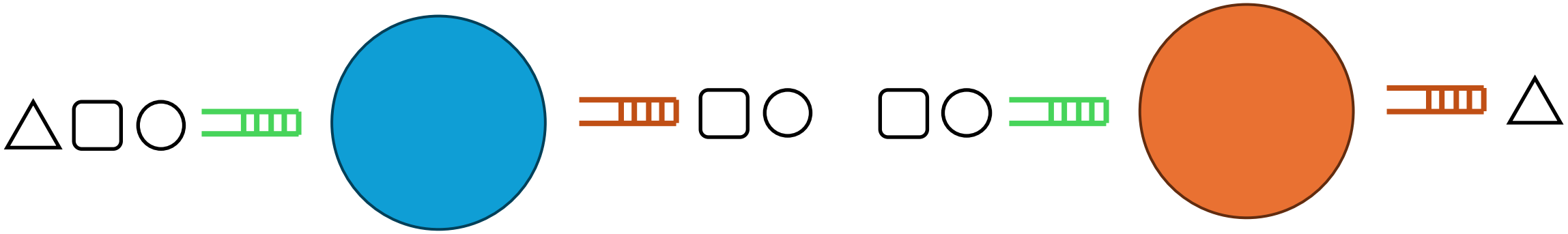


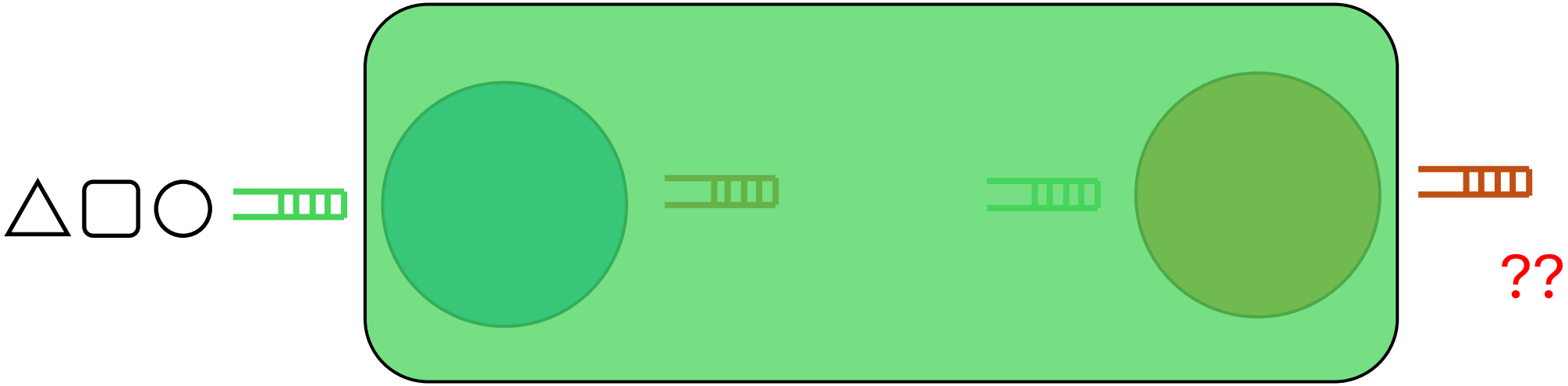


Sequential Composition

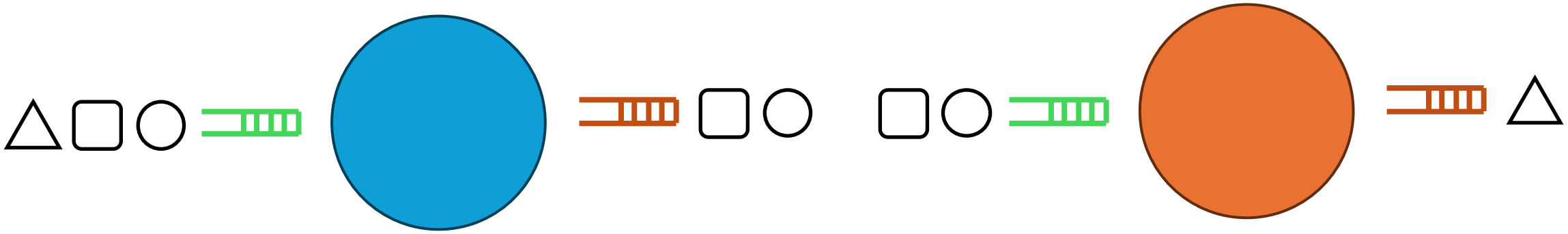


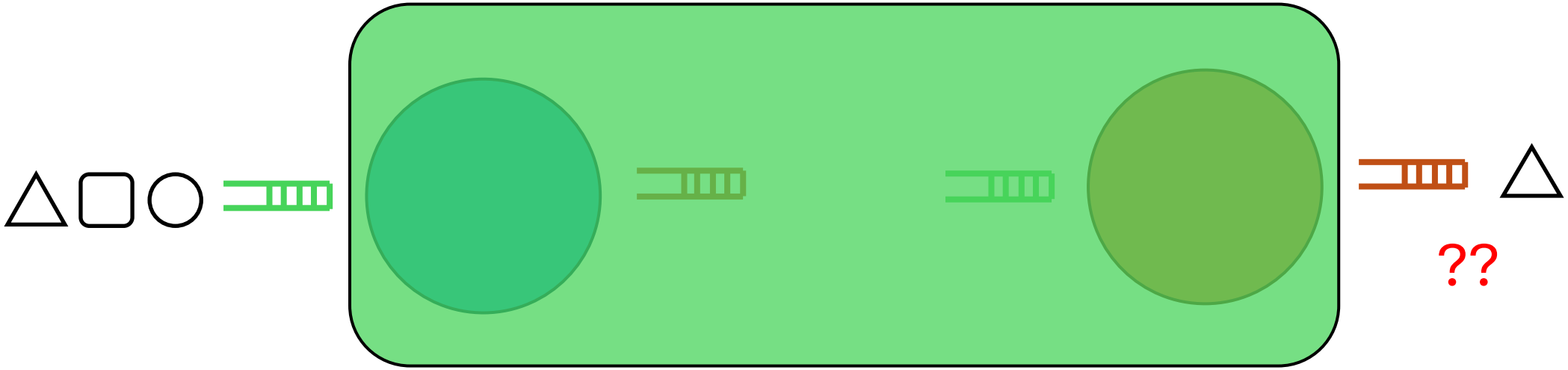
Sequential Composition



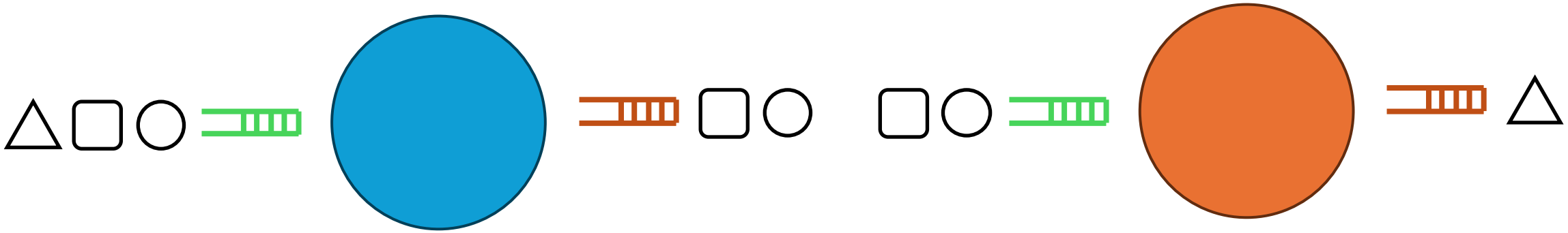


Sequential Composition





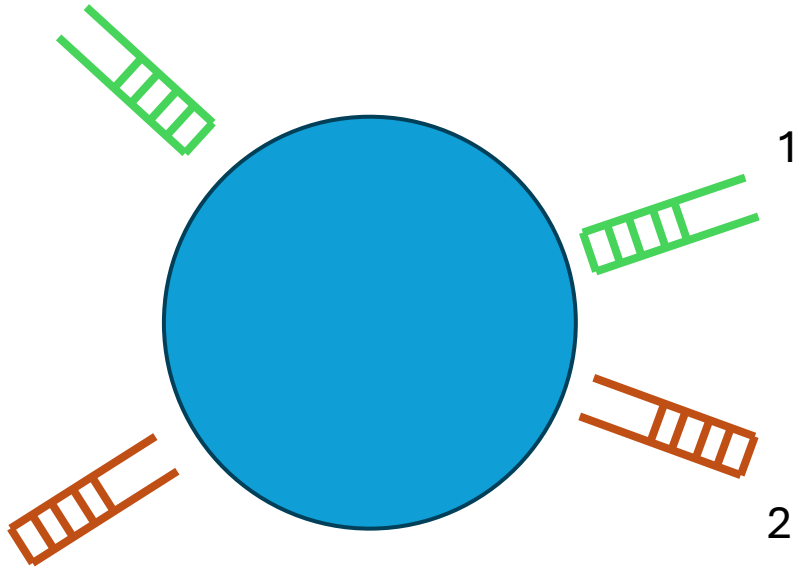
Sequential Composition



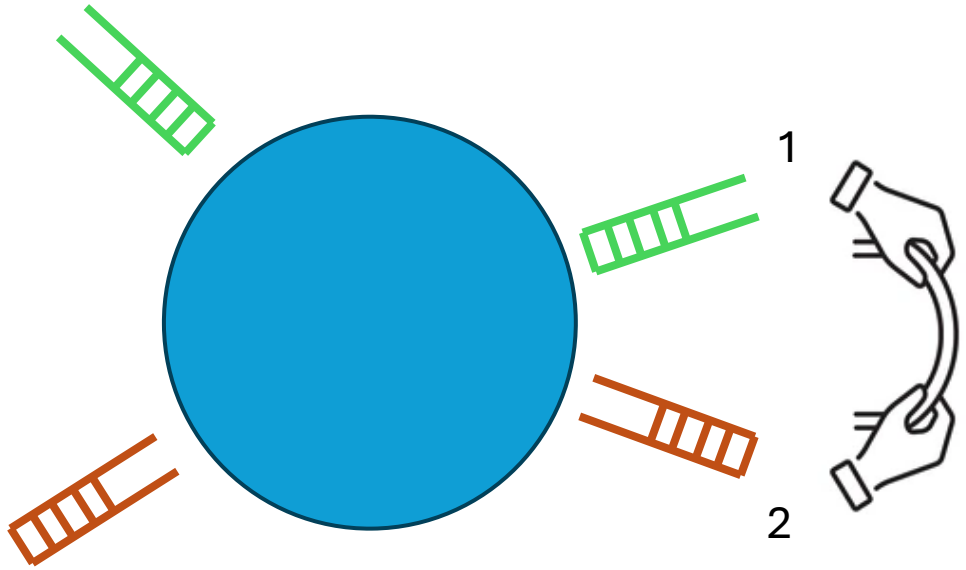


Self Composition

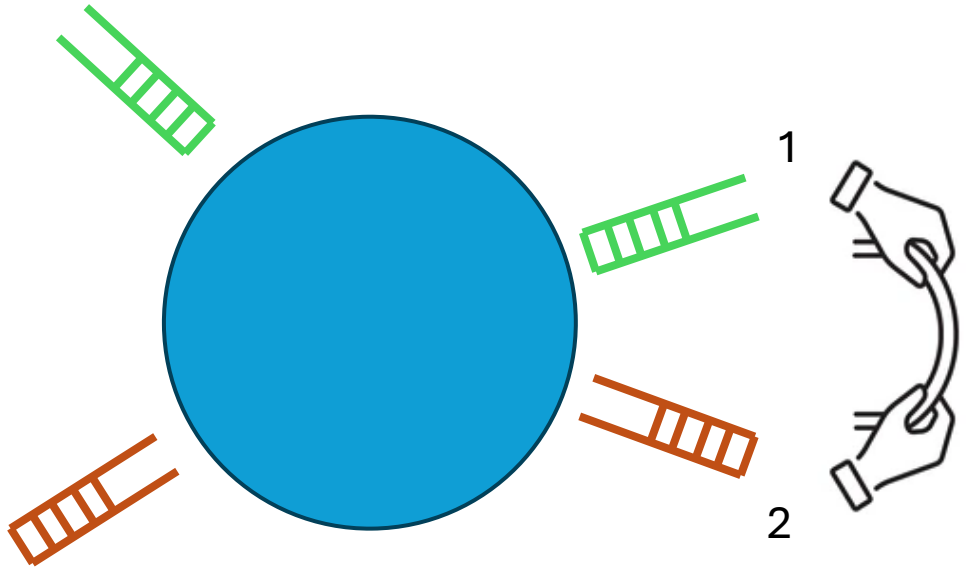
Self Composition

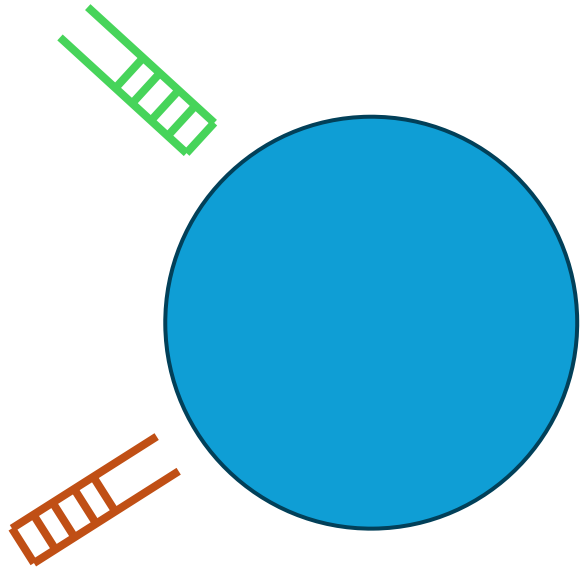


Self Composition

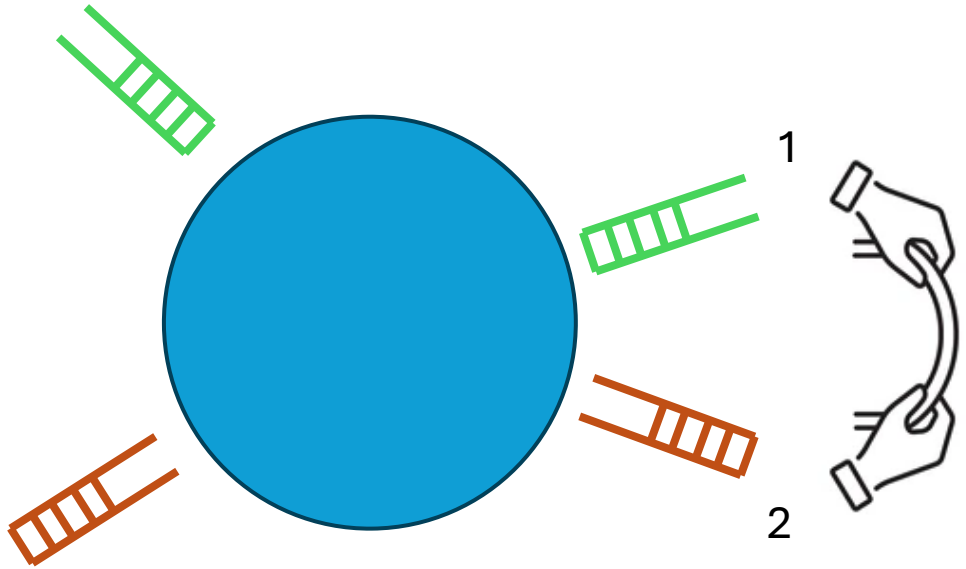


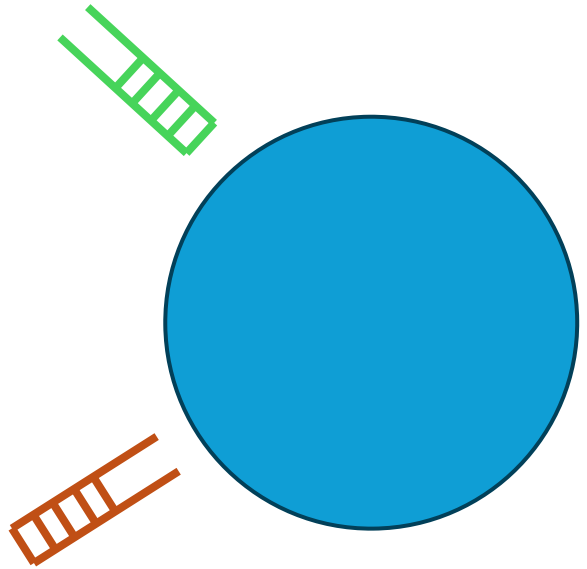
Self Composition





Self Composition

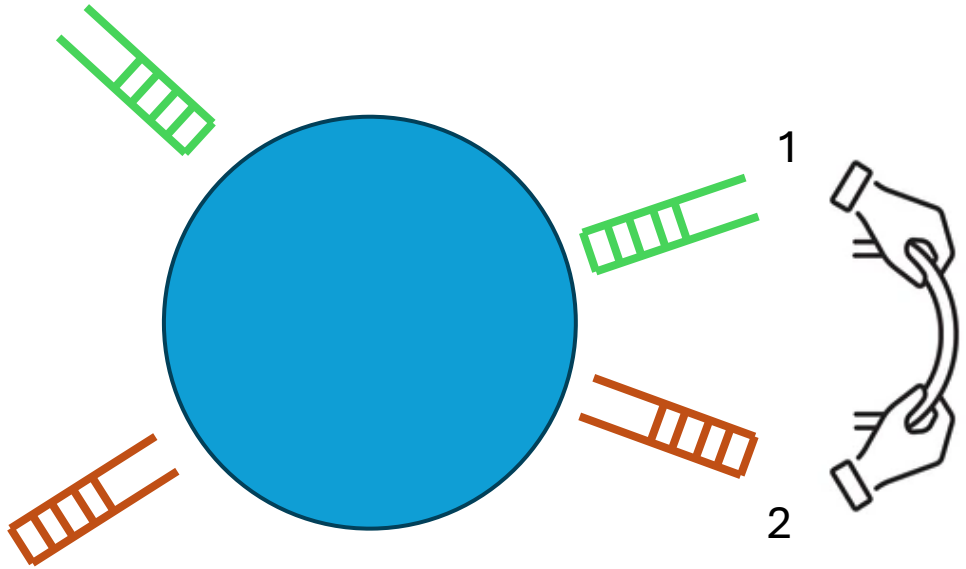


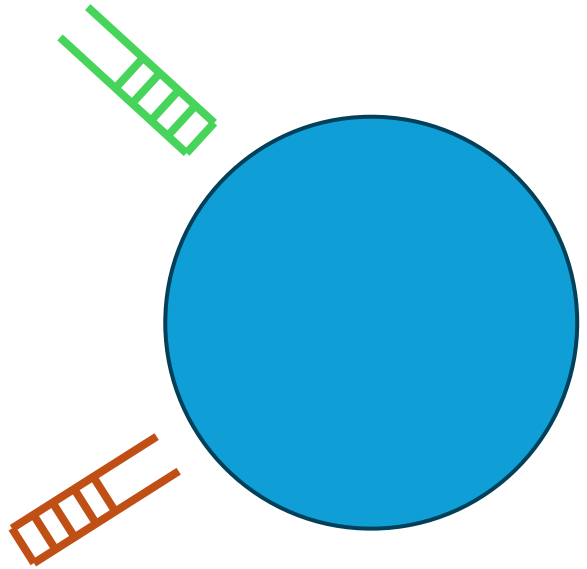


S ∈

$\{f : InputHistory \rightarrow OutputTrace \mid f\ monotone\}$

Self Composition



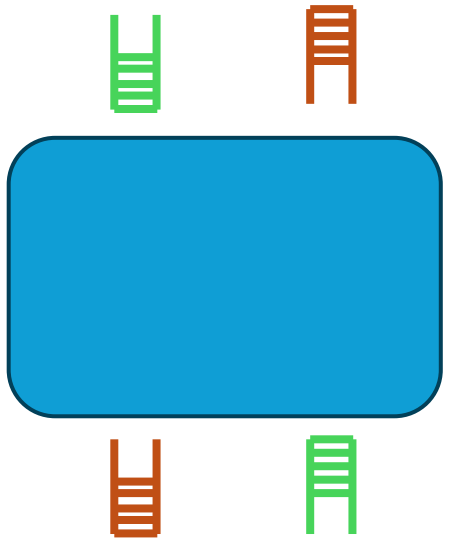


S ∈
$\{f : InputHistory \rightarrow OutputTrace \mid f\ monotone\}$

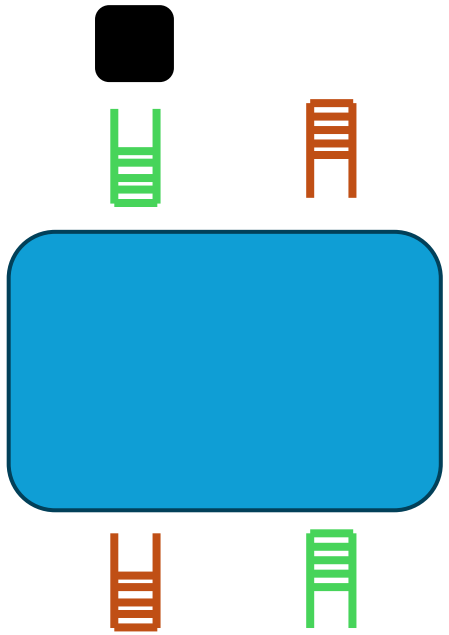
γ_{1-2}(S) ∈
$\{f : InputHistory \rightarrow OutputTrace \mid f\ monotone\}$
??

Example

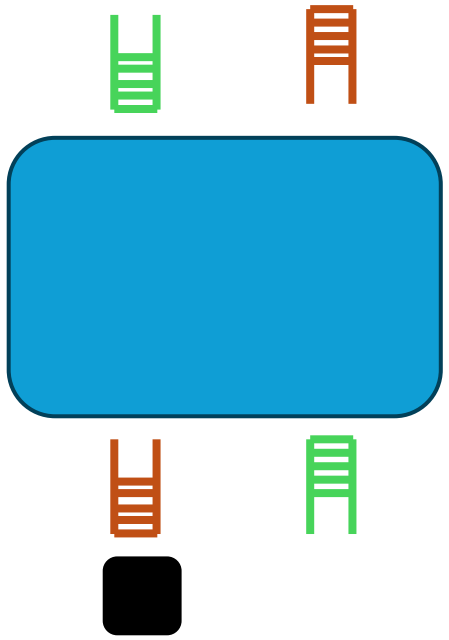
Example



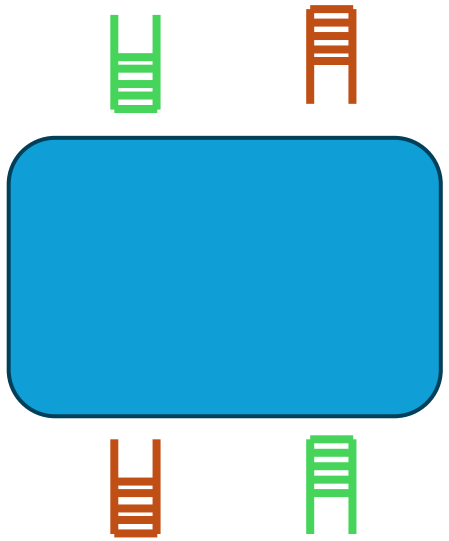
Example



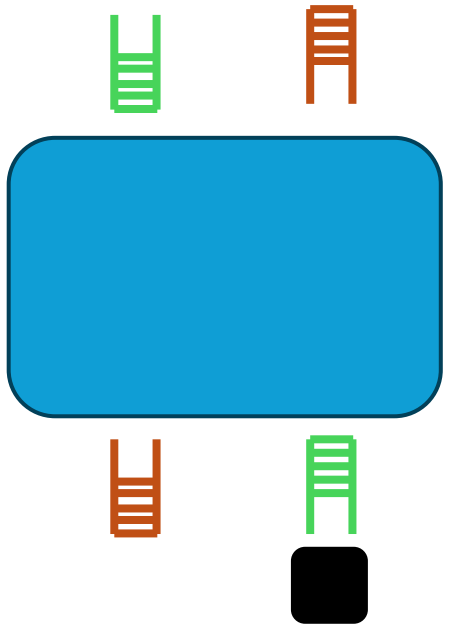
Example



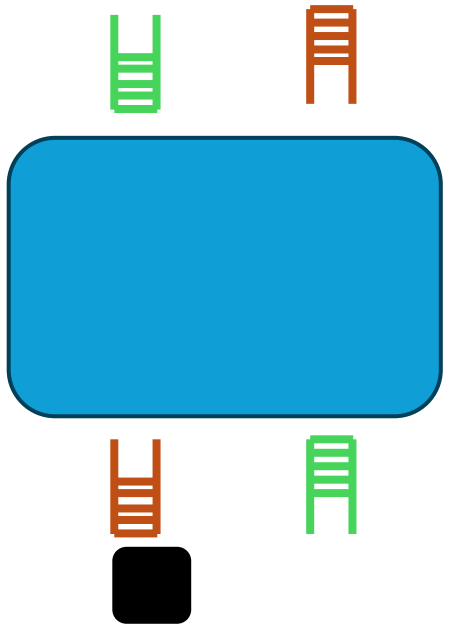
Example



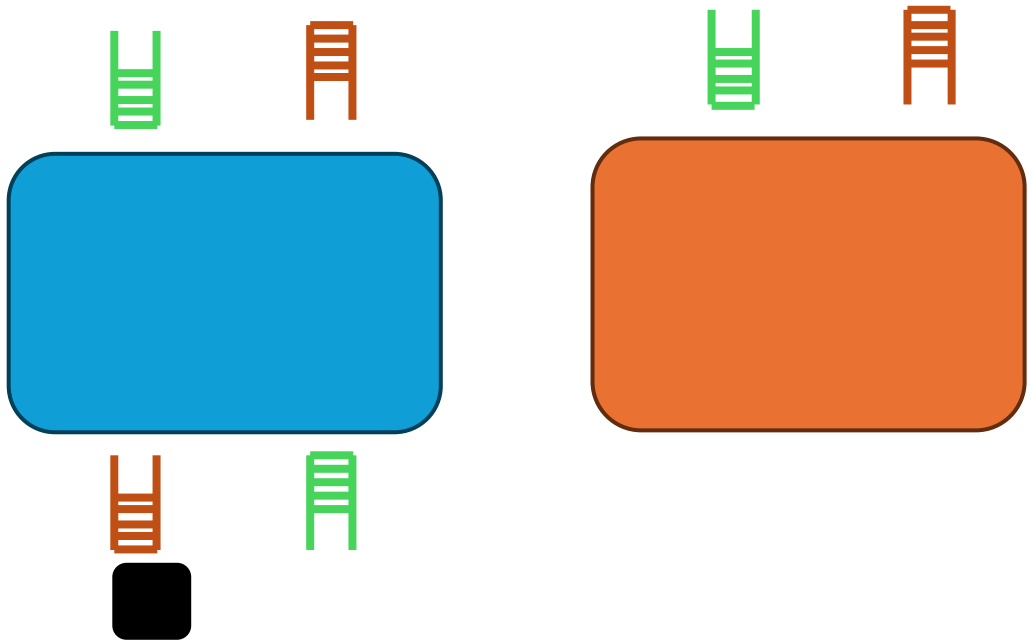
Example



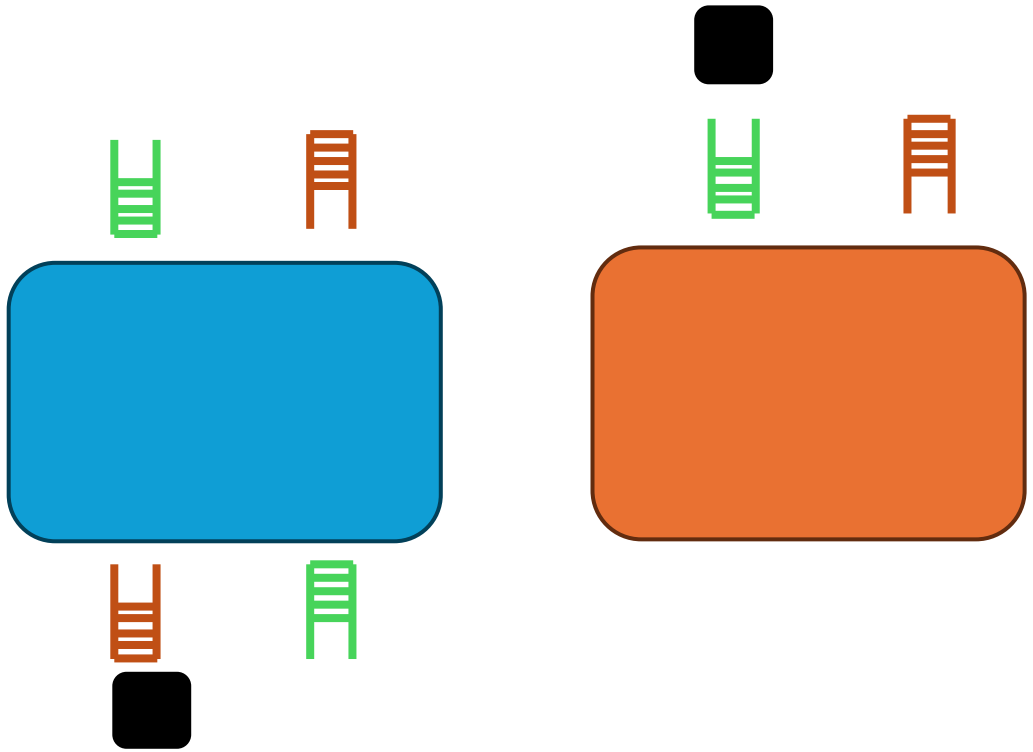
Example



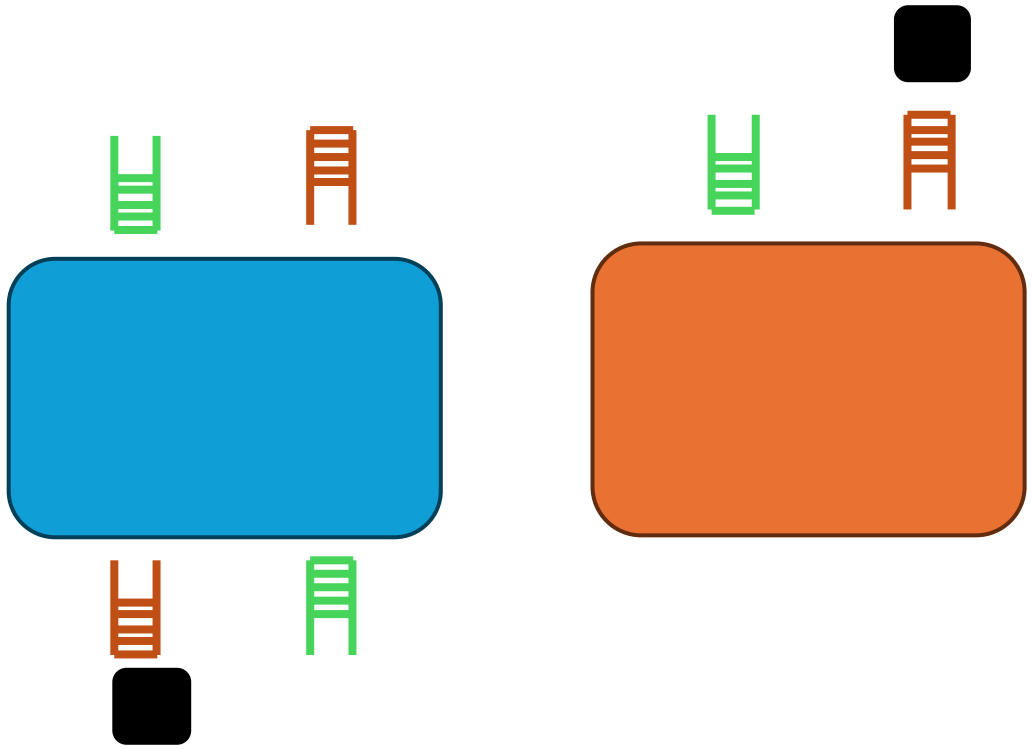
Example



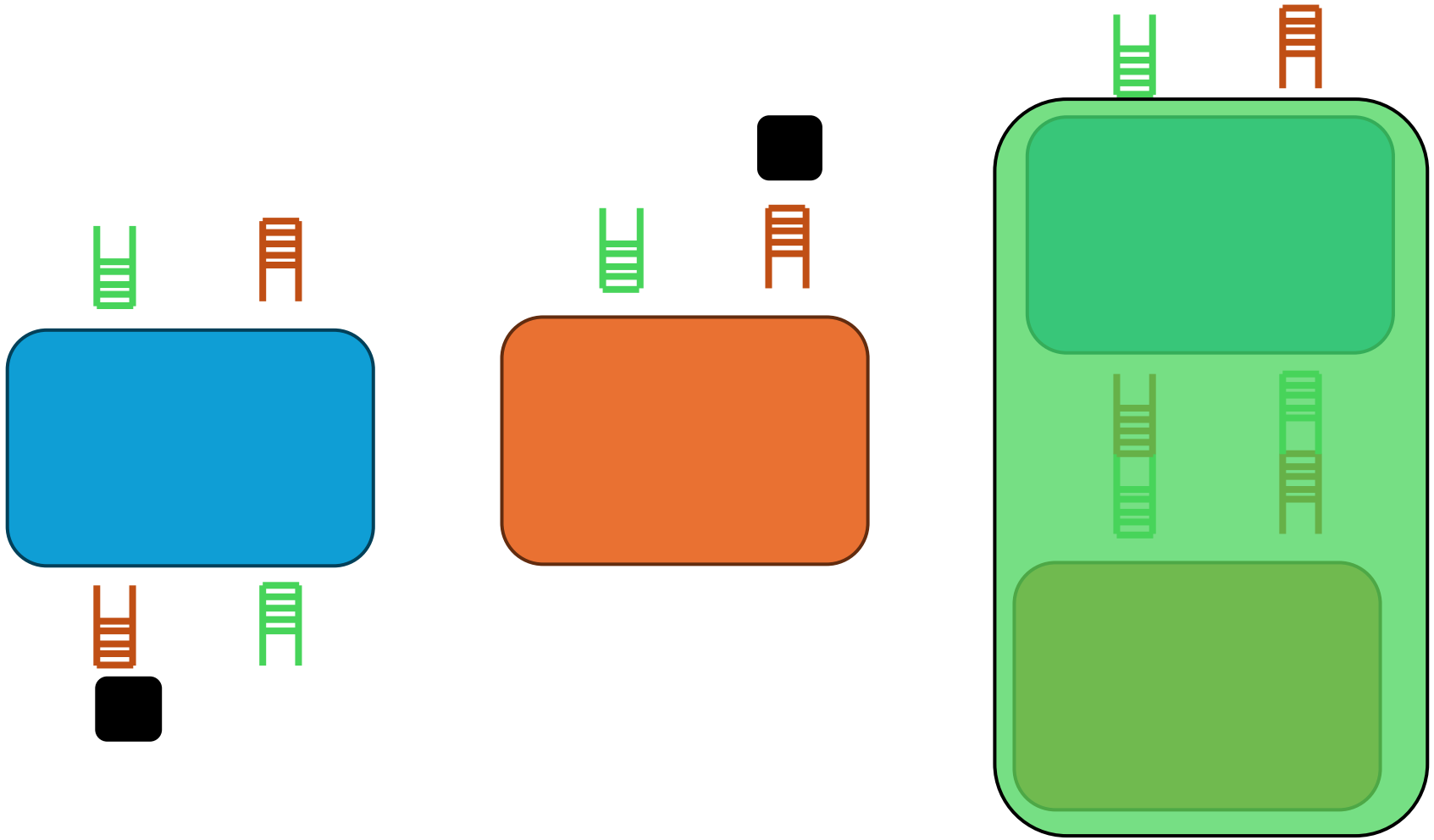
Example



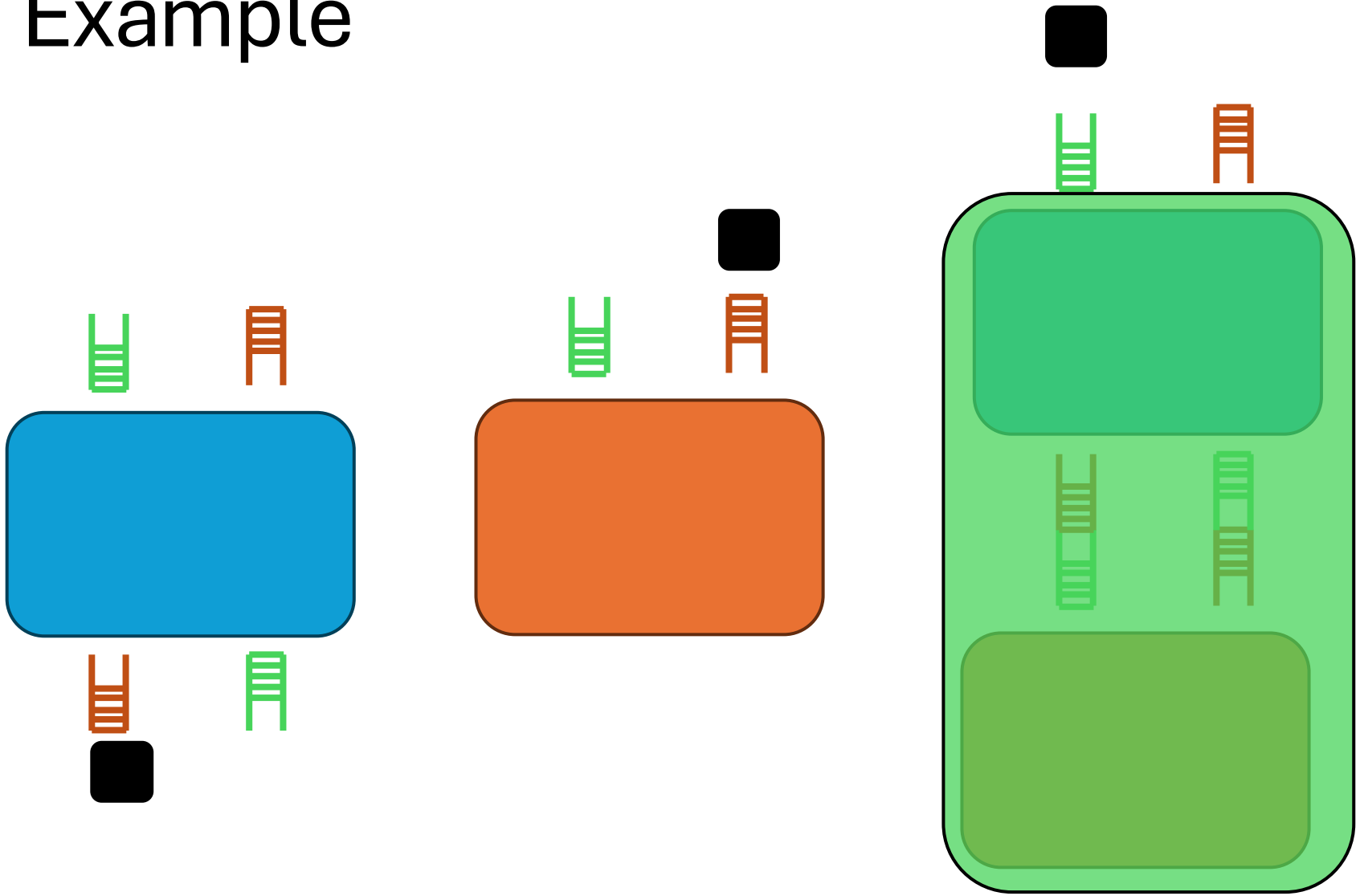
Example



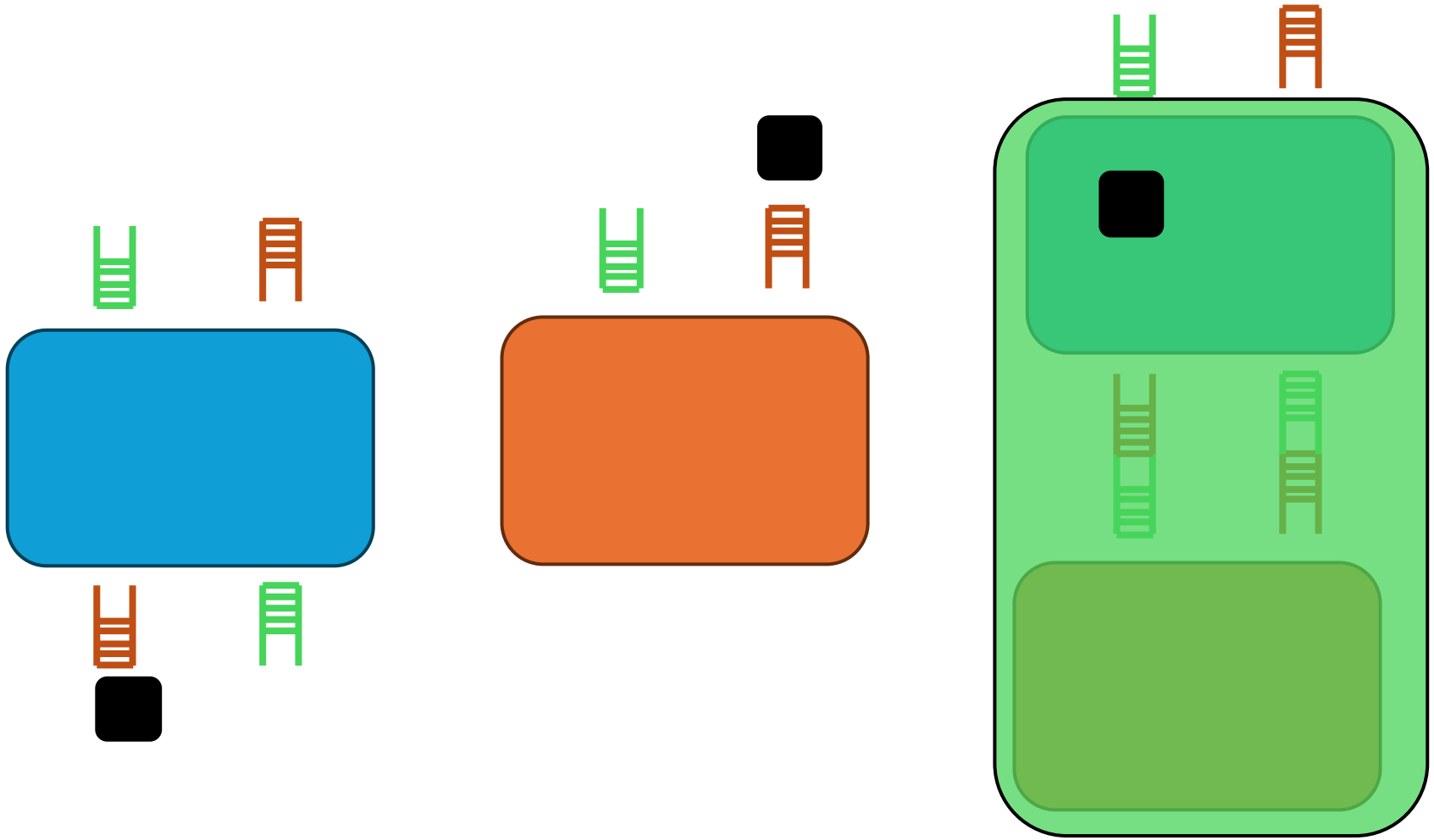
Example



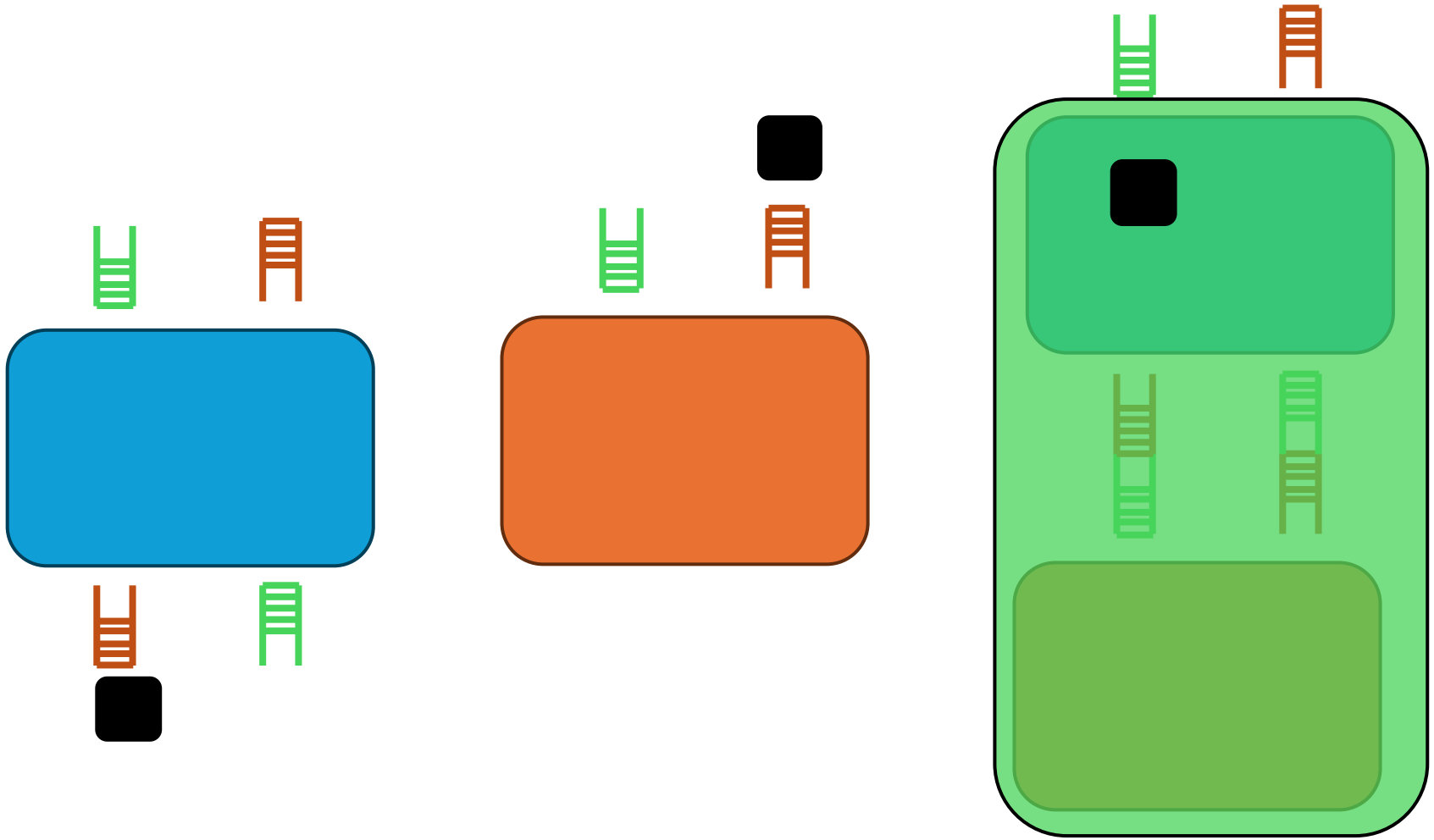
Example



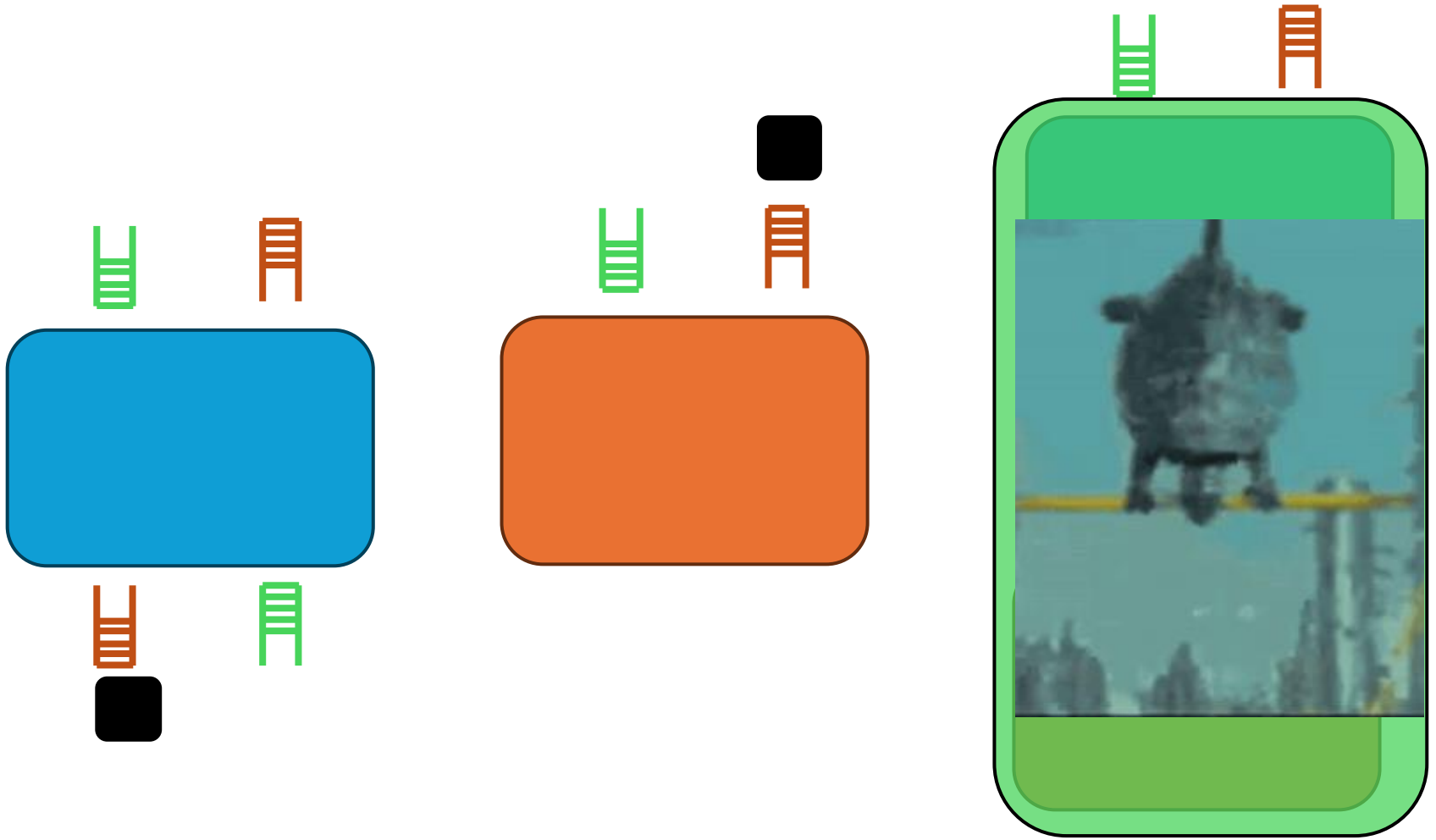
Example



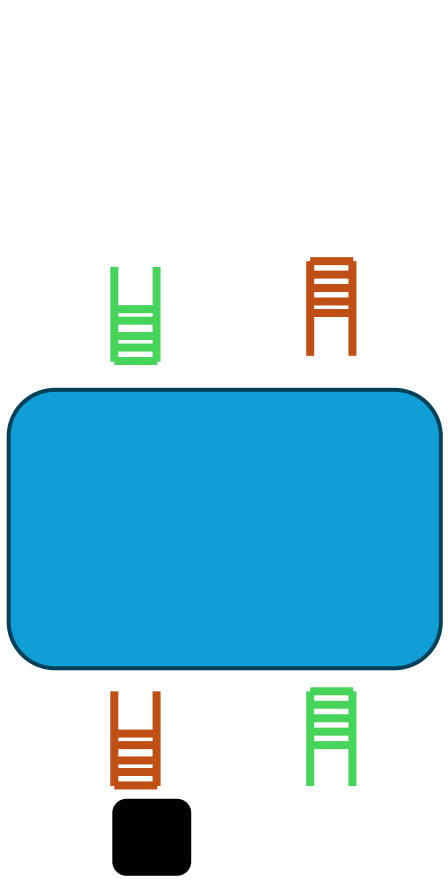
Example

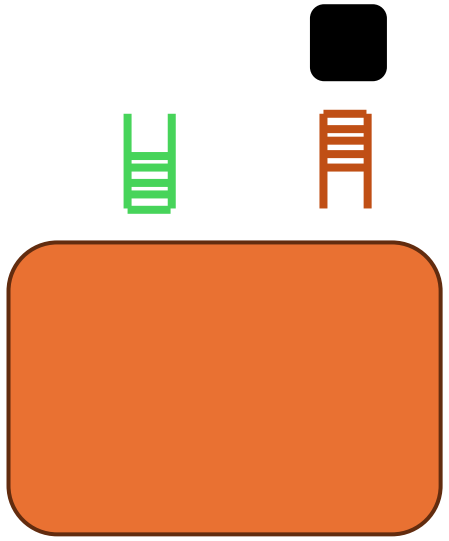


Example

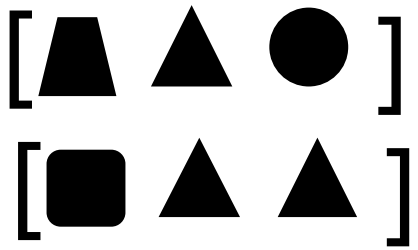


Example

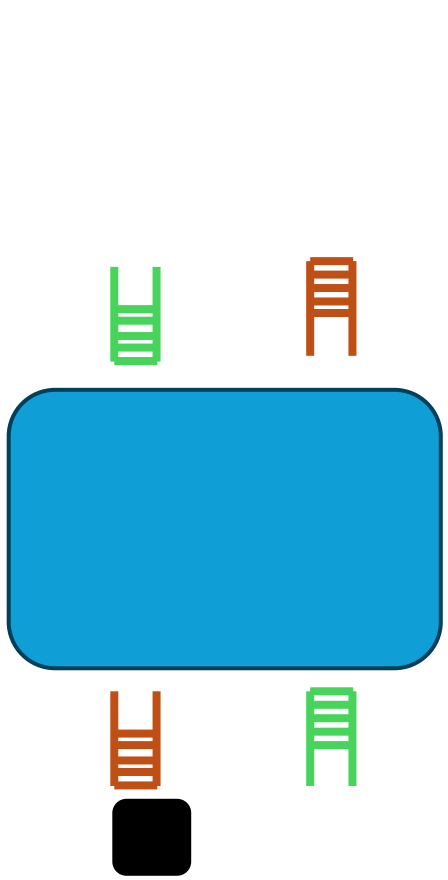


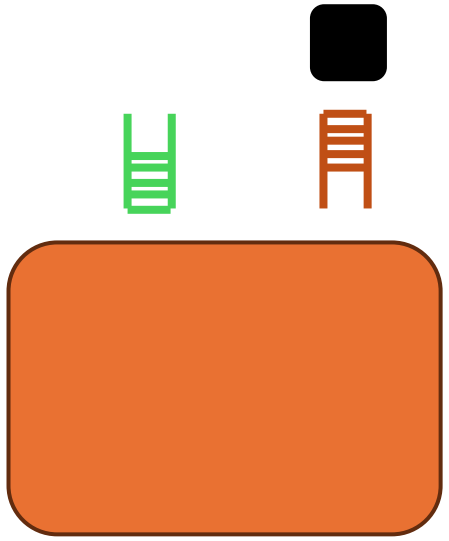




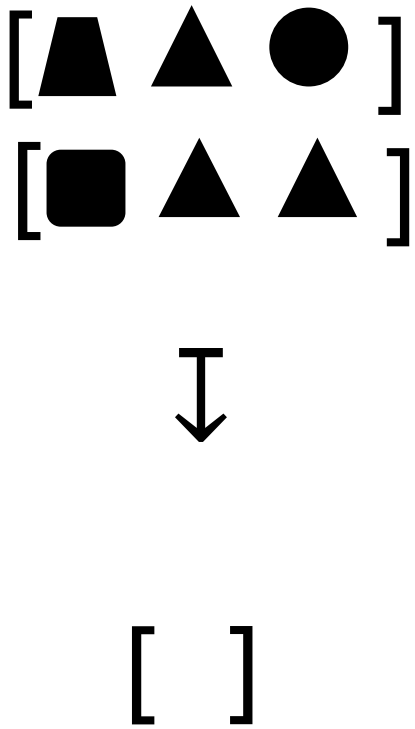


Example



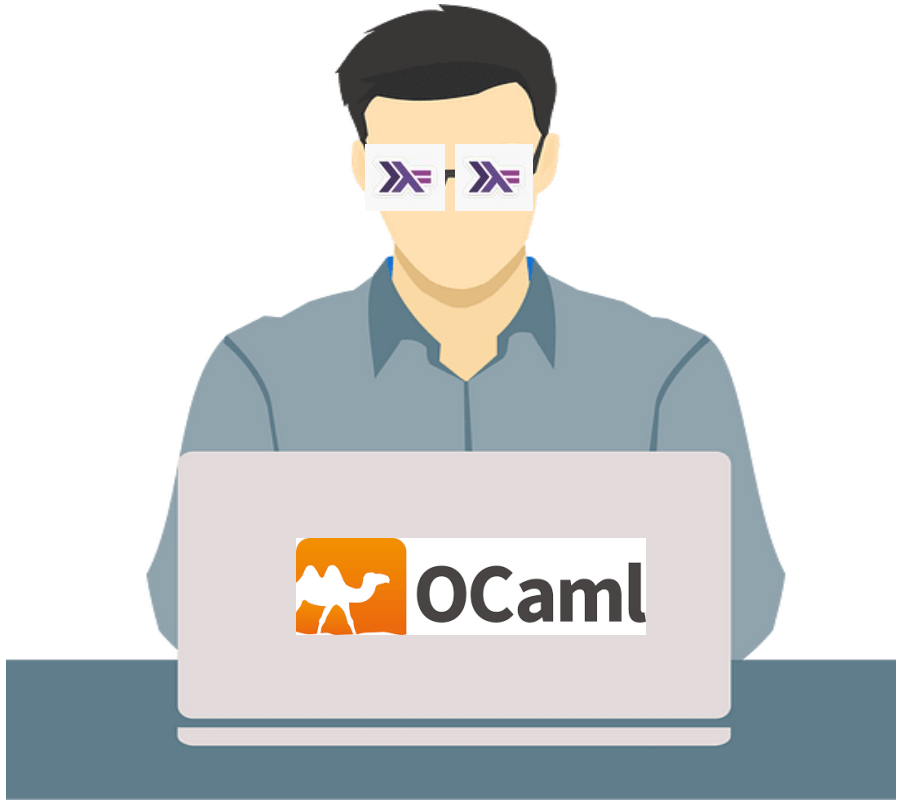




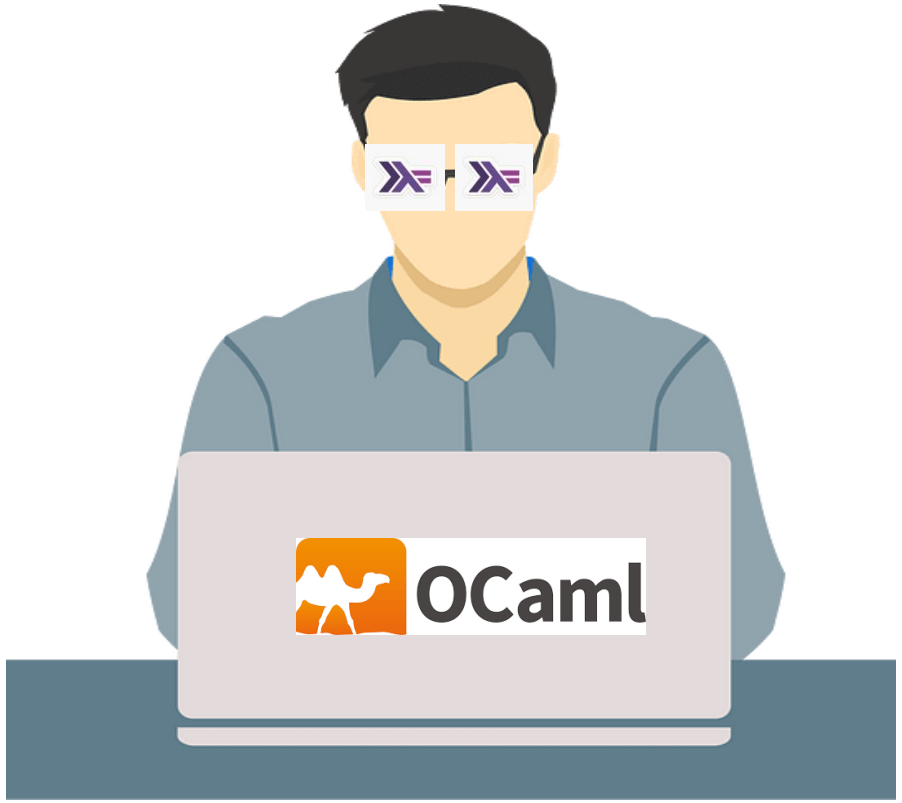


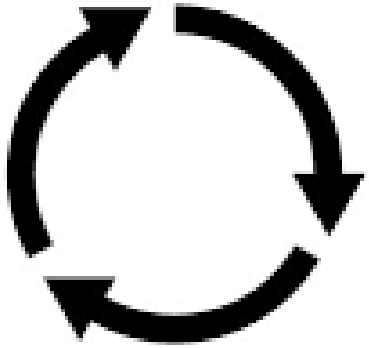
Programming Languages

Programming Languages

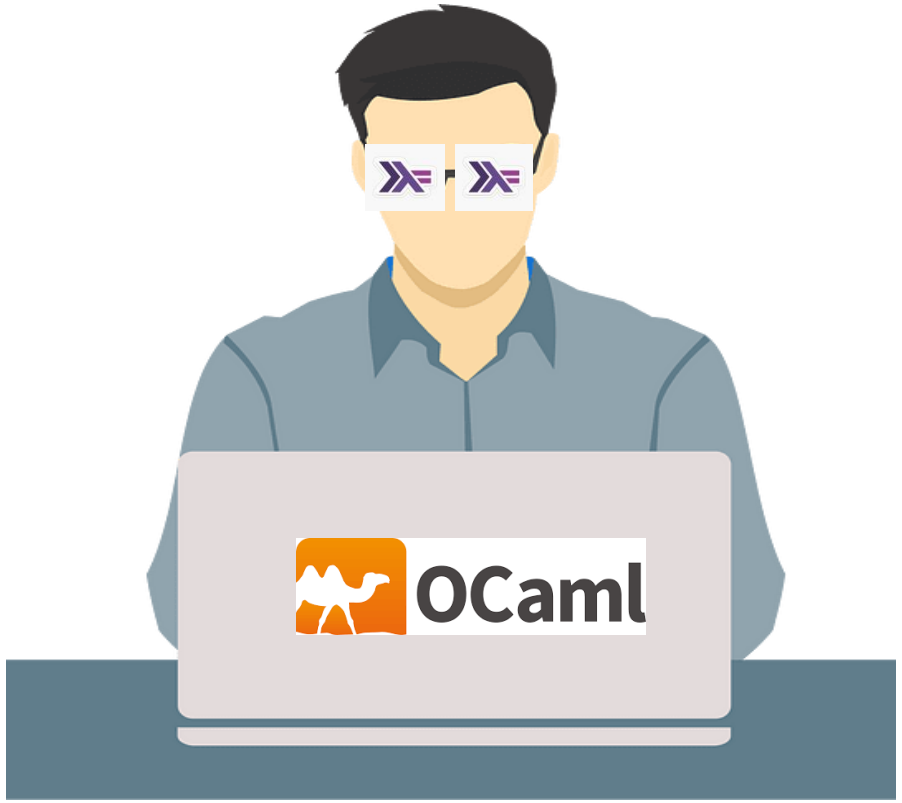


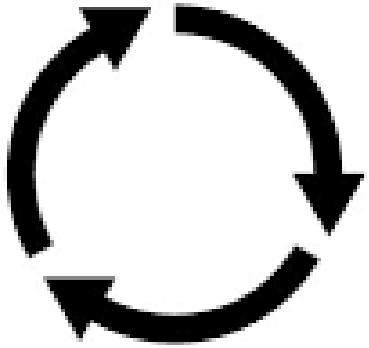
Programming Languages

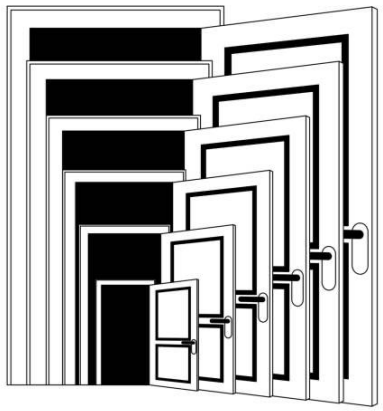




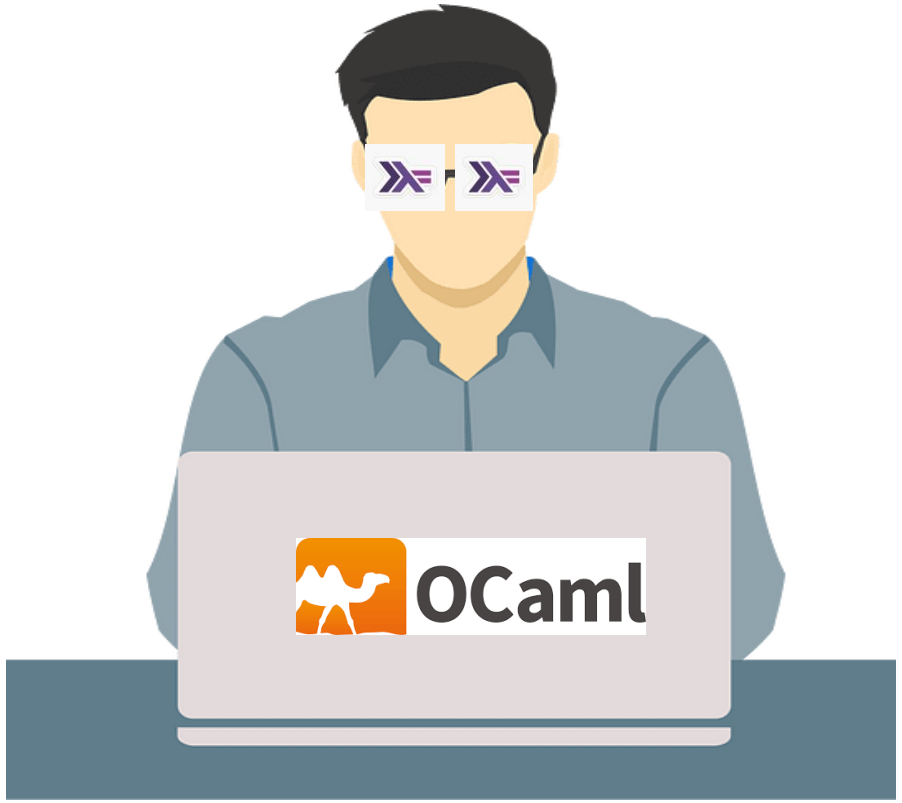
Programming Languages



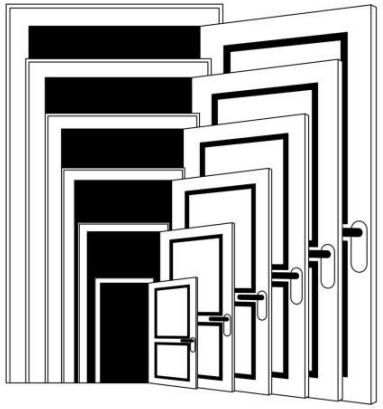


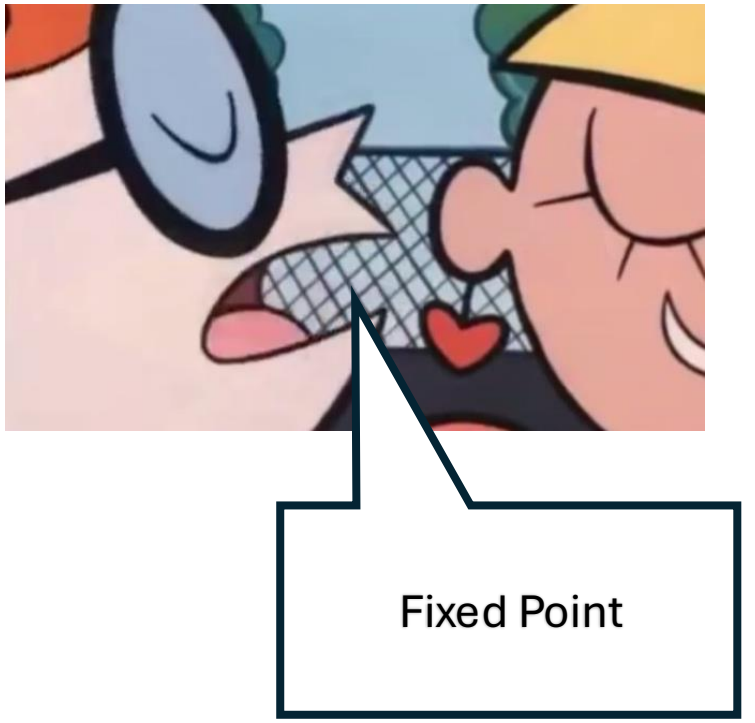


Programming Languages



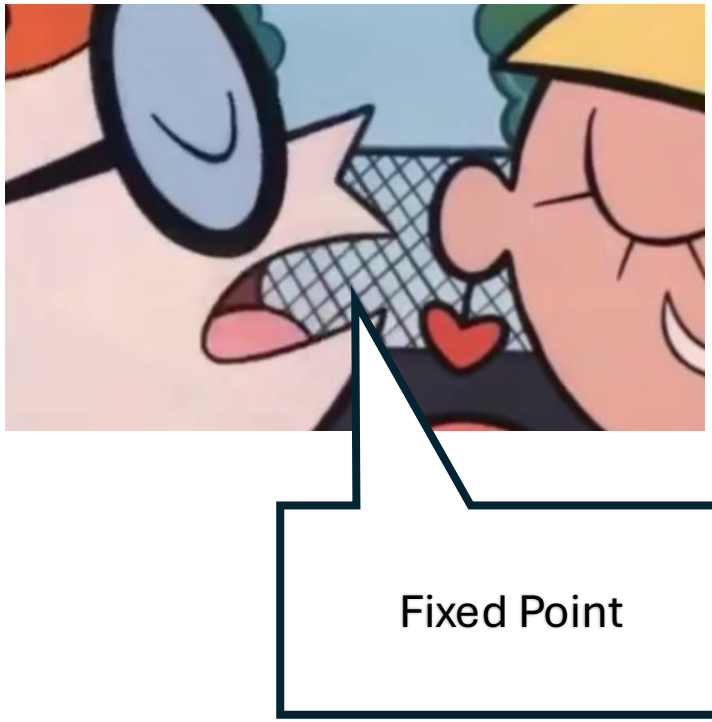






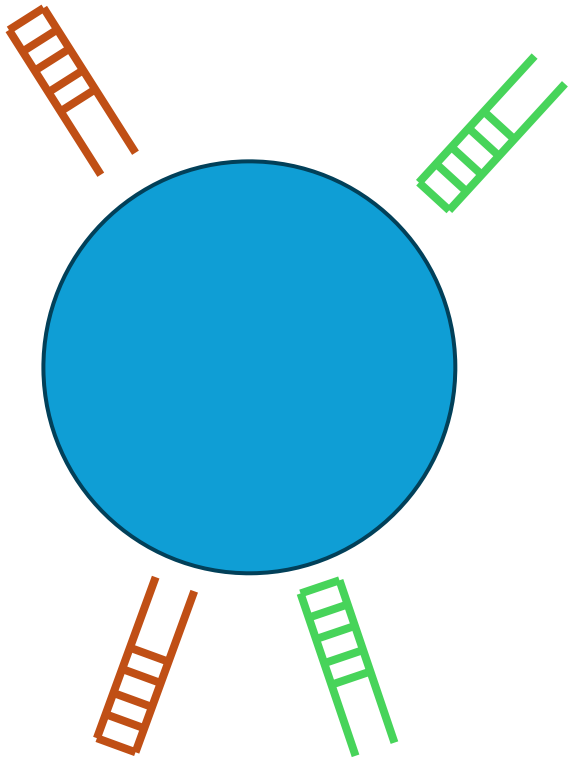
Programming Languages



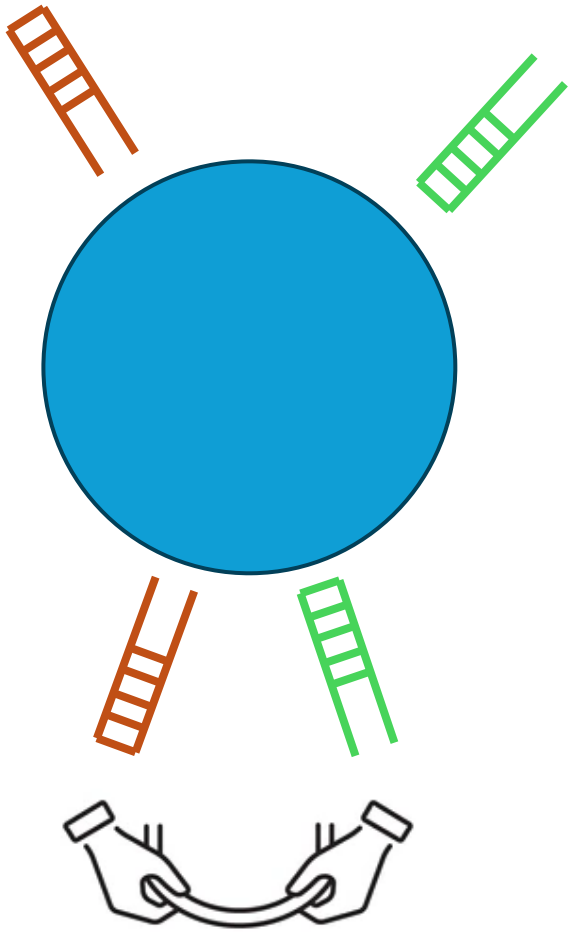


Fixed Points

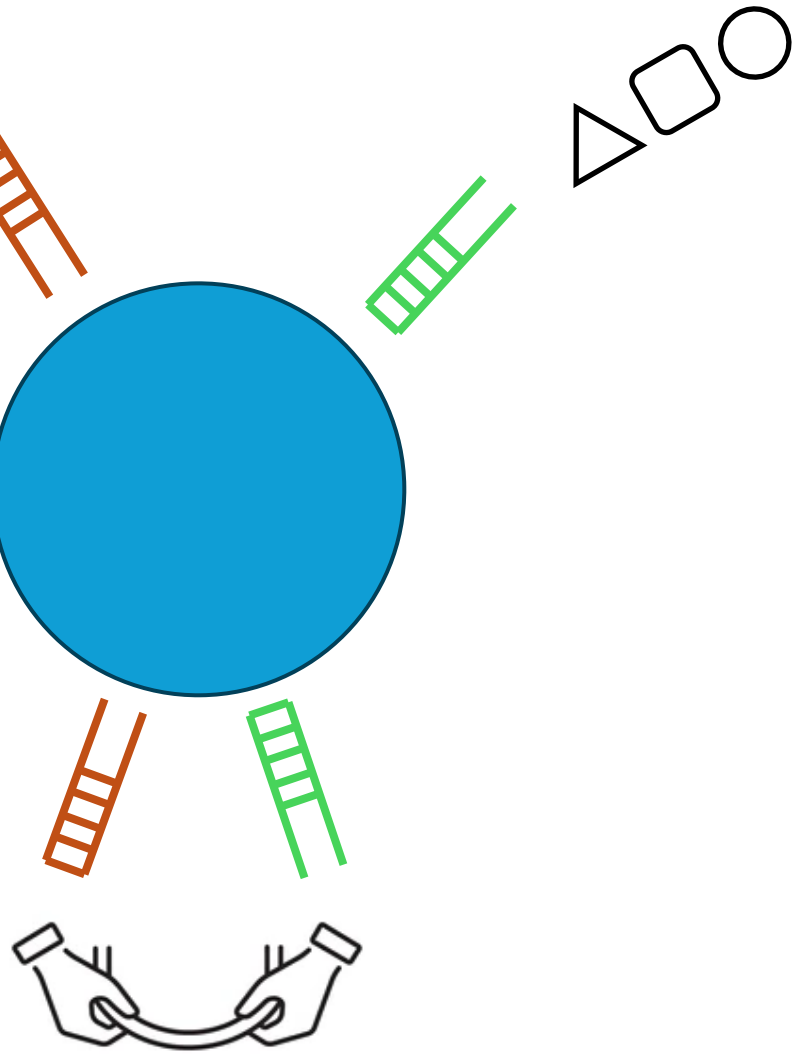
Fixed Points



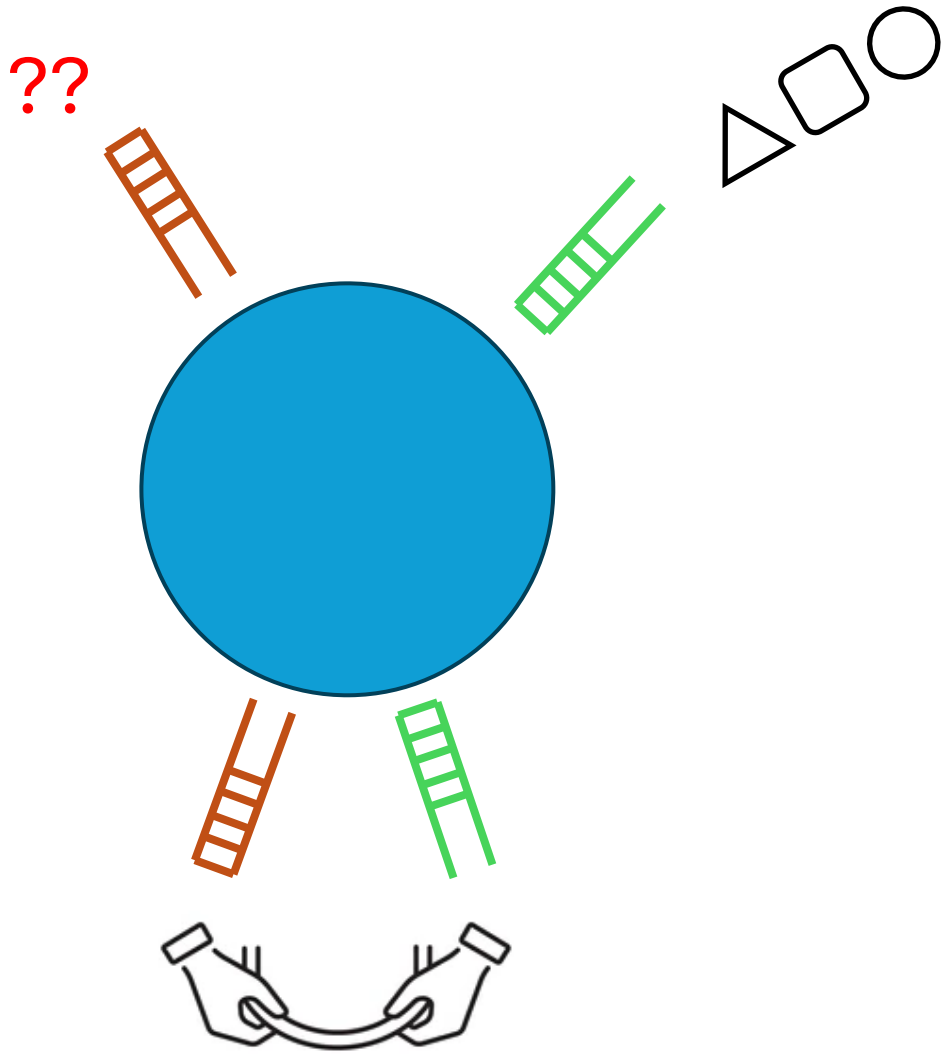
Fixed Points



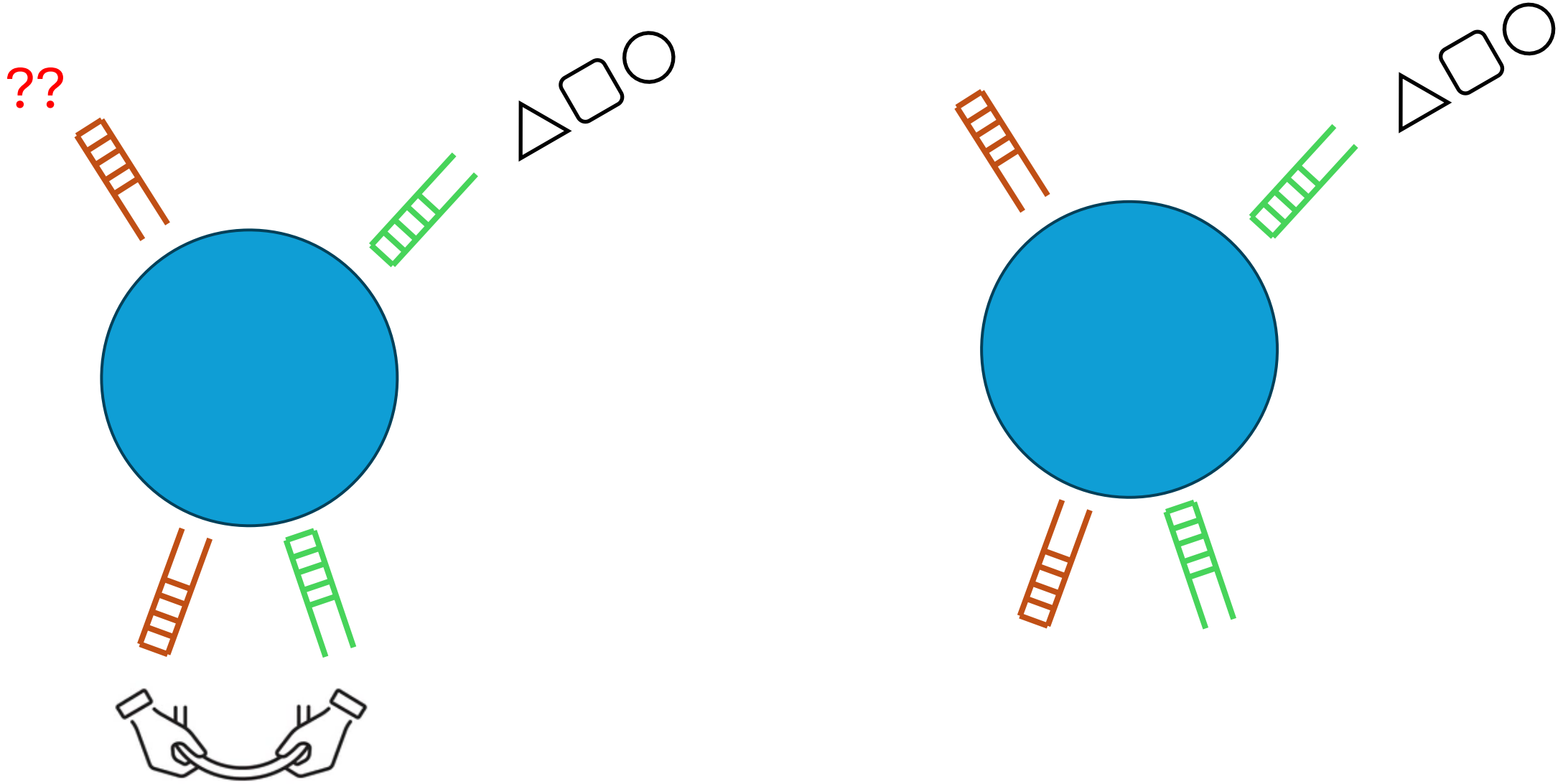
Fixed Points



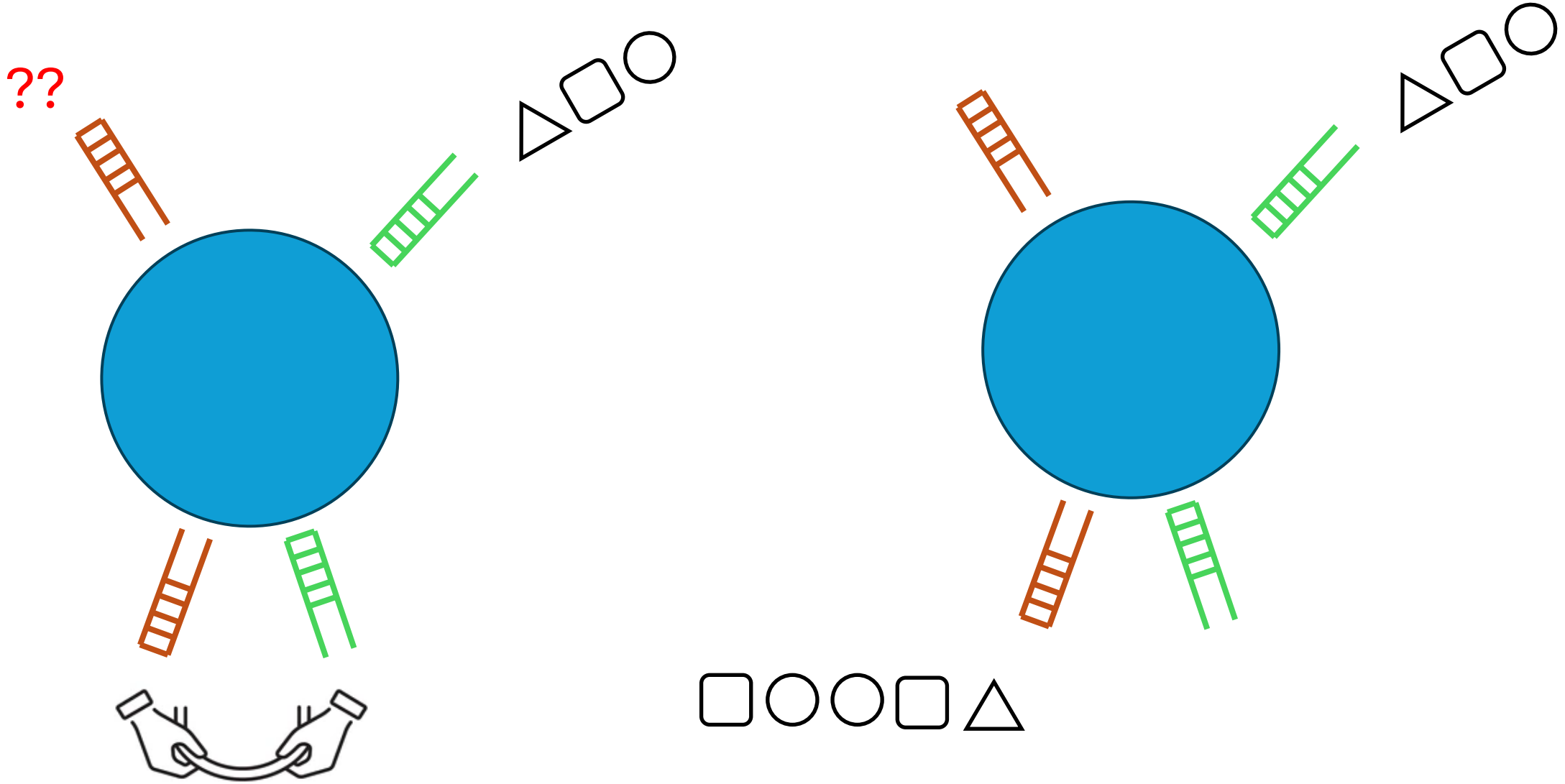
Fixed Points



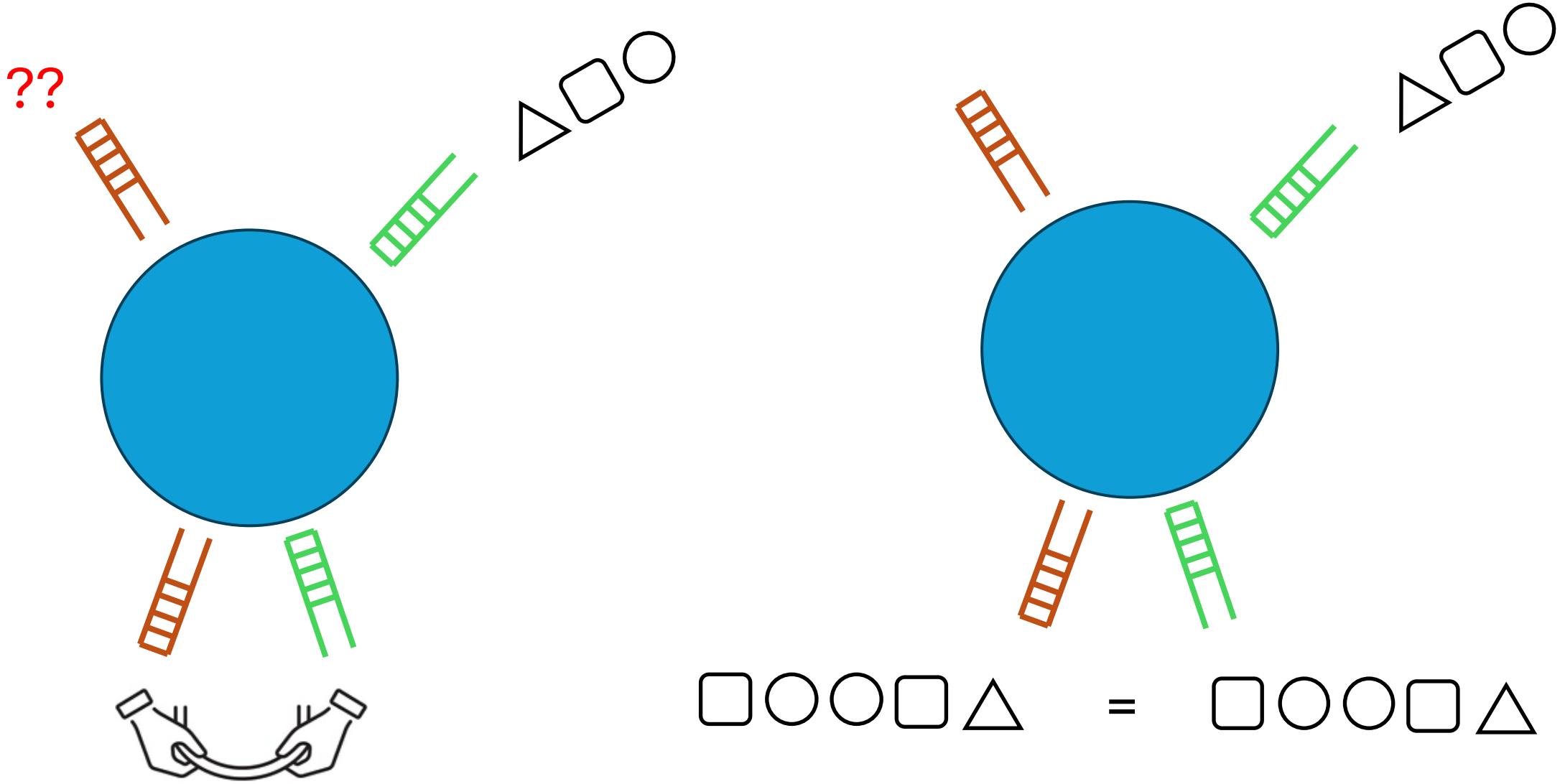
Fixed Points



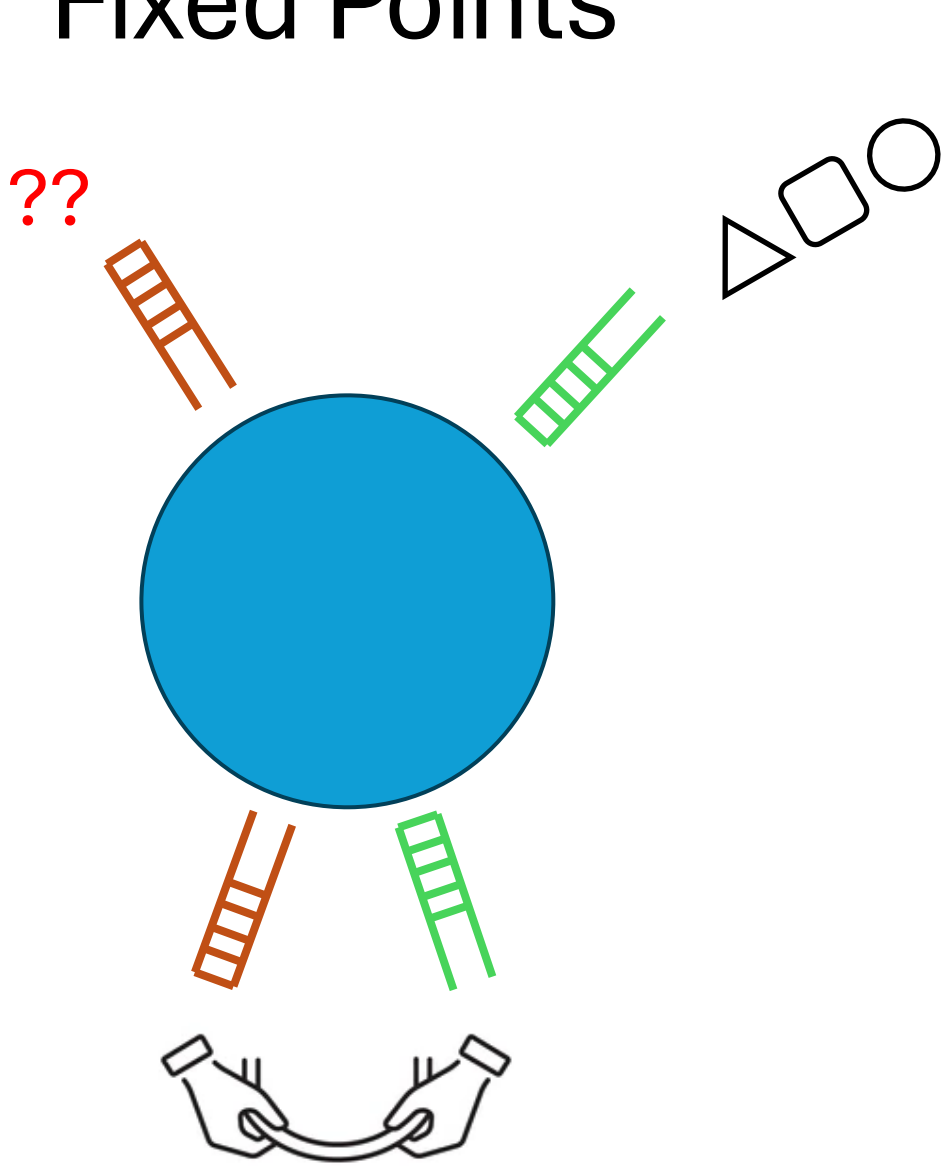
Fixed Points

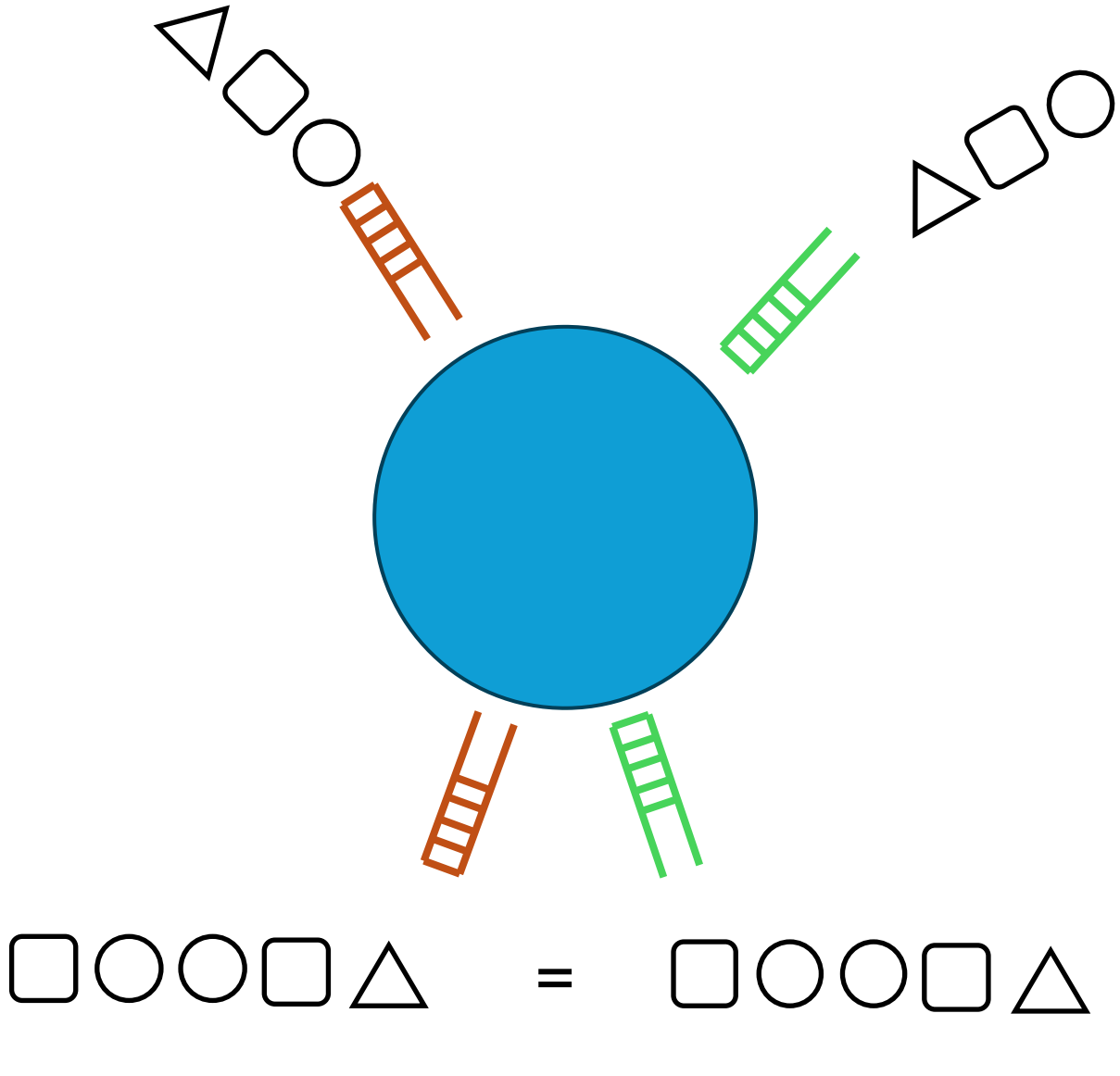


Fixed Points



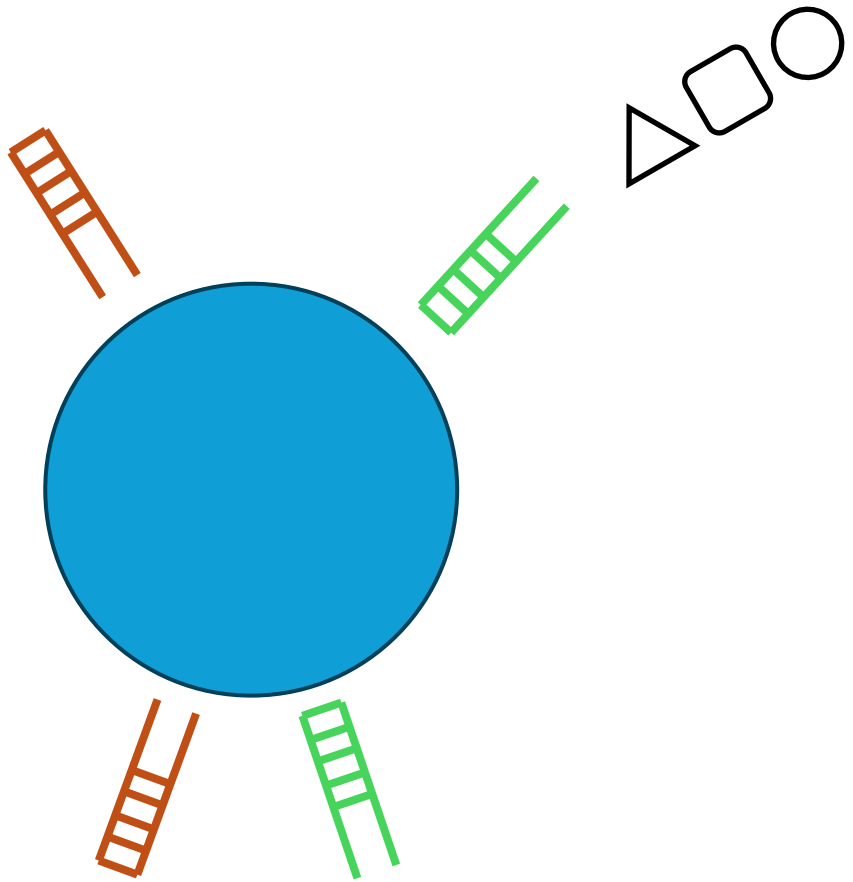
Fixed Points

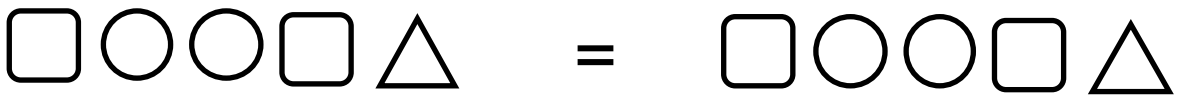




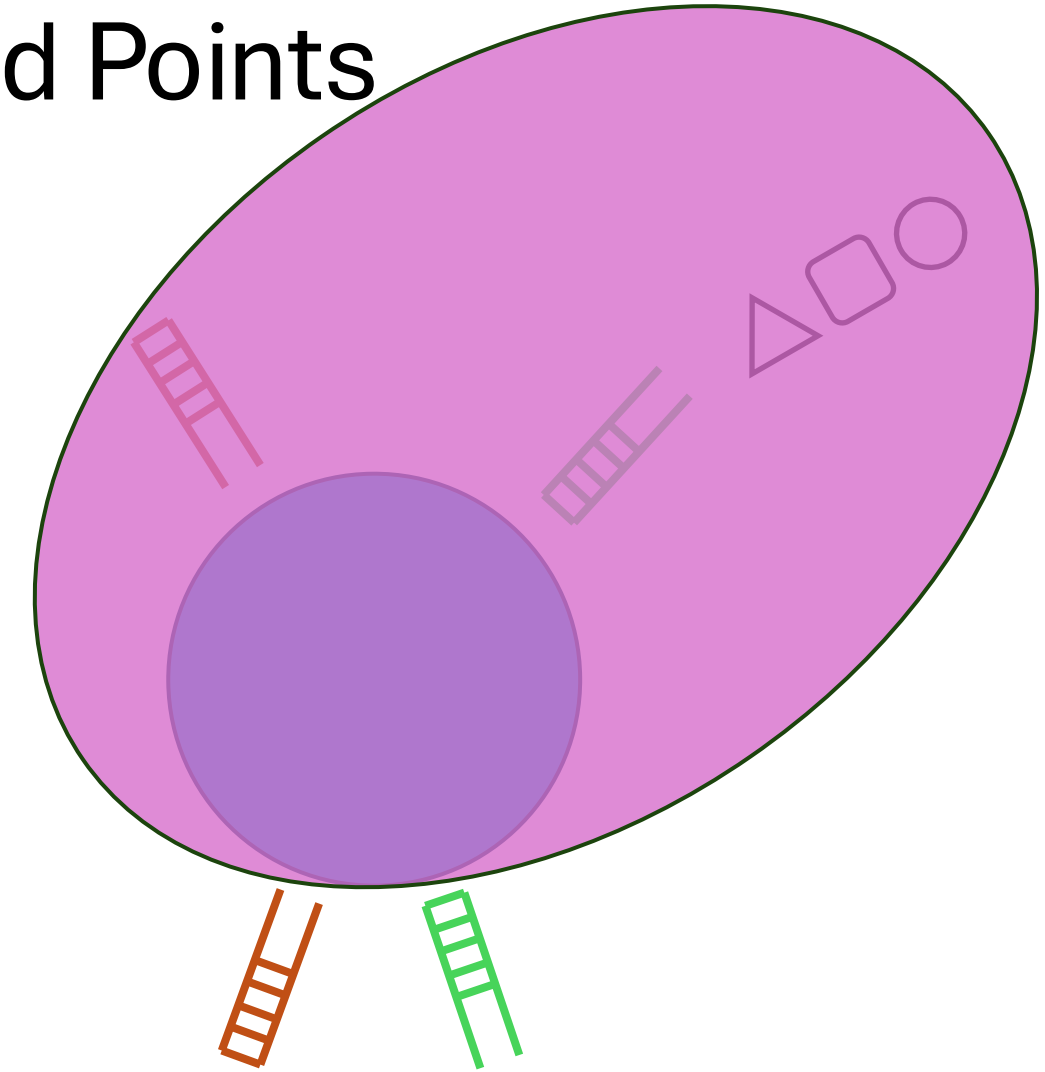
Fixed Points

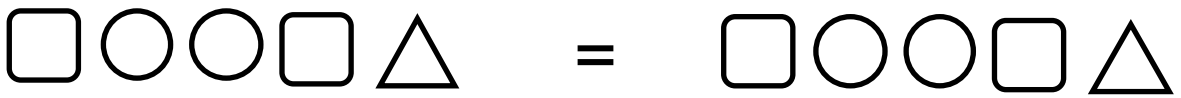
Fixed Points



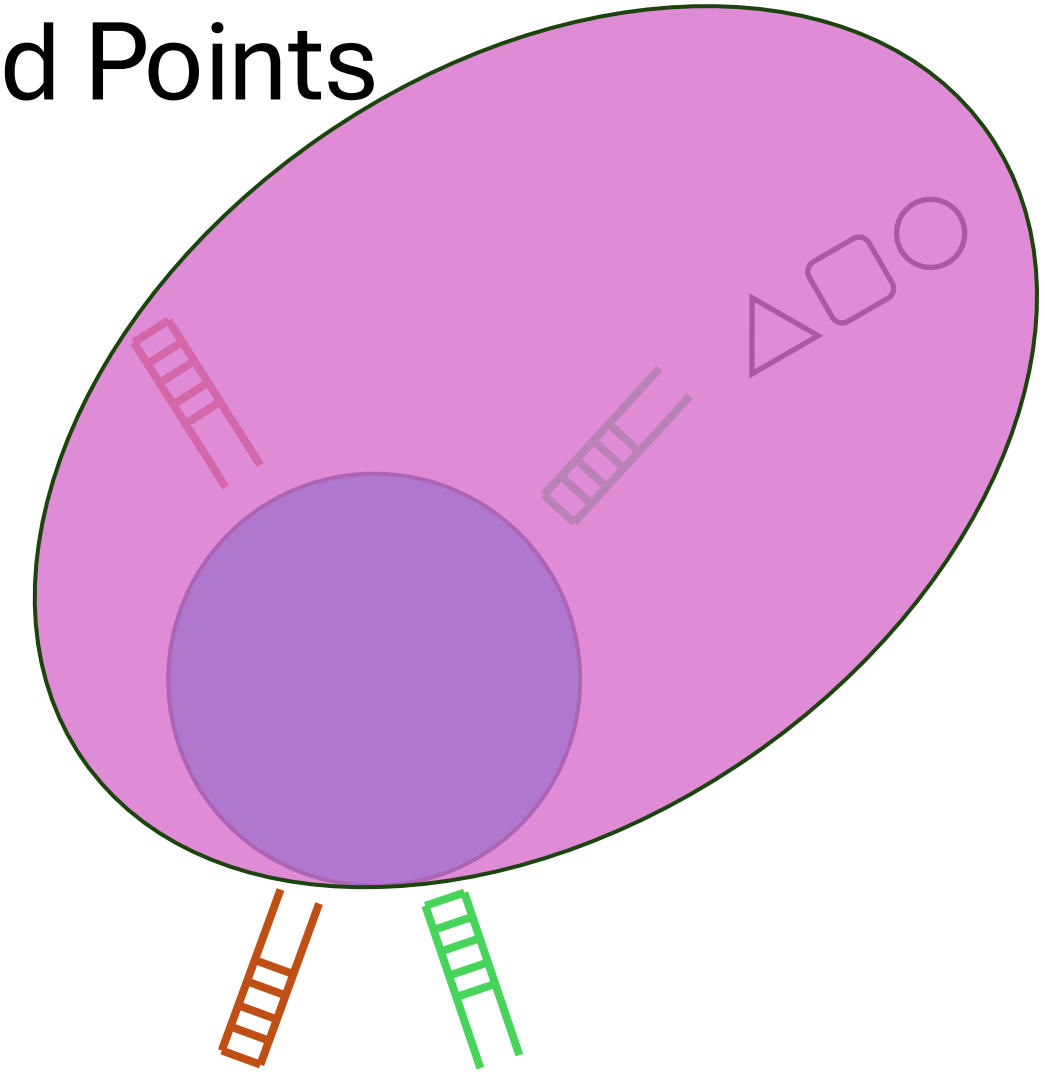


Fixed Points

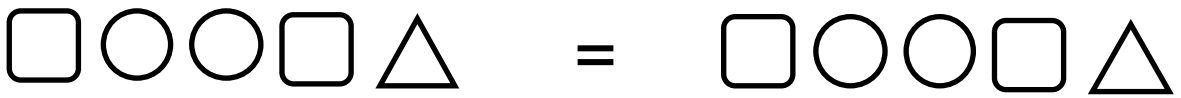




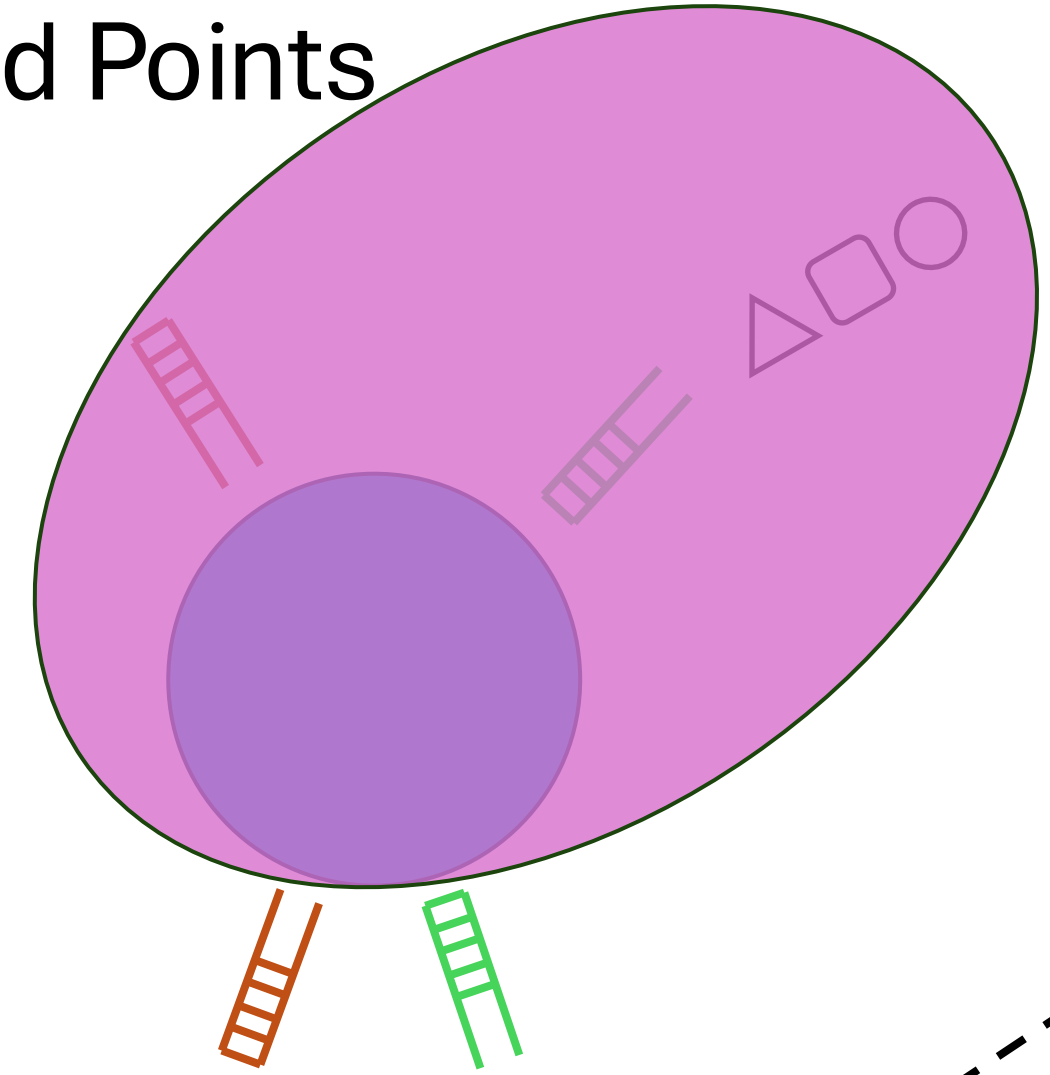
Fixed Points



Function $f(seq)$



Fixed Points

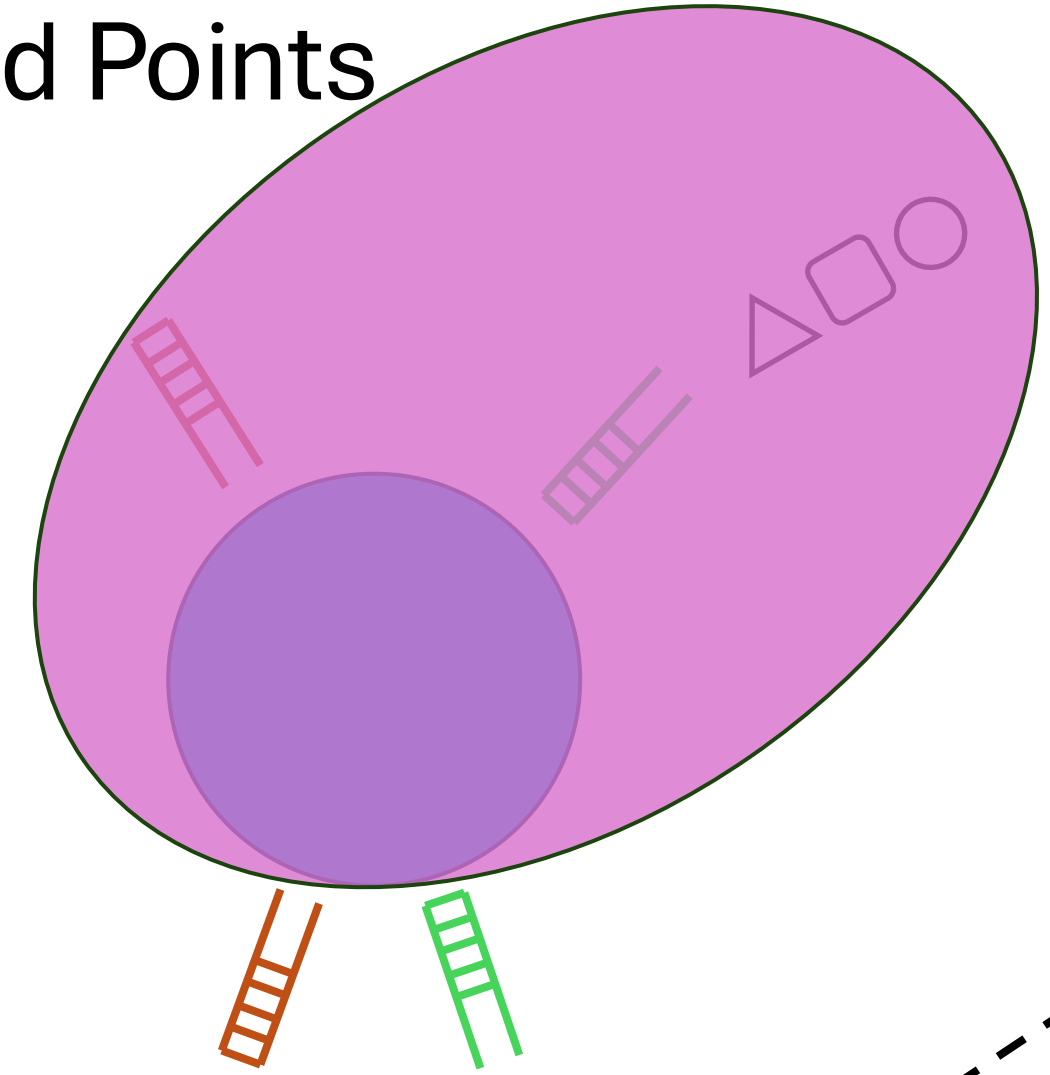


Function $f(seq)$

Input

▢○○▢△ = ▢○○▢△

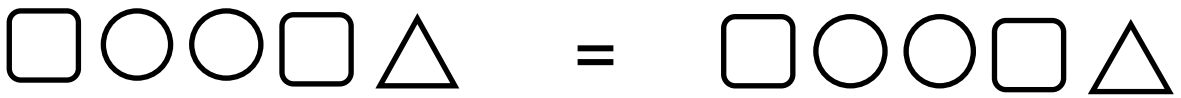
Fixed Points



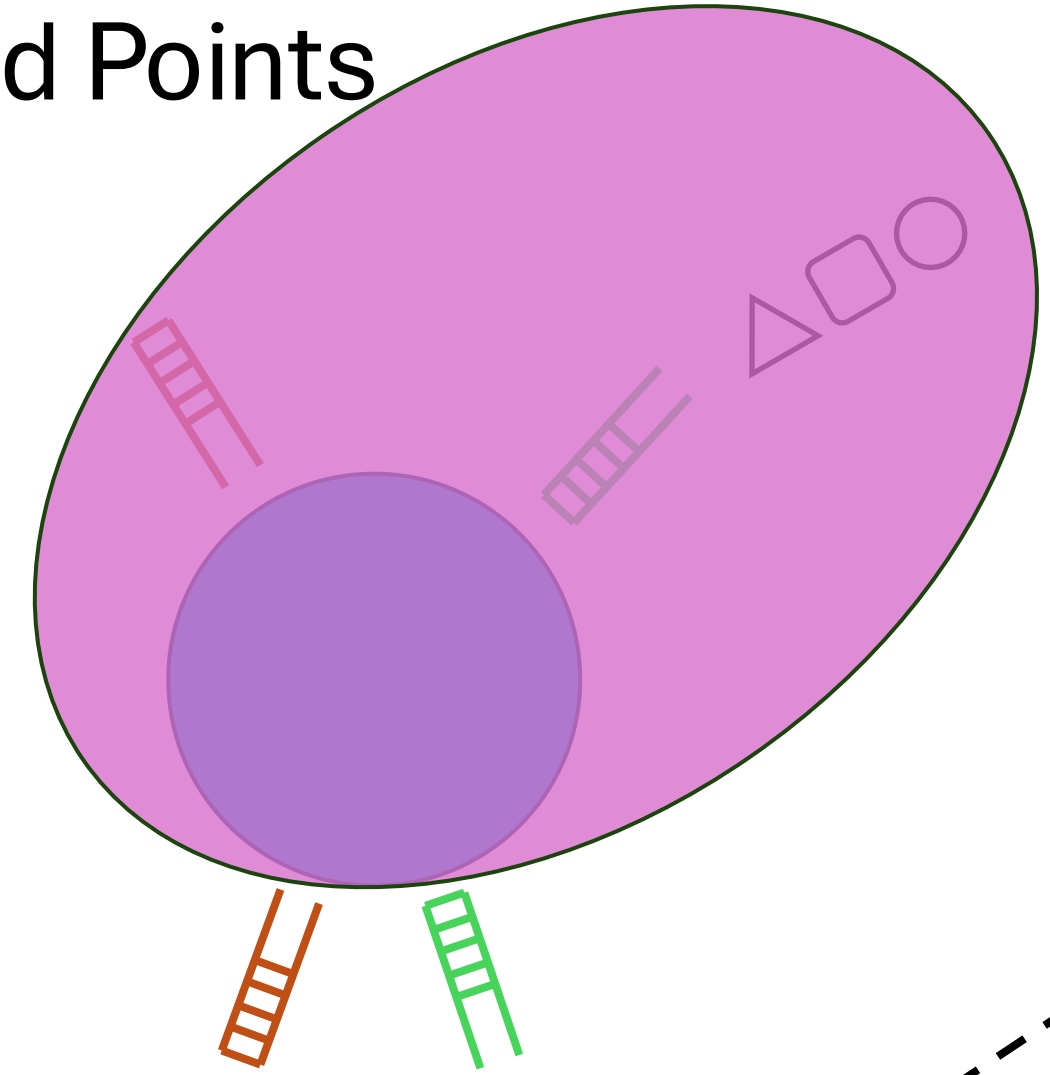
Function $f(seq)$

Output

Input



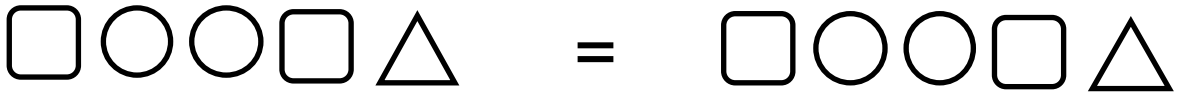
Fixed Points



Function $f(seq)$

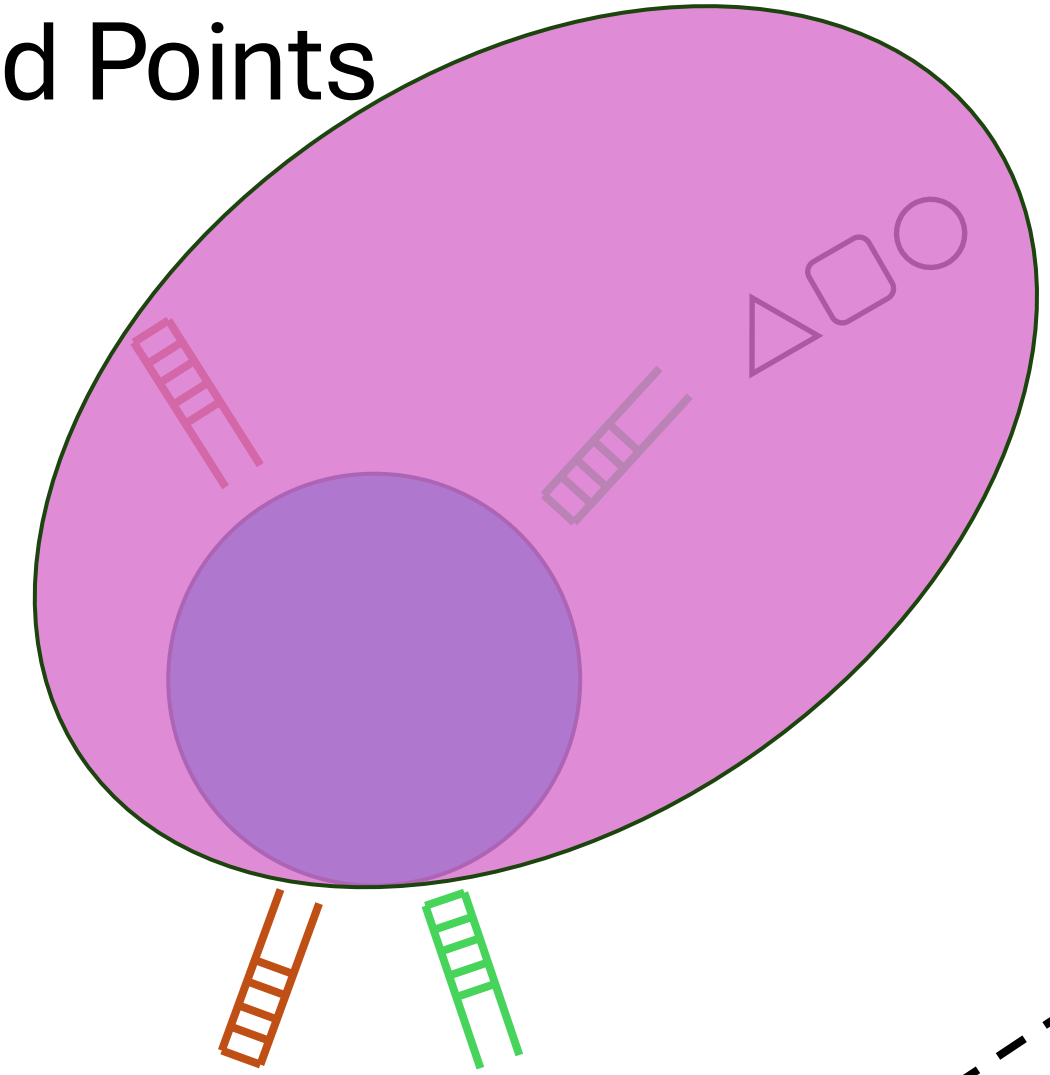
Output

Input



$$f(seq) = seq$$

Fixed Points



Output

Input

▢○○▢△ = ▢○○▢△

Fun



$f(seq) = seq$

We just discovered Kahn Process Networks

We just discovered Kahn Process Networks

INFORMATION PROCESSING 74 – NORTH-HOLLAND PUBLISHING COMPANY (1974)

THE SEMANTICS OF A SIMPLE LANGUAGE FOR PARALLEL PROGRAMMING

Gilles KAHN

IRIA-Laboria, Domaine de Voluceau, 78150 Rocquencourt, France

and

Commissariat à l'Energie Atomique, France

In this paper, we describe a simple language for parallel programming. Its semantics is studied thoroughly. The desirable properties of this language and its deficiencies are exhibited by this theoretical study. Basic results on parallel program schemata are given. We hope in this way to make a case for a more formal (i.e. mathematical) approach to the design of languages for systems programming and the design of operating systems.

We just discovered Kahn Process Networks

INFORMATION PROCESSING 74 – NORTH-HOLLAND PUBLISHING COMPANY (1974)

THE SEMANTICS OF A SIMPLE LANGUAGE FOR PARALLEL PROGRAMMING

Gilles KAHN

IRIA-Laboria, Domaine de Voluceau, 78150 Rocquencourt, France

and

Commissariat à l'Energie Atomique, France

In this paper, we describe a simple language for parallel programming. Its semantics is studied thoroughly. The desirable properties of this language and its deficiencies are exhibited by this theoretical study. Basic results on parallel program schemata are given. We hope in this way to make a case for a more formal (i.e. mathematical) approach to the design of languages for systems programming and the design of operating systems.



Useful?

Useful?

Kahn Process Networks are a Flexible Alternative to MapReduce

Publisher: IEEE

Cite This PDF

Željko Vrba ; Pål Halvorsen ; Carsten Griwodz ; Paul Beskow **All Authors**

Useful?

Kahn Process Networks are a Fle
Alternative to MapReduce

Publisher: IEEE

Cite This | PDF

Željko Vrba ; Pål Halvorsen ; Carsten Griwodz ; Paul Beskow

SEMICONDUCTOR ENGINEERING
DEEP INSIGHTS FOR THE TECH INDUSTRY

Chines

MENU

SYSTEMS & DESIGN

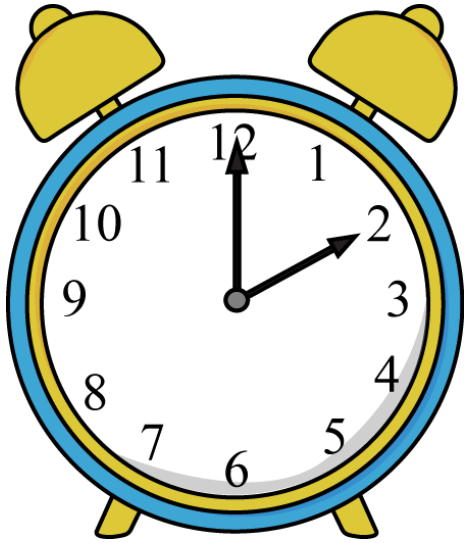
WHITEPAPERS

Kahn Process Network: Parallel Programming Without Races And Non-Determinism

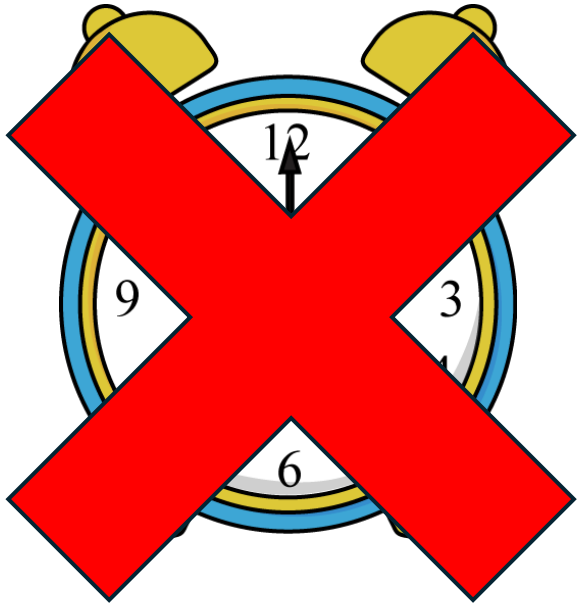
How a 1974 parallel programming model may solve problems with the threading approach, including racing and non-determinism.

More Abstraction?

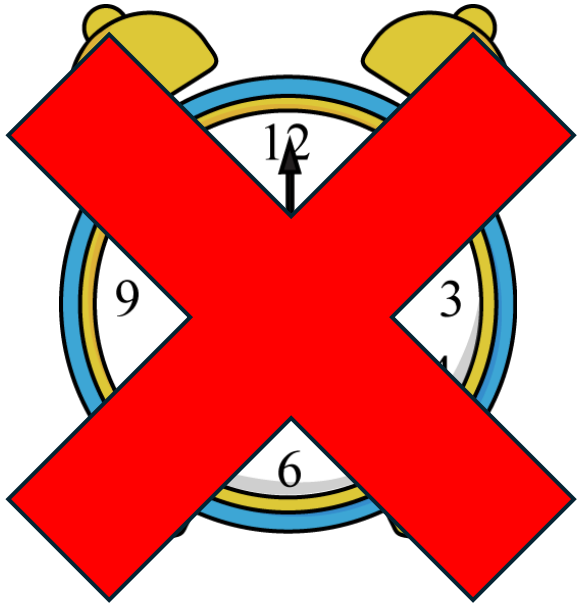
More Abstraction?

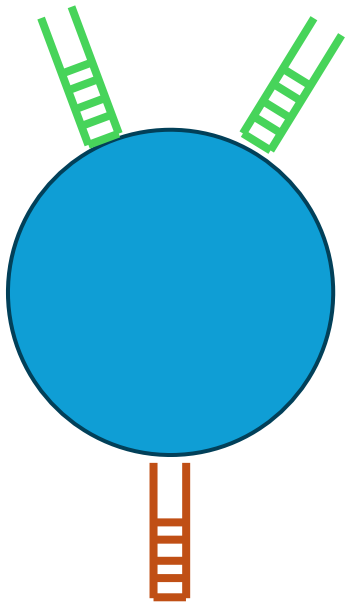


More Abstraction?

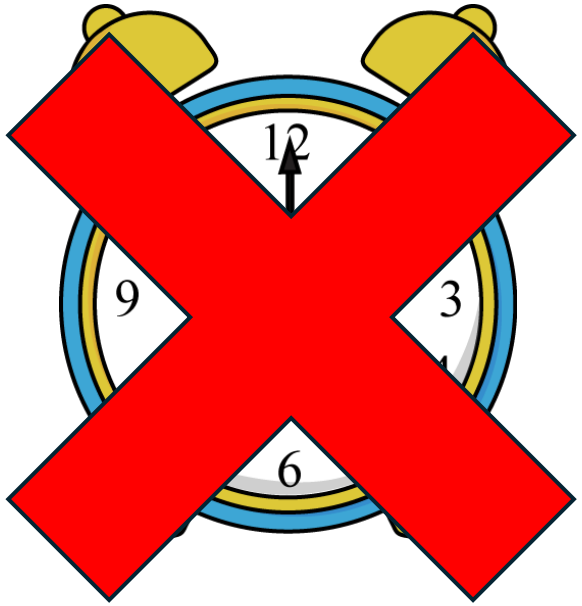


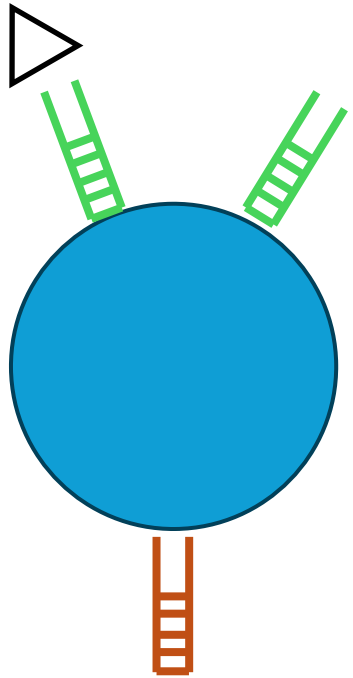
More Abstraction?



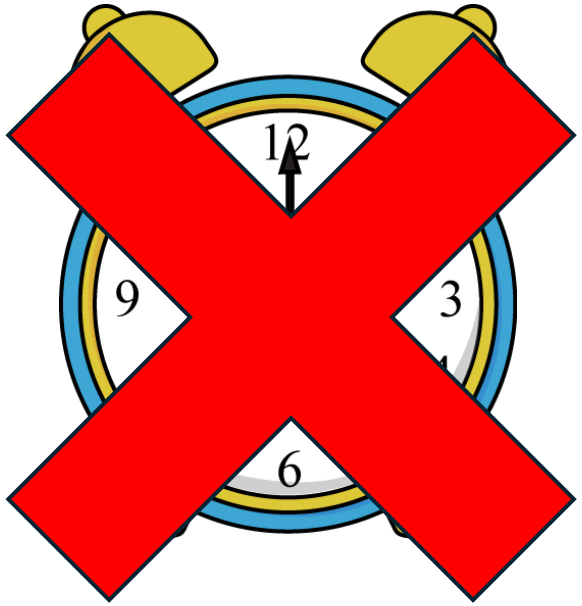


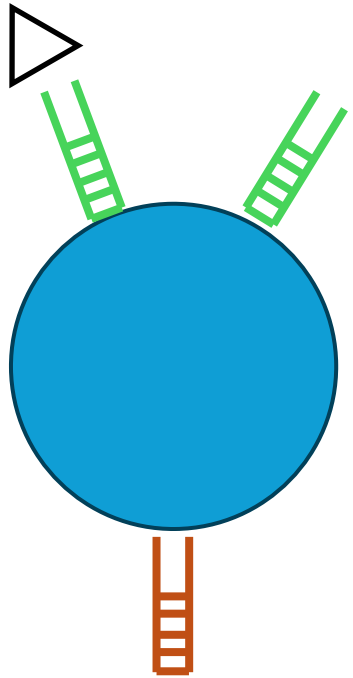
More Abstraction?





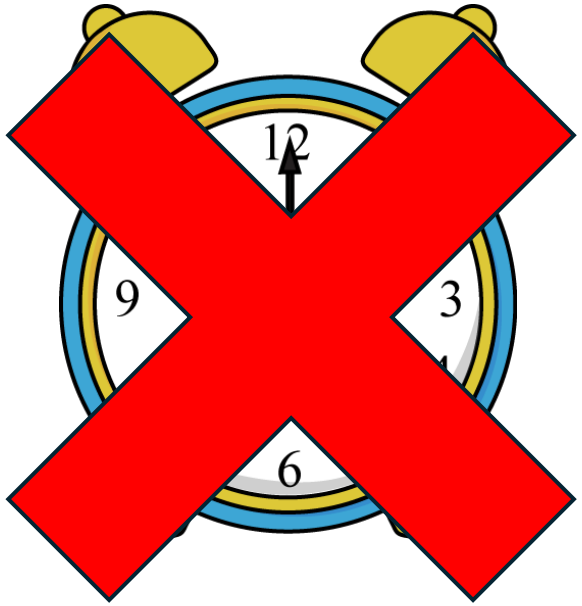
More Abstraction?

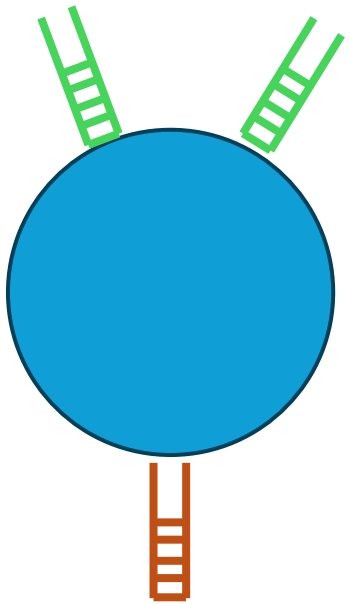




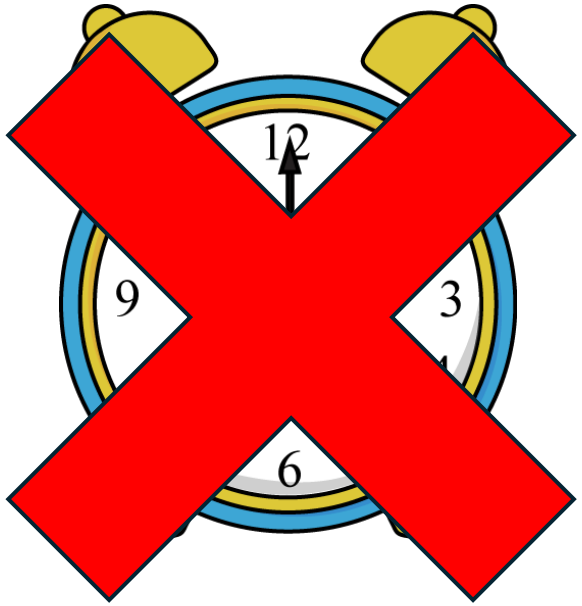
1

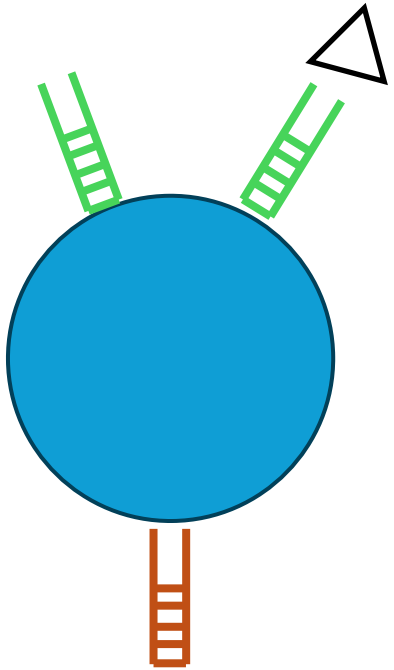
More Abstraction?



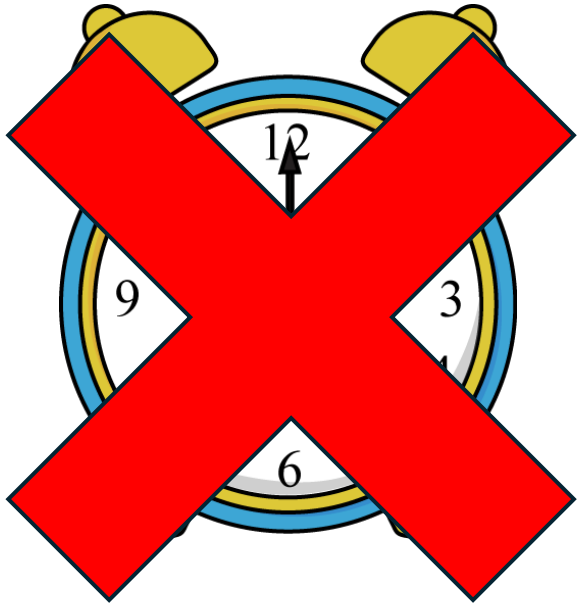


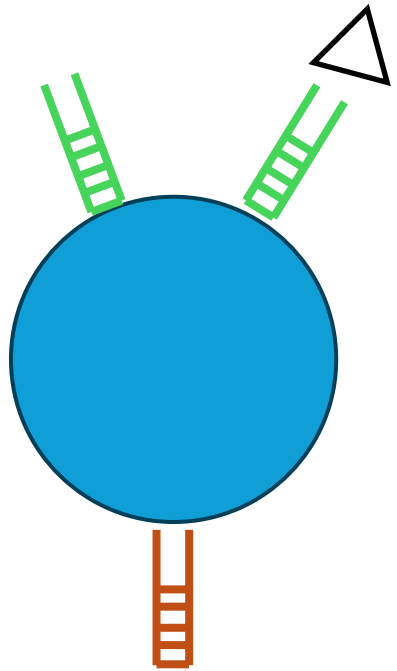
More Abstraction?





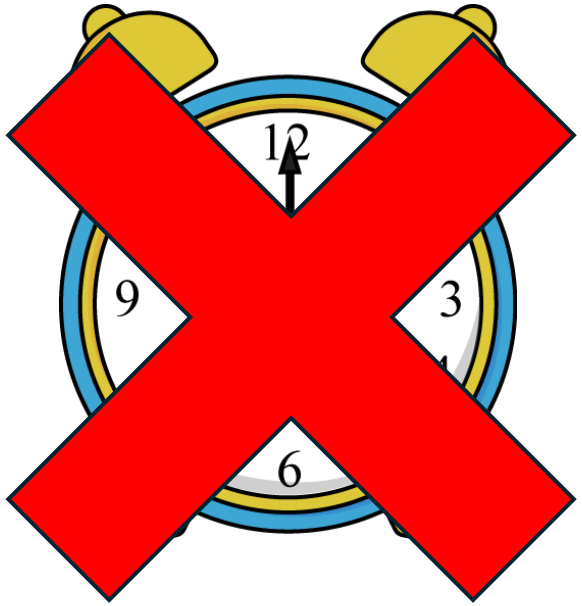
More Abstraction?

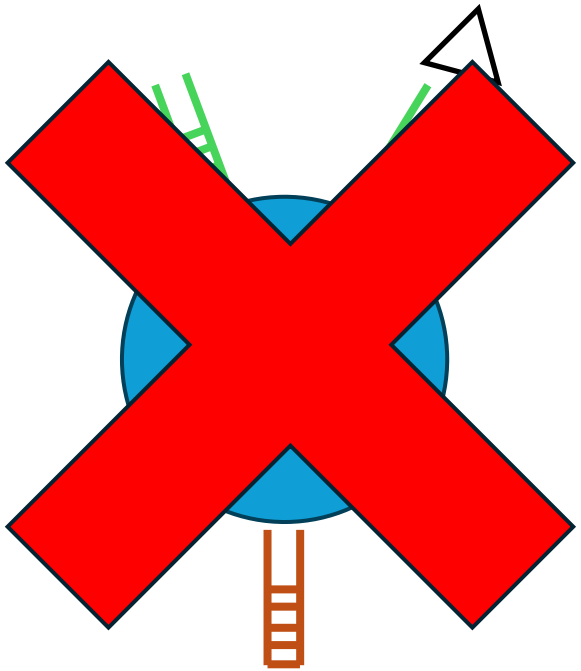




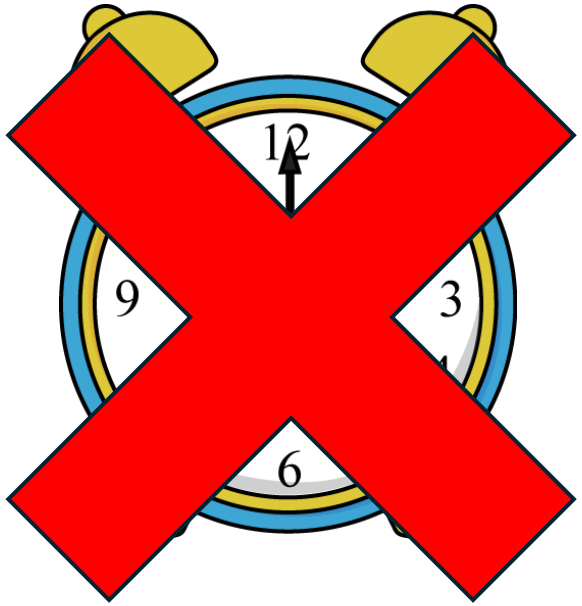
2

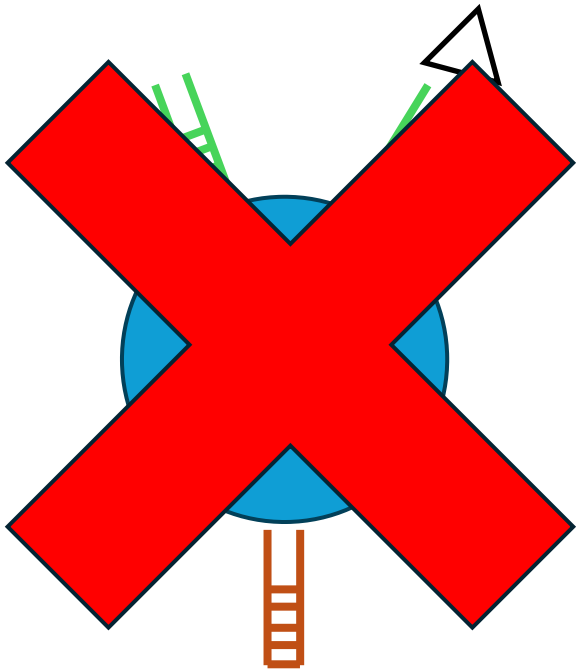
More Abstraction?





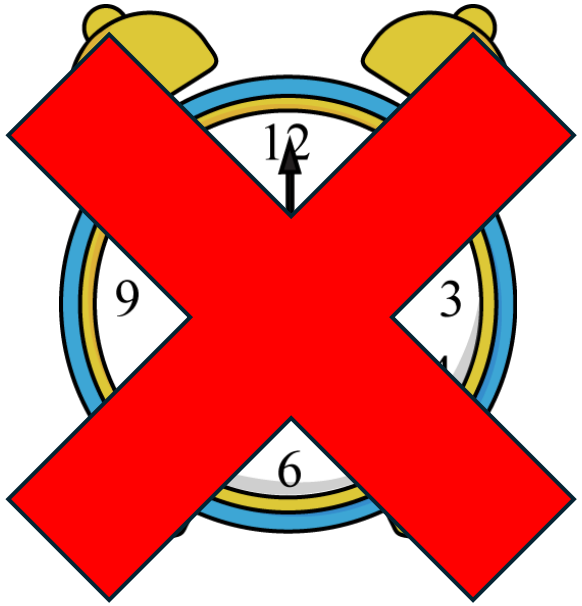
More Abstraction?

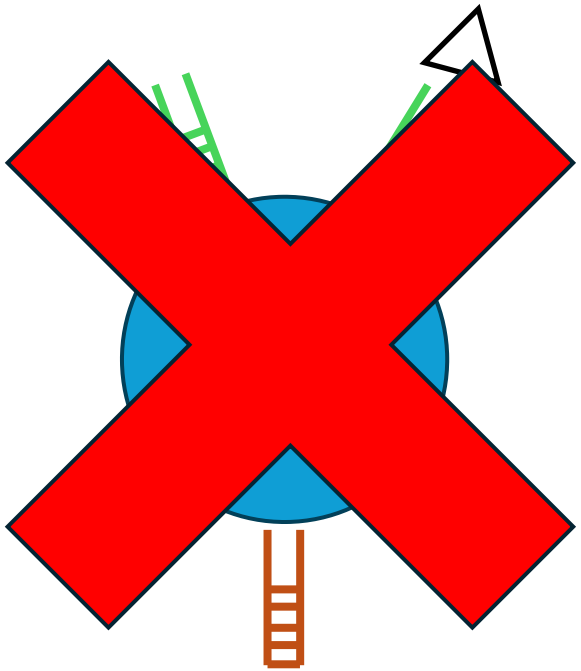




$$\begin{bmatrix} \triangle \\ [\,] \end{bmatrix} \mapsto [\,1\,]$$

More Abstraction?

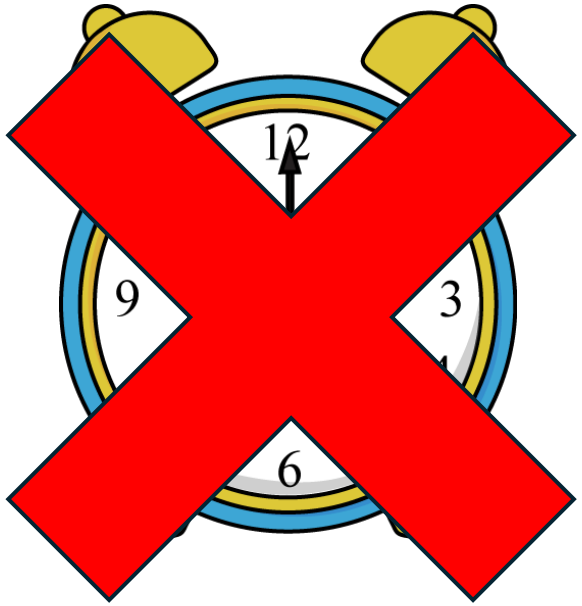


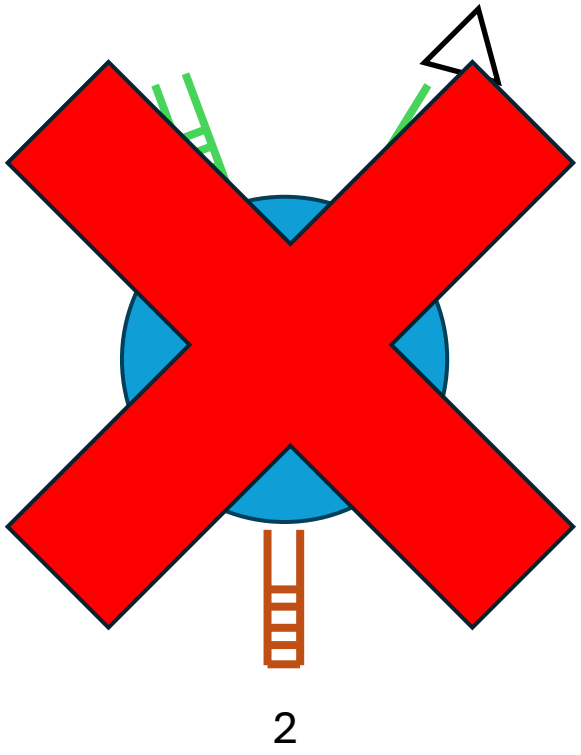


[△] ↦ [1]
[ ]

[ ] ↦ [2]
[△]

More Abstraction?





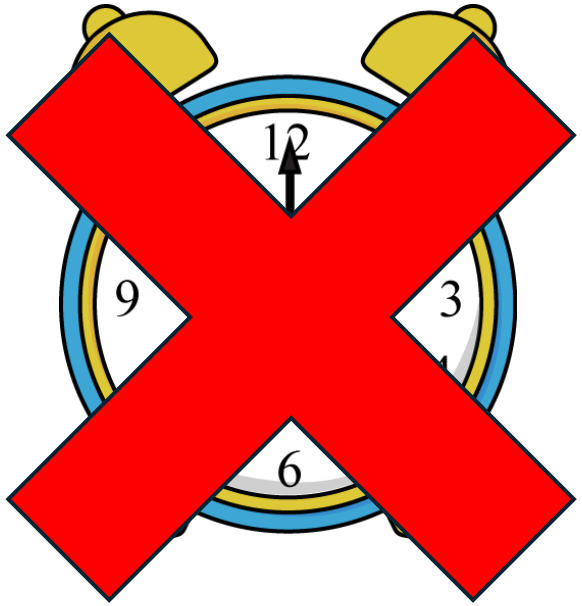
2

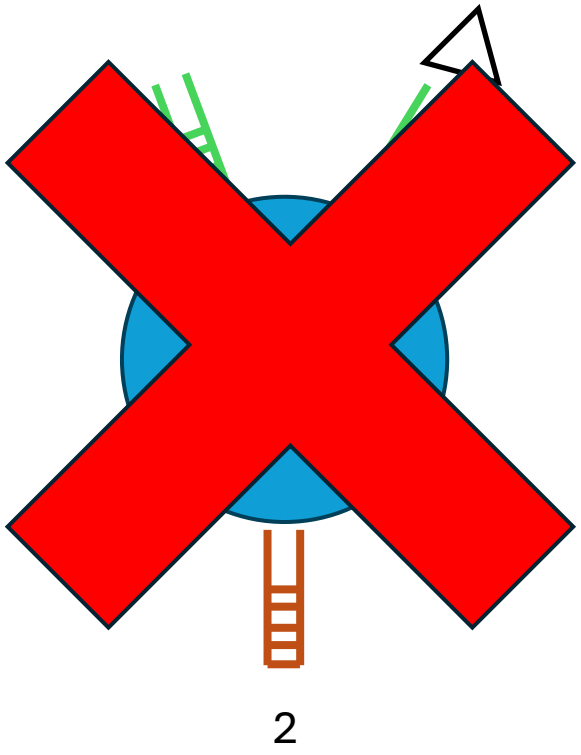
$$\begin{bmatrix}\triangle\\ \end{bmatrix} \begin{bmatrix}\\ \end{bmatrix} \mapsto [\,1\,]$$

$$\begin{bmatrix}\\ \end{bmatrix} \begin{bmatrix}\triangle\\ \end{bmatrix} \mapsto [\,2\,]$$

$$\begin{bmatrix}\triangle\\ \end{bmatrix} \begin{bmatrix}\triangle\\ \end{bmatrix} \mapsto \text{??}$$

More Abstraction?





$$\begin{bmatrix}\triangle\\ \ \end{bmatrix} \mapsto [\,1\,]$$

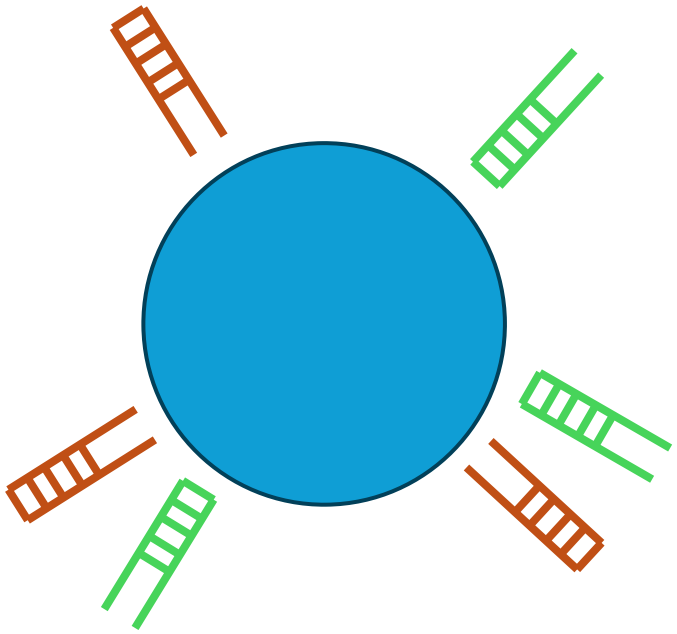
$$\begin{bmatrix}\ \\ \triangle\end{bmatrix} \mapsto [\,2\,]$$

$$\begin{bmatrix}\triangle\\ \triangle\end{bmatrix} \mapsto \text{??}$$

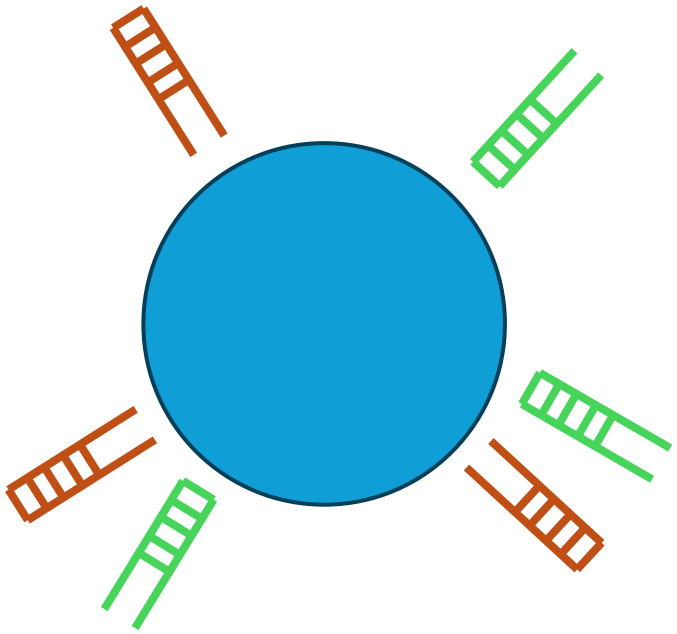
A solution: **Causal Systems**

The System Algebra

The System Algebra

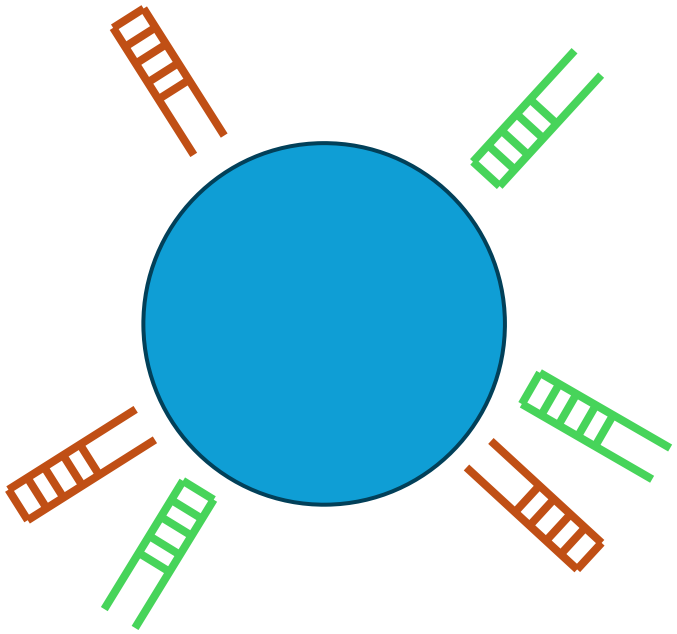


The System Algebra



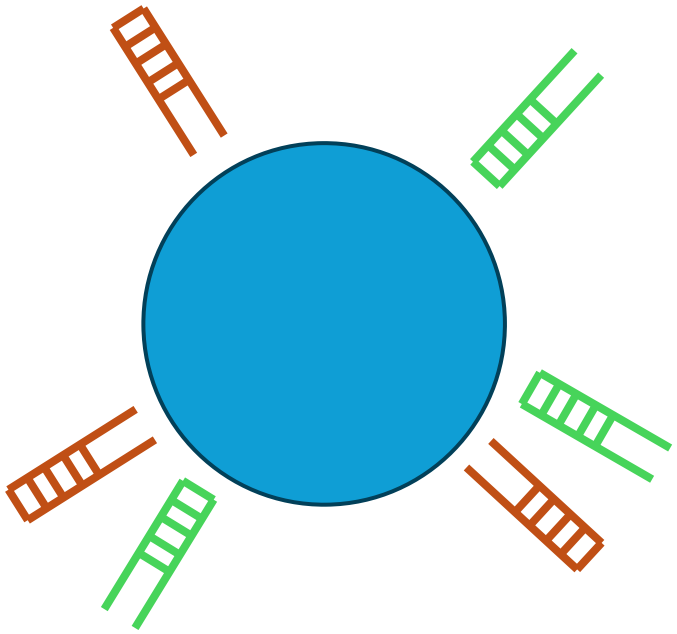
$S : Set$

The System Algebra



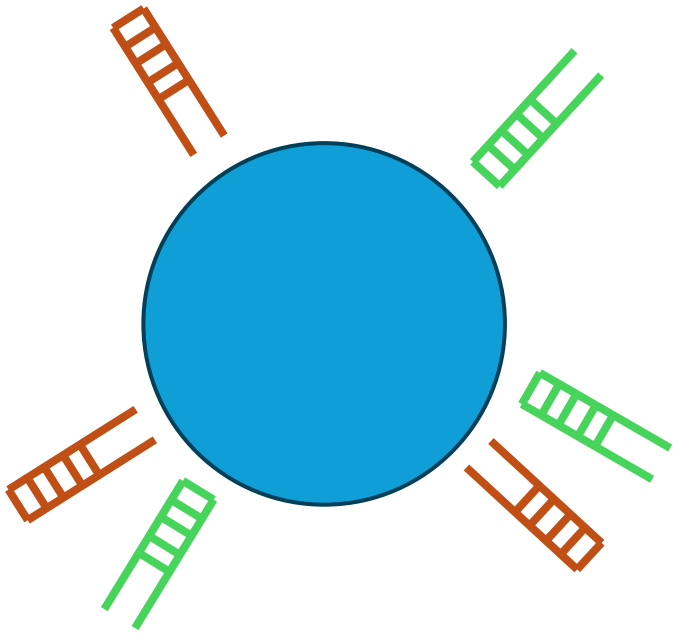
$$S : Set$$
$$\Lambda : FiniteSet$$

The System Algebra



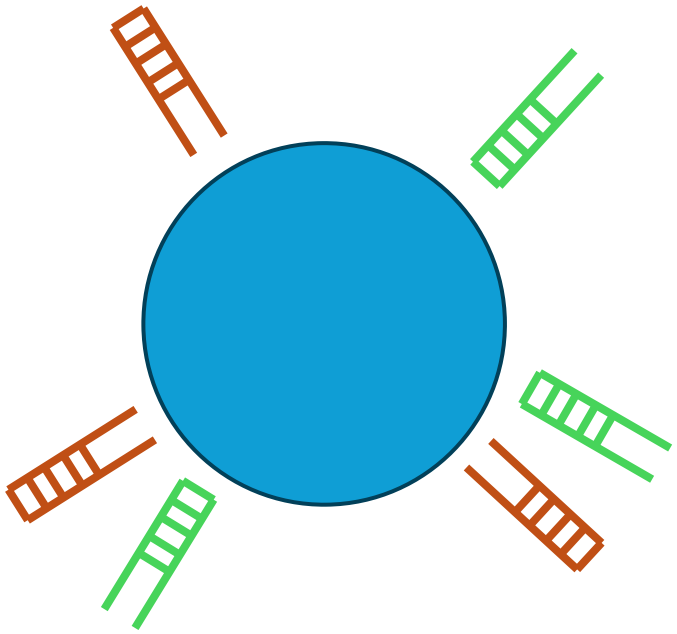
$$S : Set$$
$$\Lambda : FiniteSet$$
$$\lambda : S \rightarrow 2^{\Lambda}$$

The System Algebra



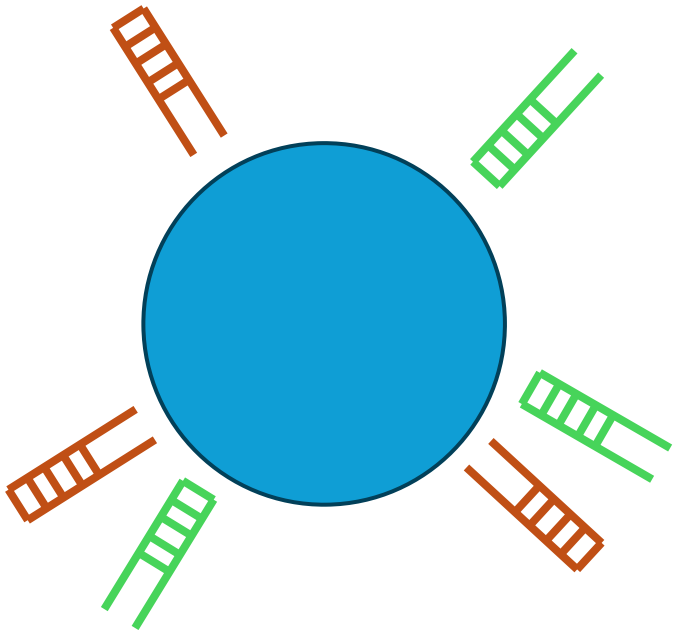
$$S : Set$$
$$\Lambda : FiniteSet$$
$$\lambda : S \to 2^{\Lambda}$$
$$||: S * S \rightharpoonup S$$

The System Algebra

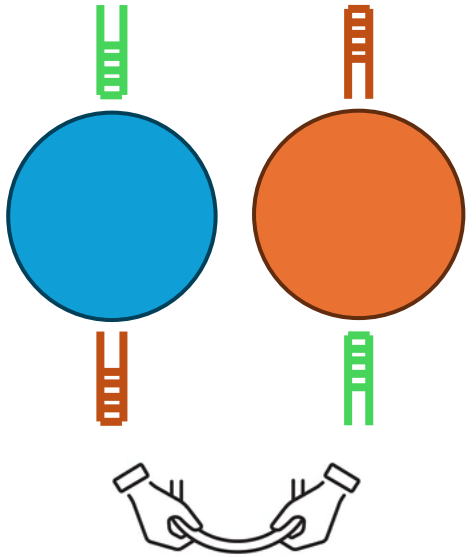


$$S : Set$$
$$\Lambda : FiniteSet$$
$$\lambda : S \to 2^{\Lambda}$$
$$||: S * S \rightharpoonup S$$
$$\gamma : S * \Lambda * \Lambda \rightharpoonup S$$

The System Algebra

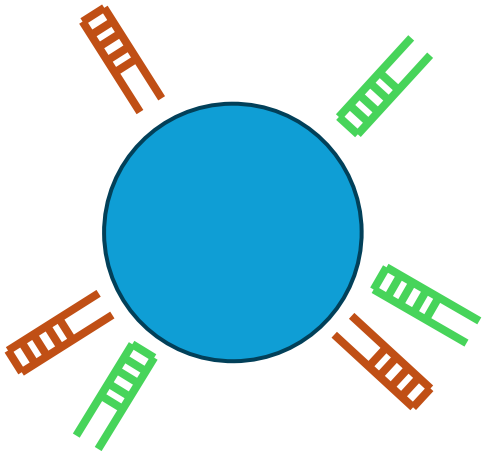


$$S : Set$$
$$\Lambda : FiniteSet$$
$$\lambda : S \to 2^{\Lambda}$$
$$||: S * S \rightharpoonup S$$
$$\gamma : S * \Lambda * \Lambda \rightharpoonup S$$

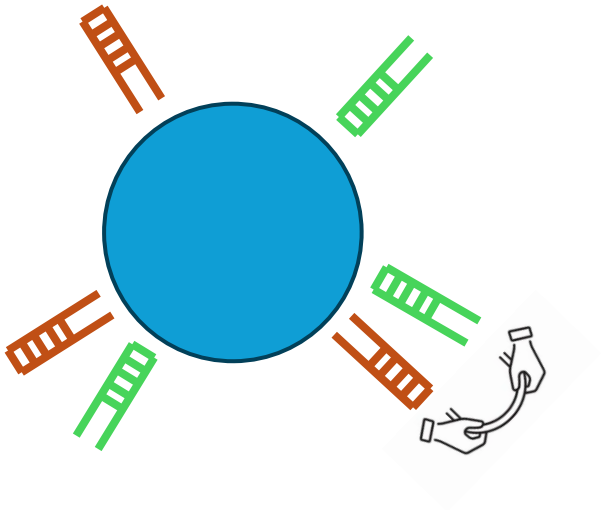


Laws?

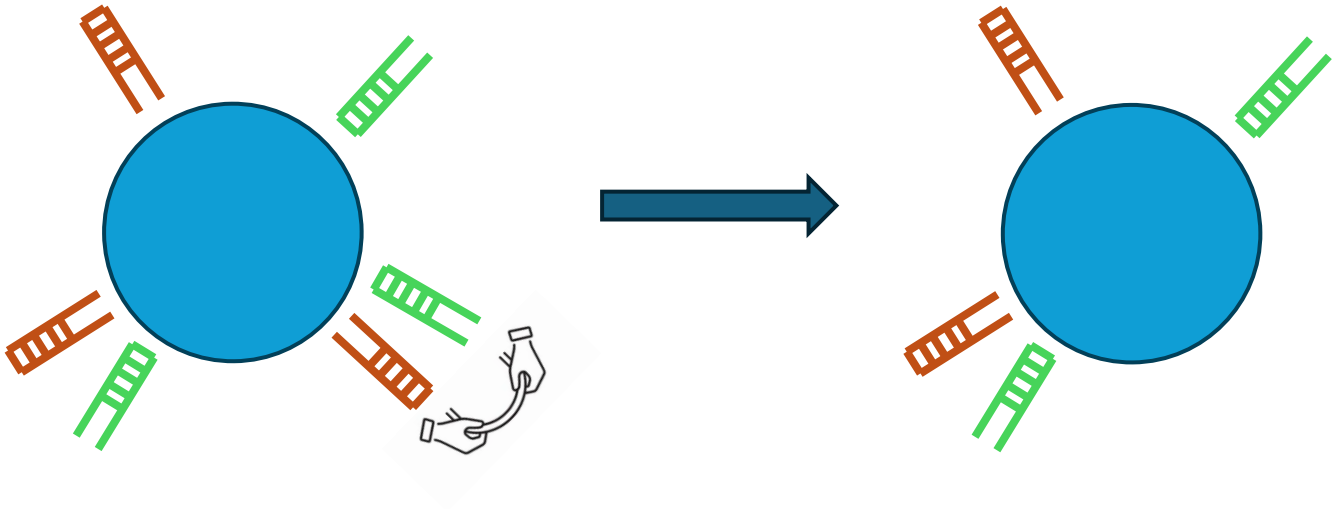
Laws?



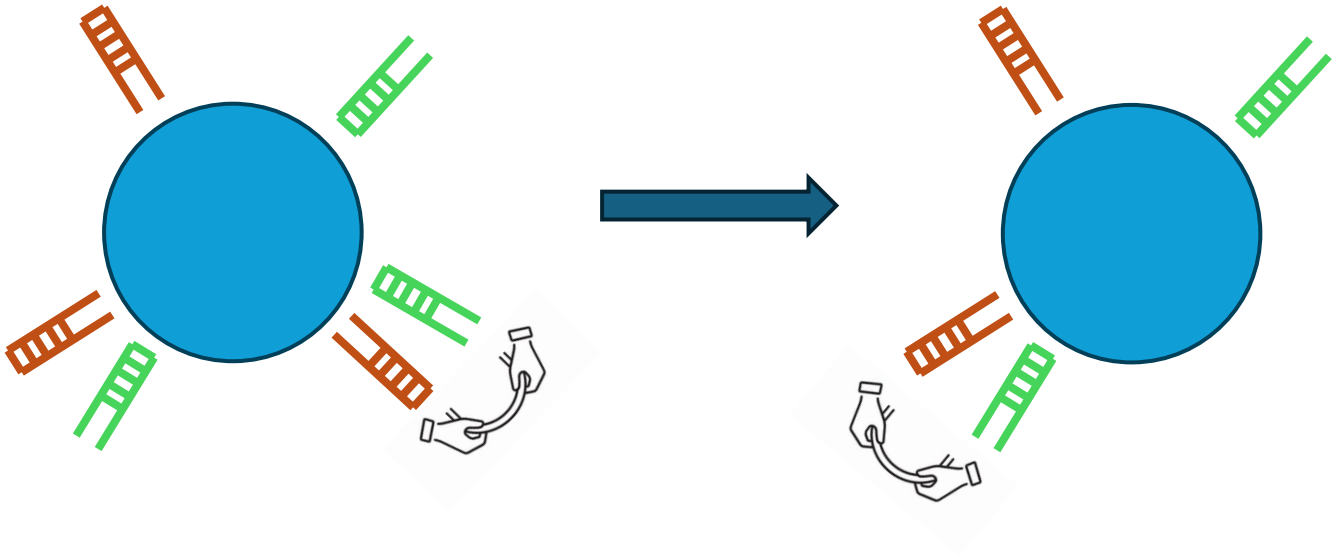
Laws?



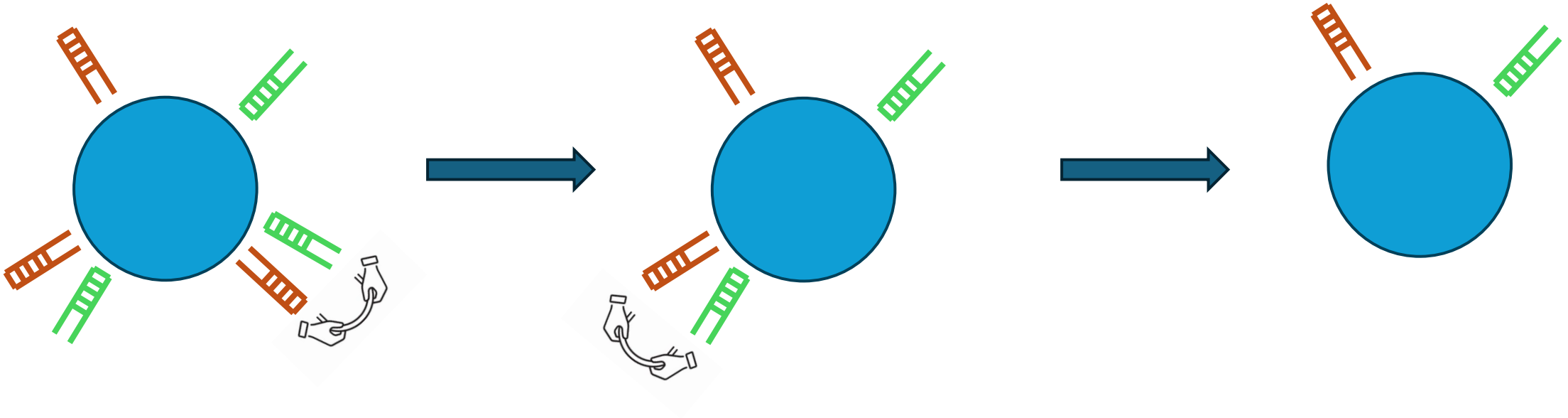
Laws?



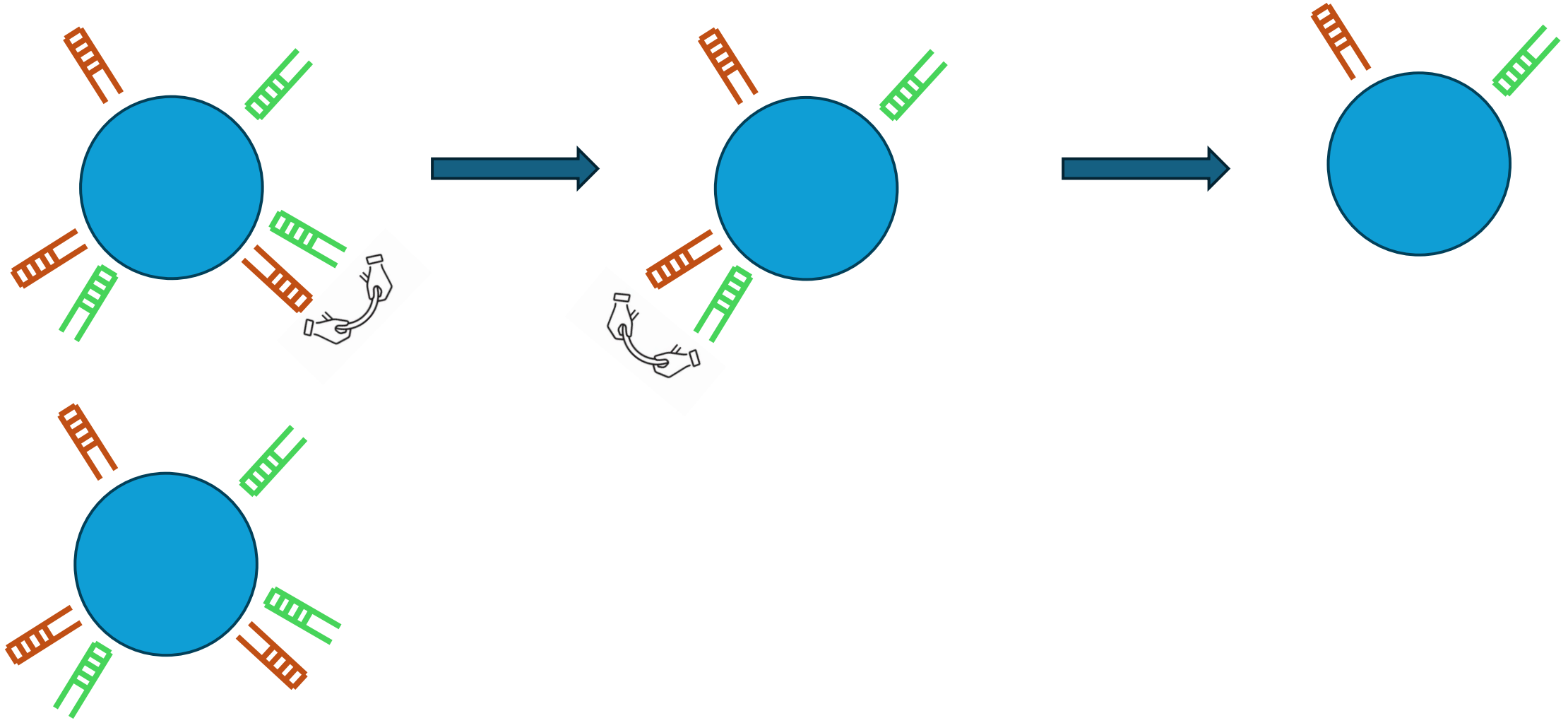
Laws?



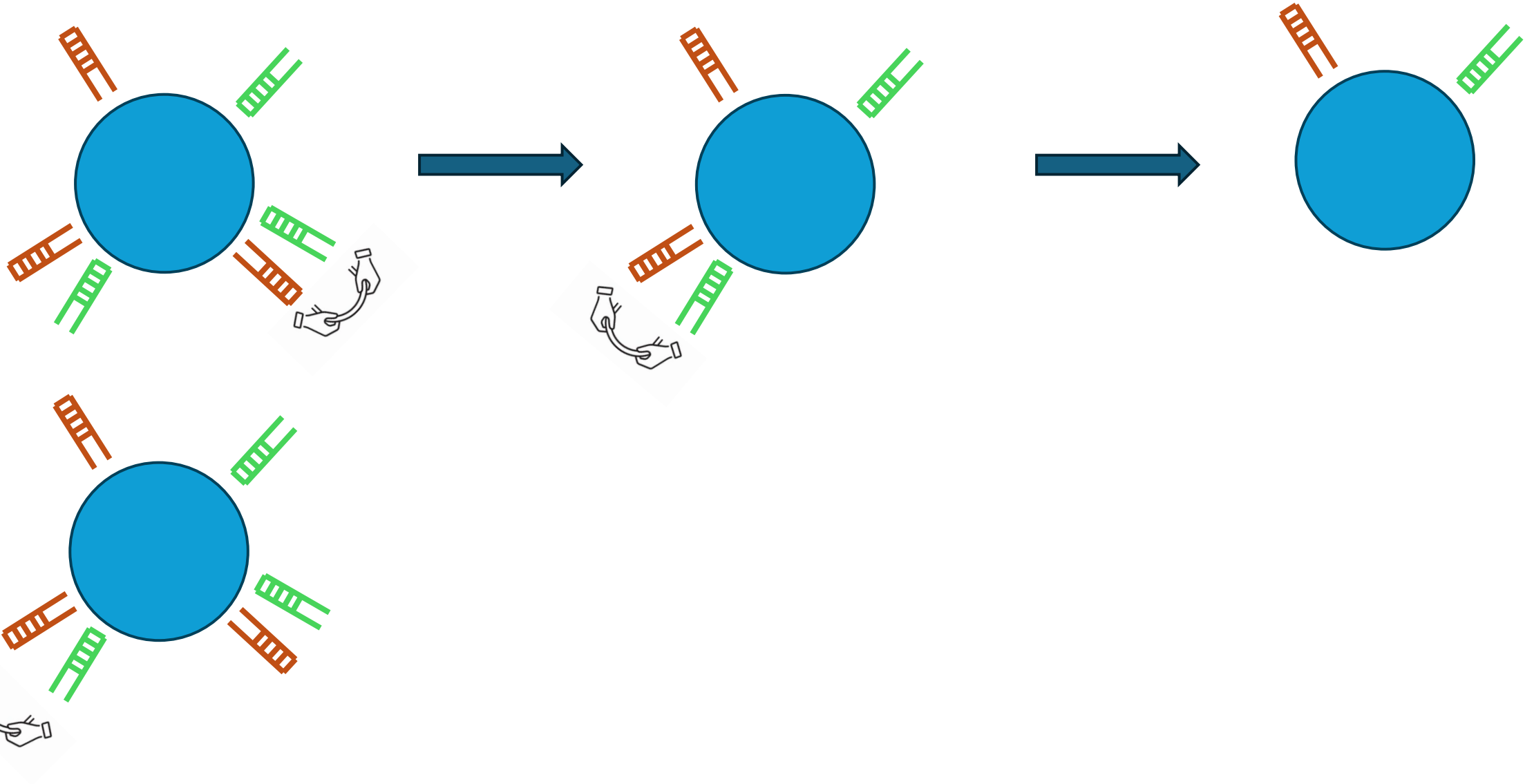
Laws?



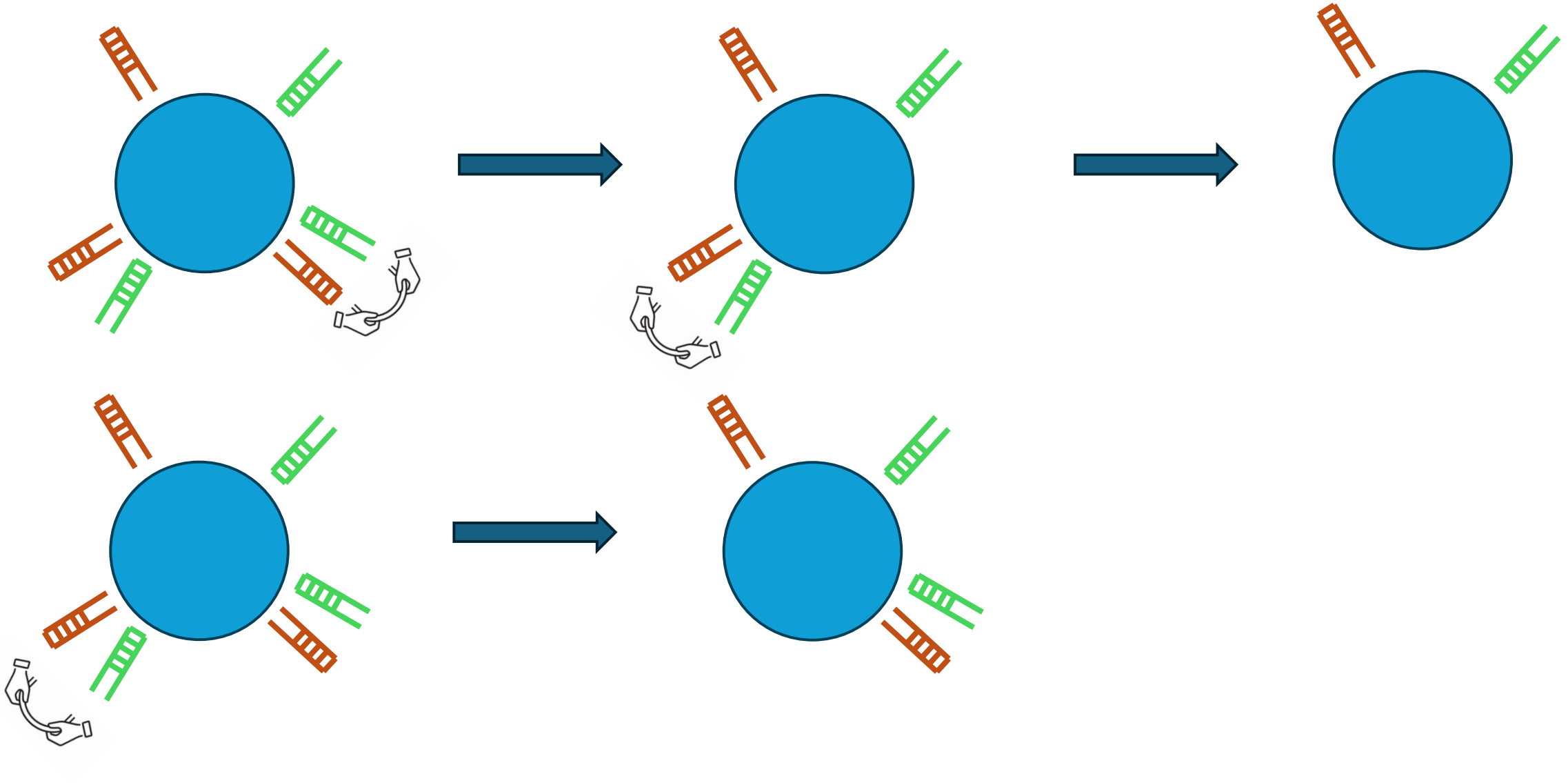
Laws?



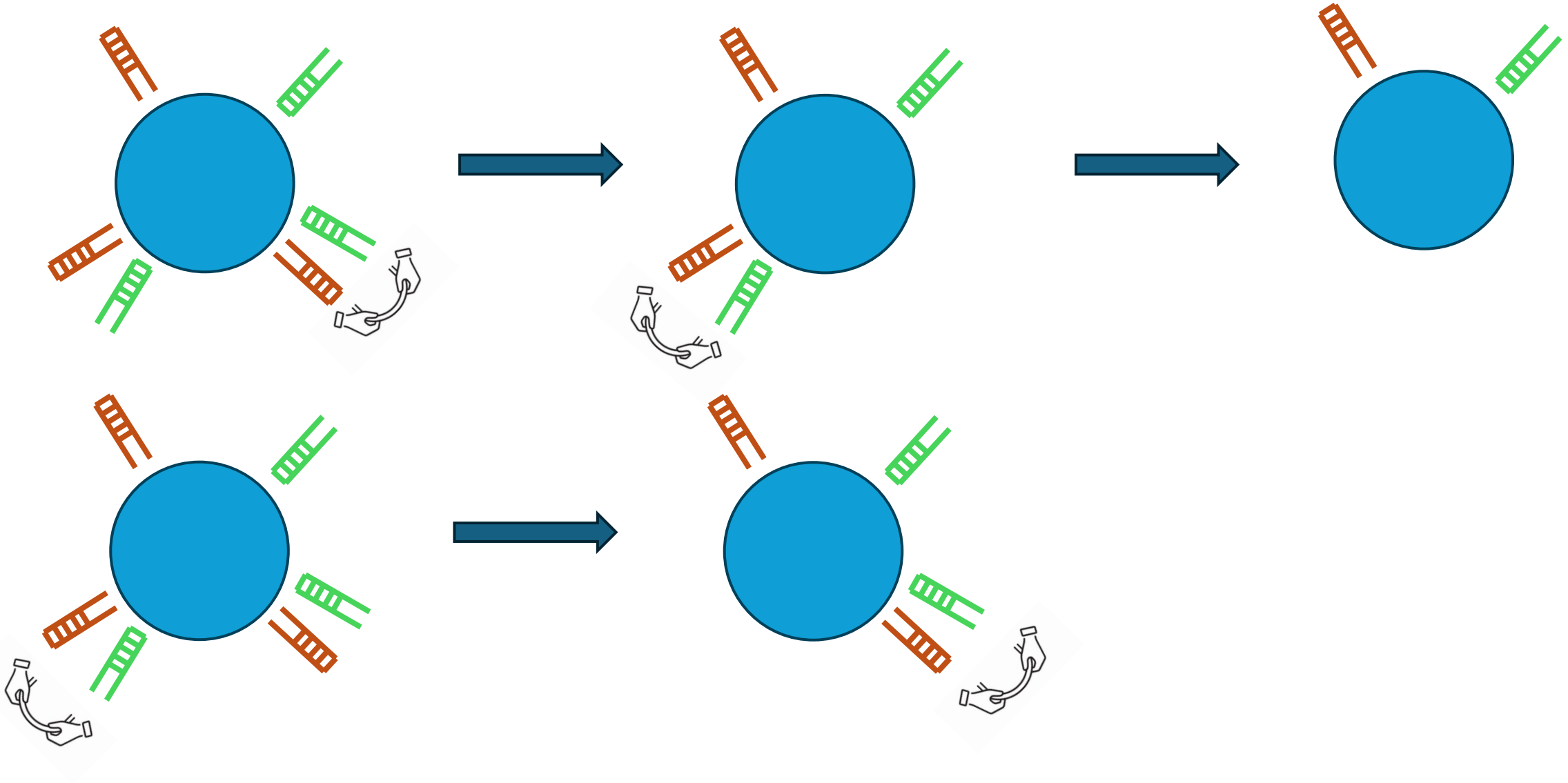
Laws?



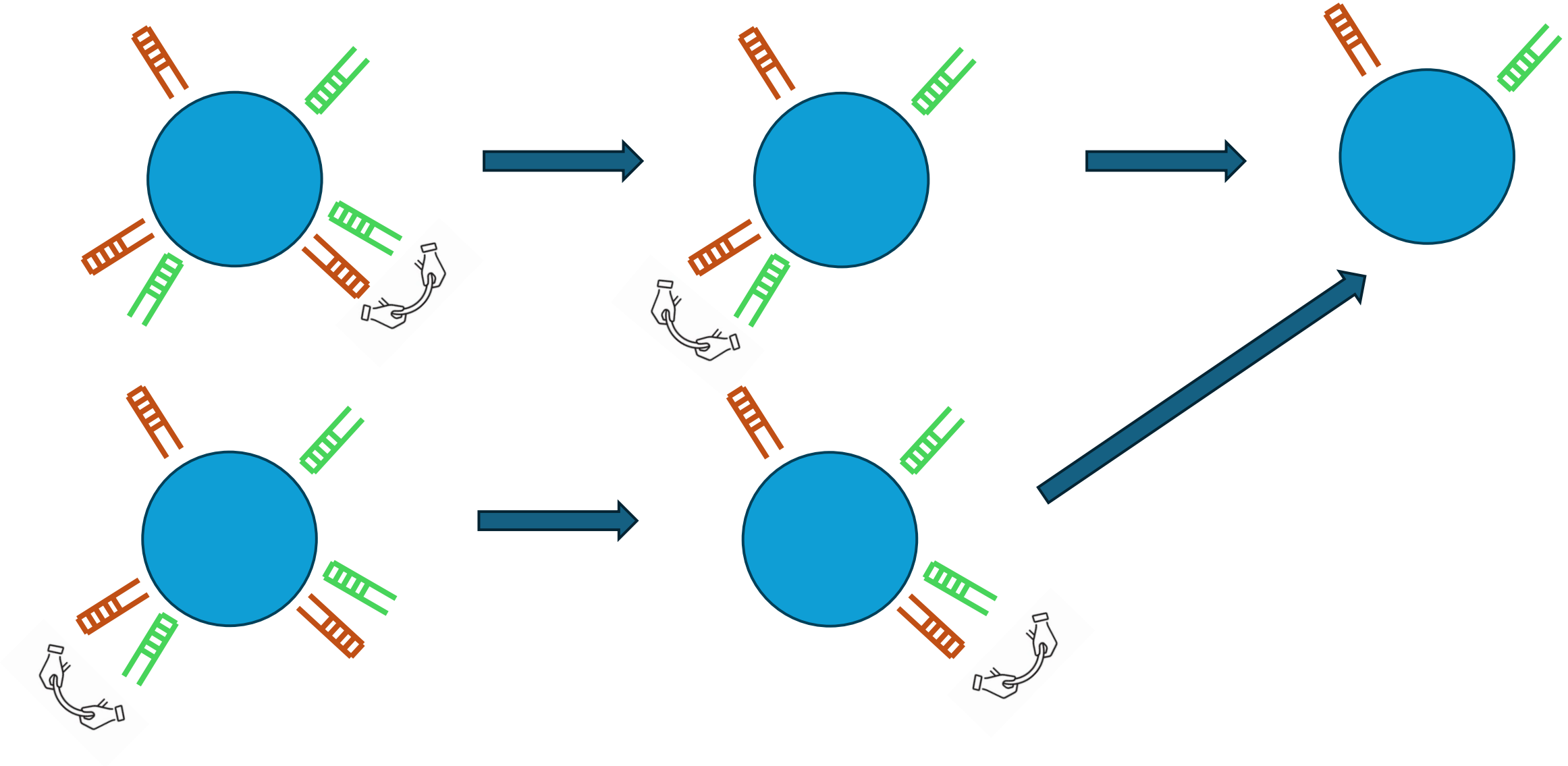
Laws?



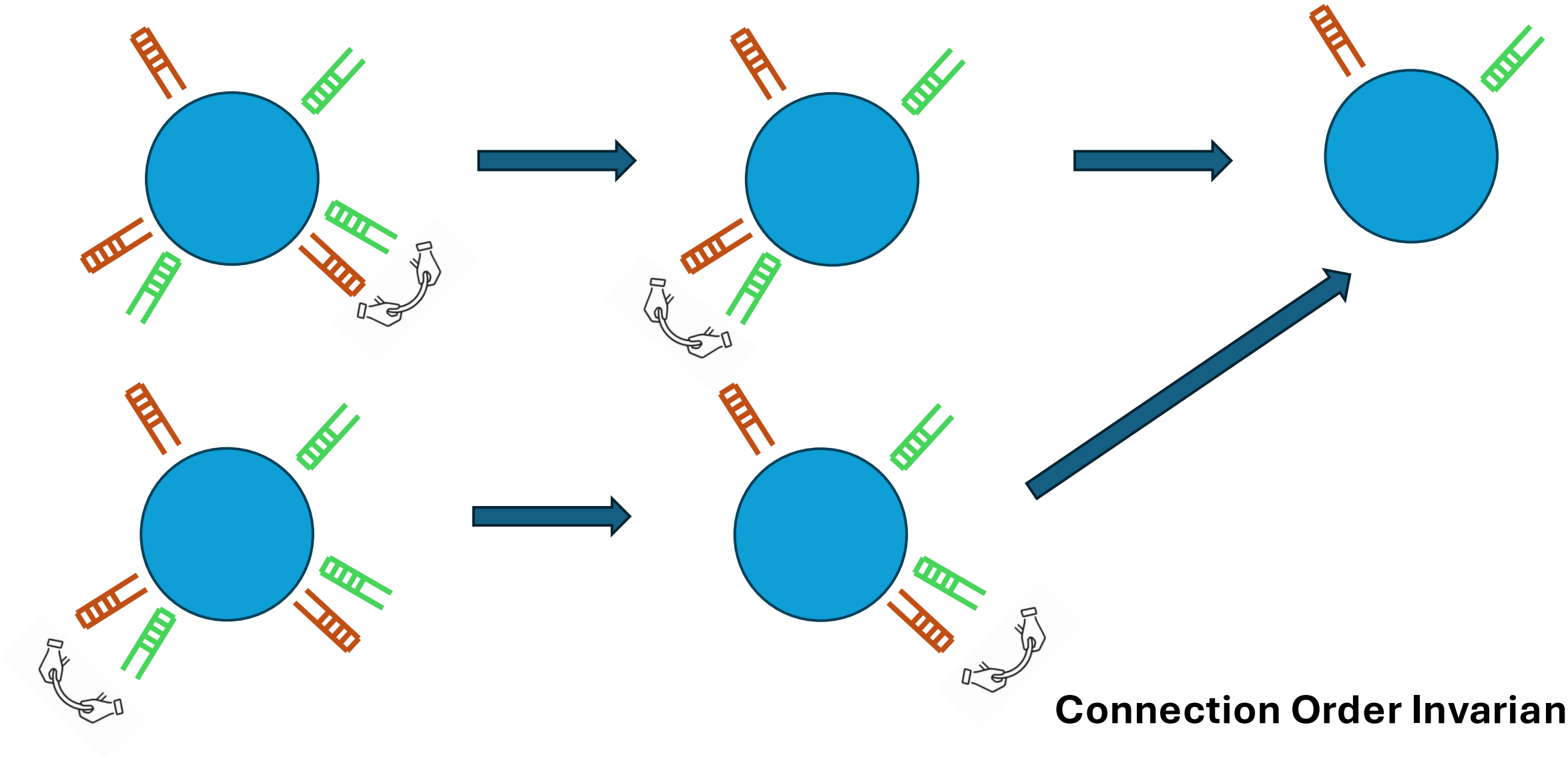
Laws?



Laws?



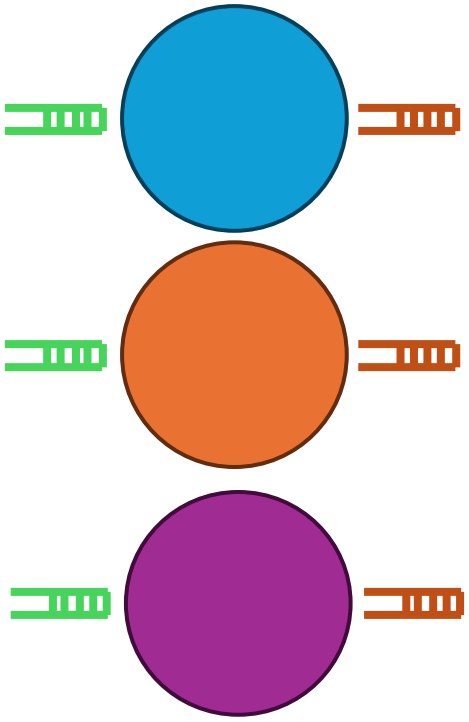
Laws?



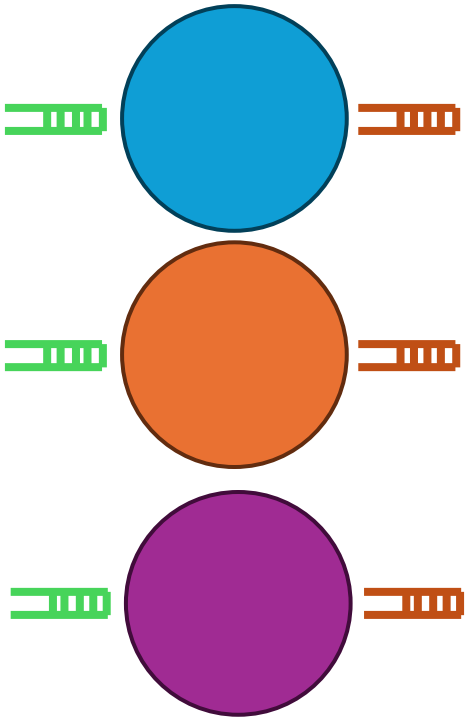
Connection Order Invariance

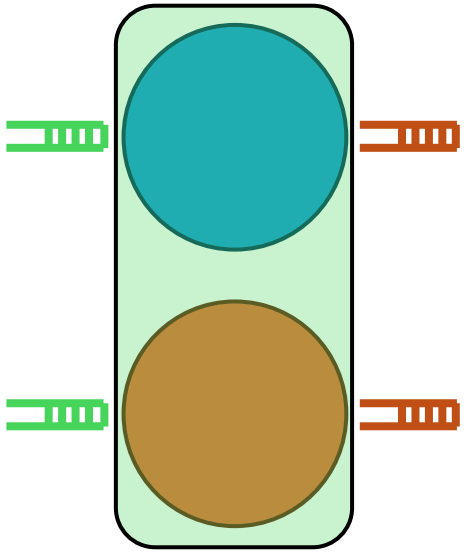
Laws?

Laws?

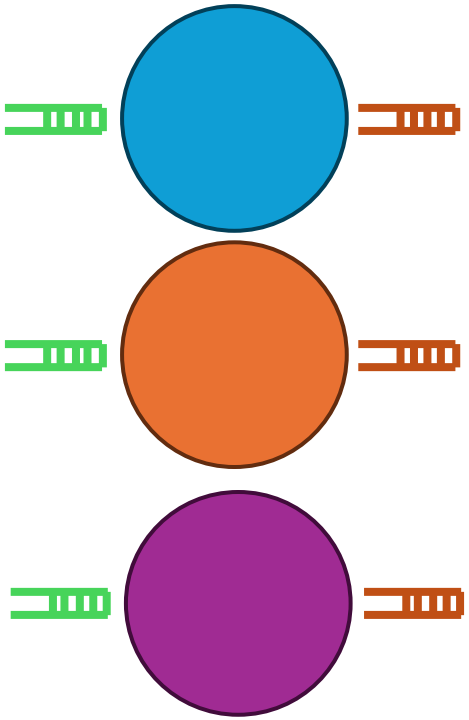


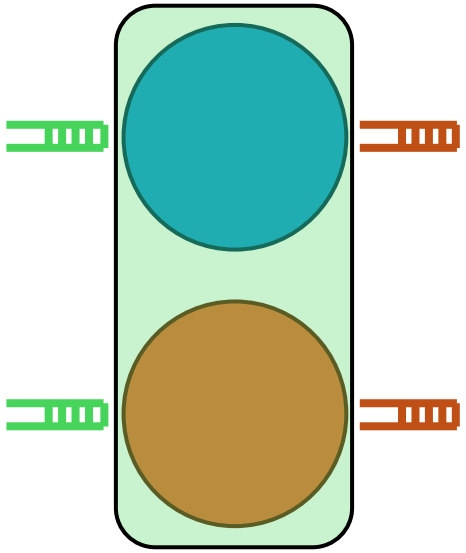
Laws?

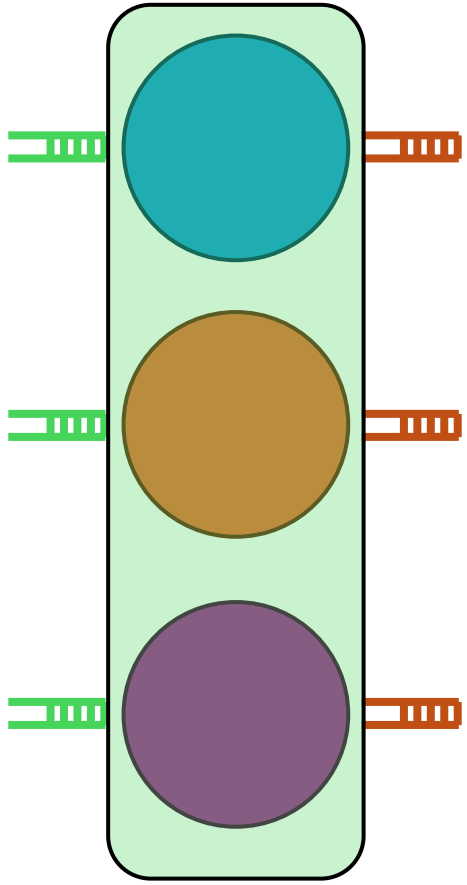




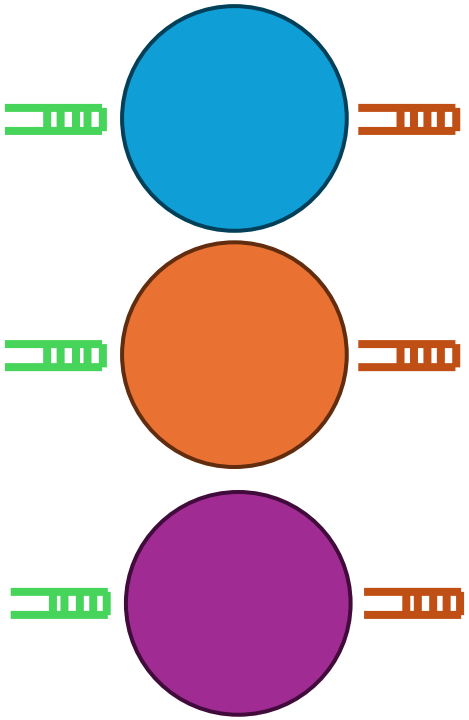
Laws?

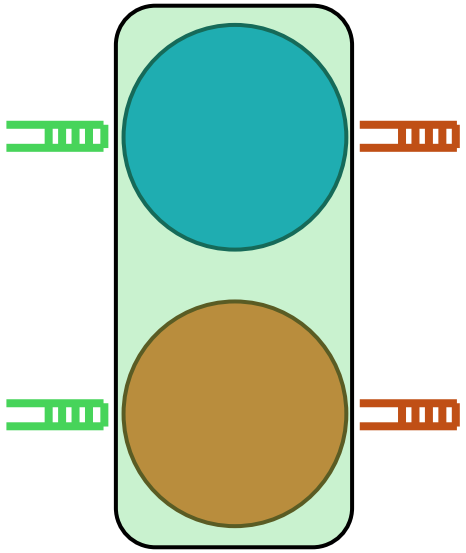


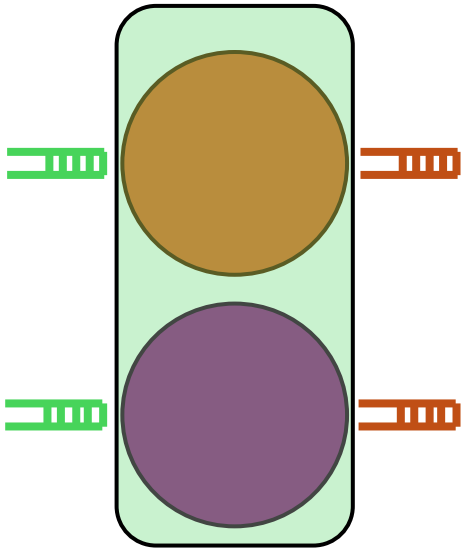


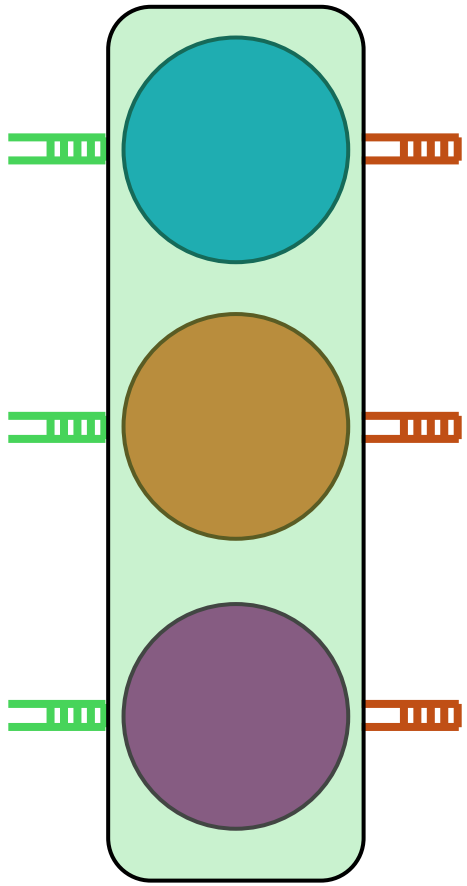


Laws?

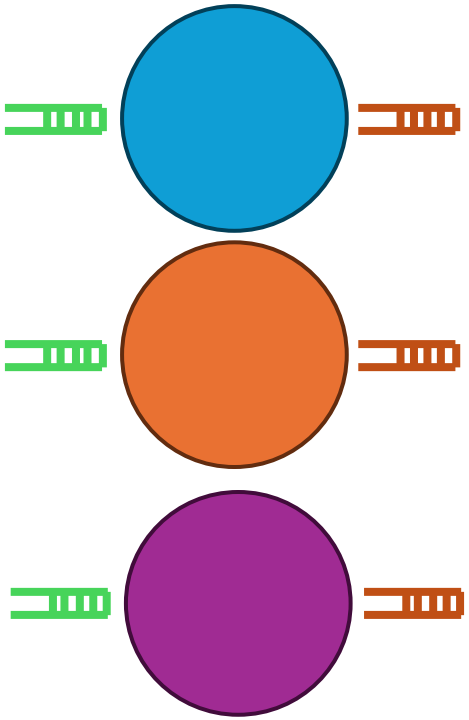


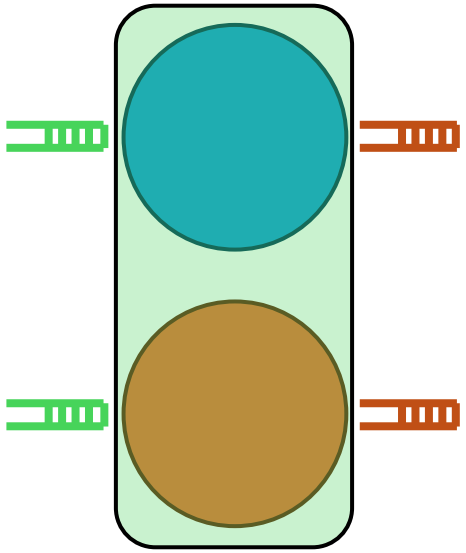


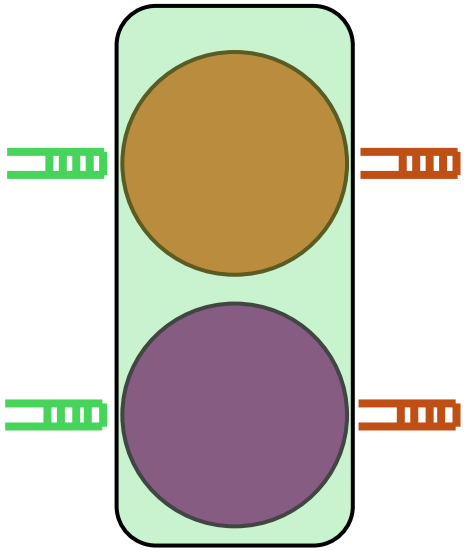


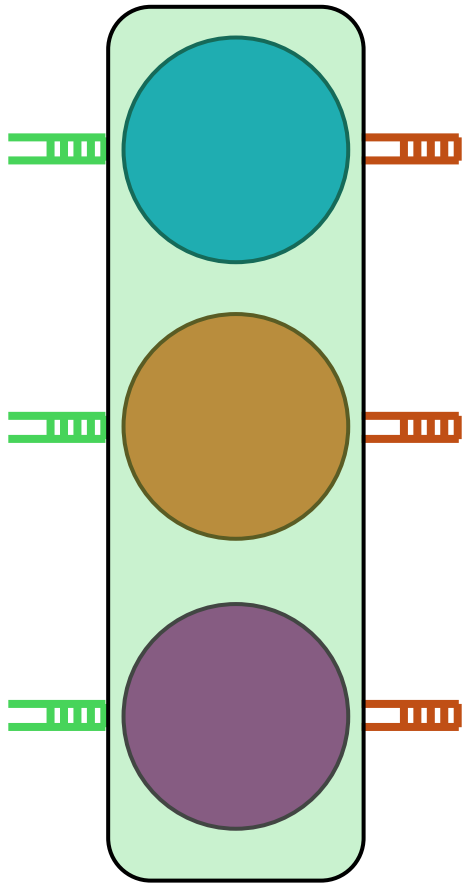


Laws?





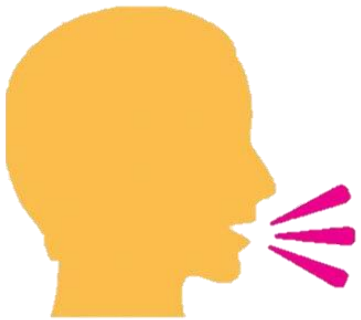




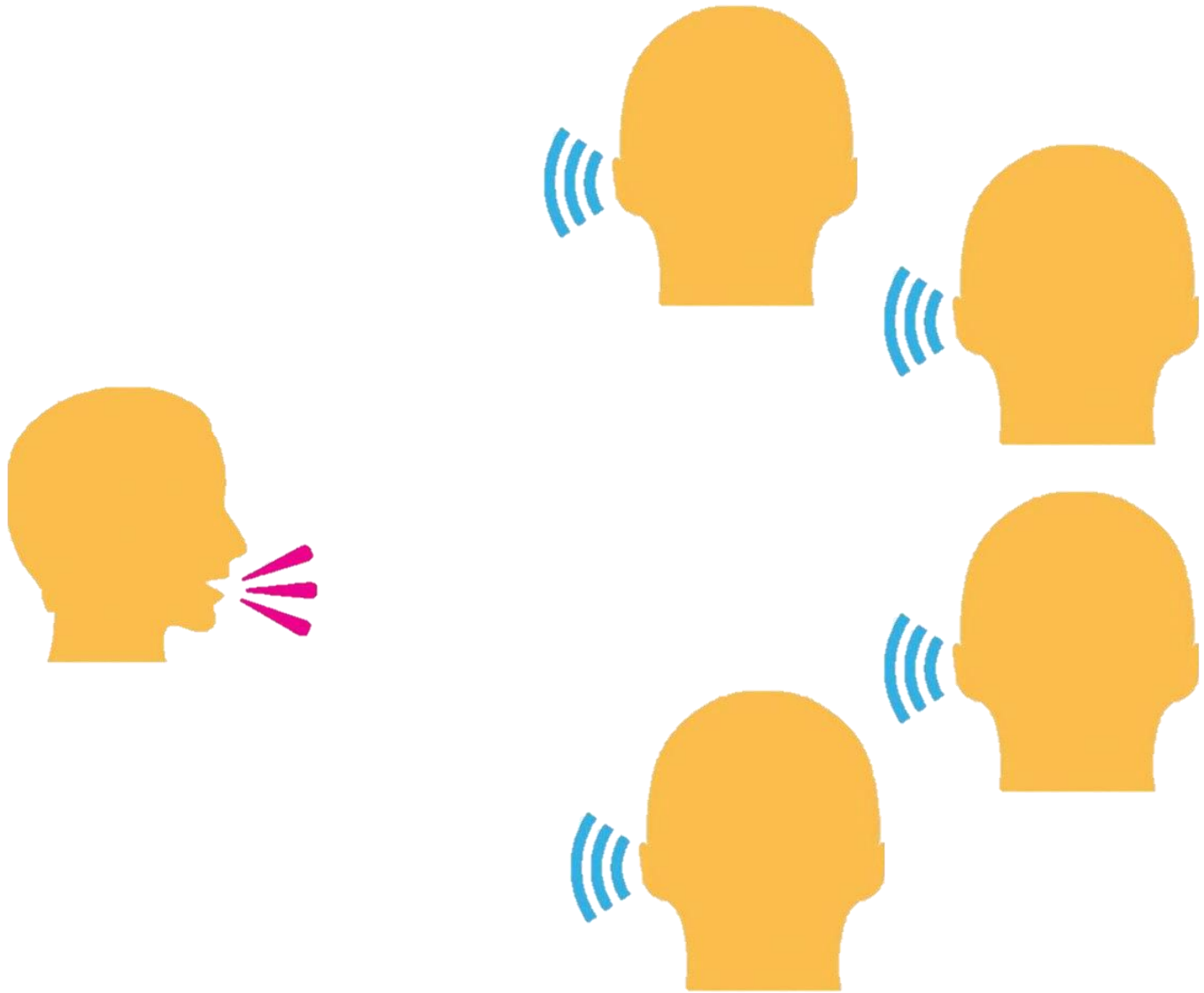
Composition Order Invariance

Byzantine Broadcast

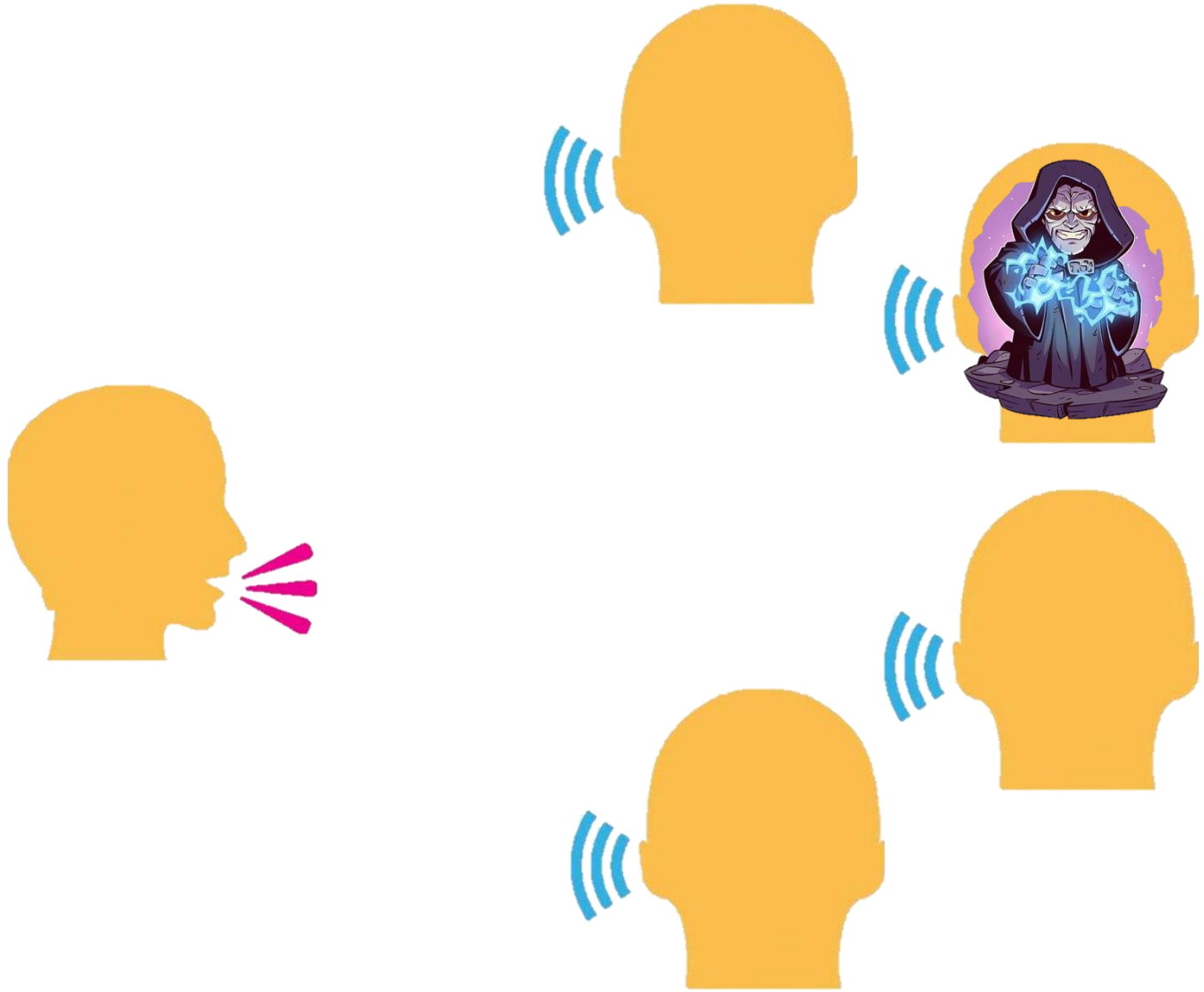
Byzantine Broadcast



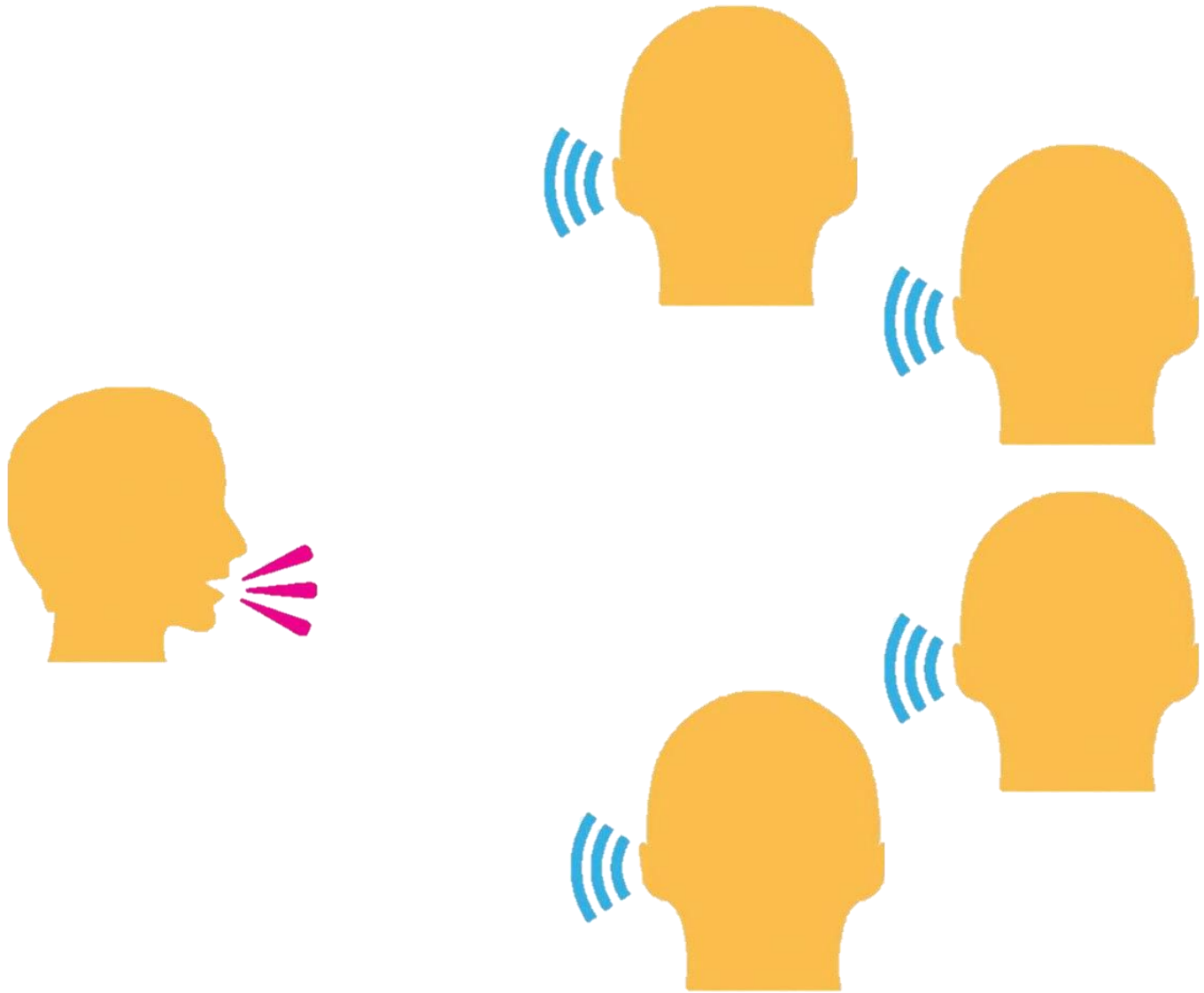
Byzantine Broadcast



Byzantine Broadcast

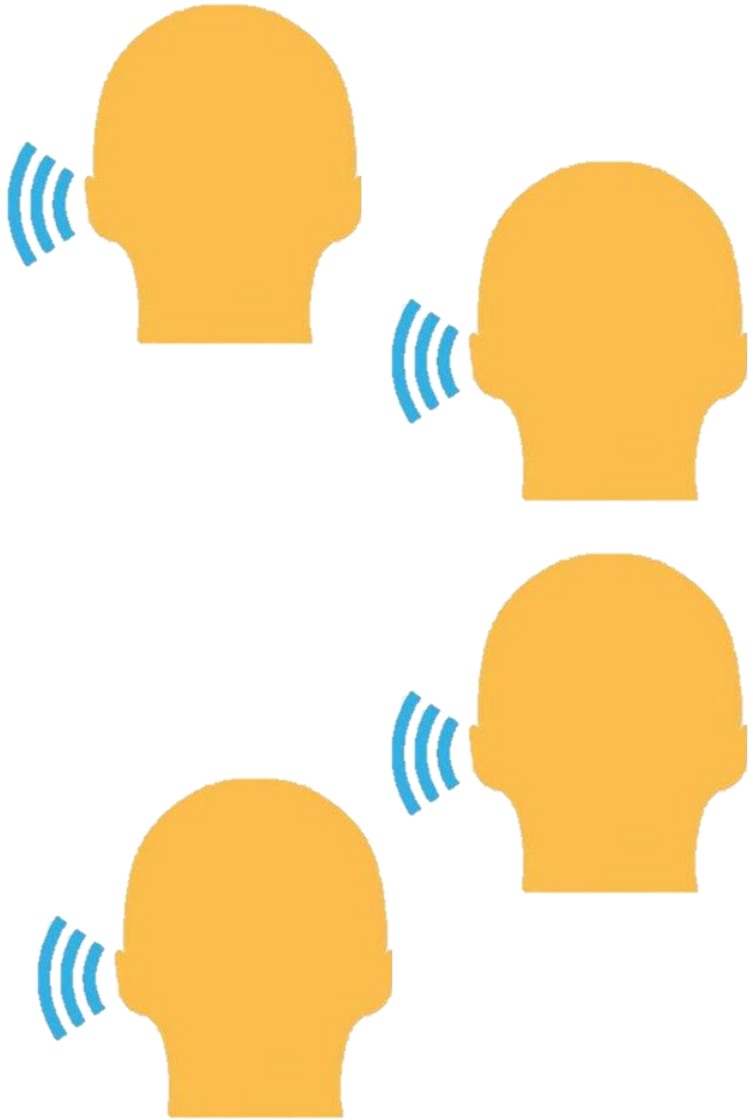


Byzantine Broadcast



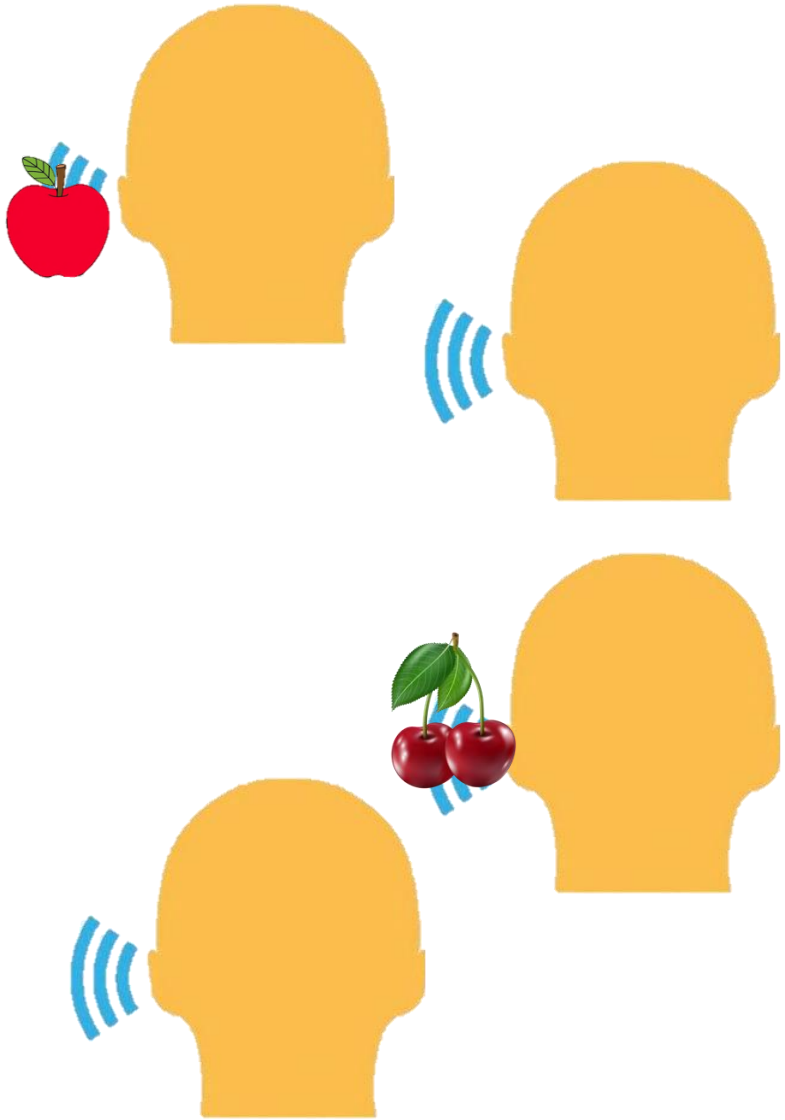
Byzantine Broadcast





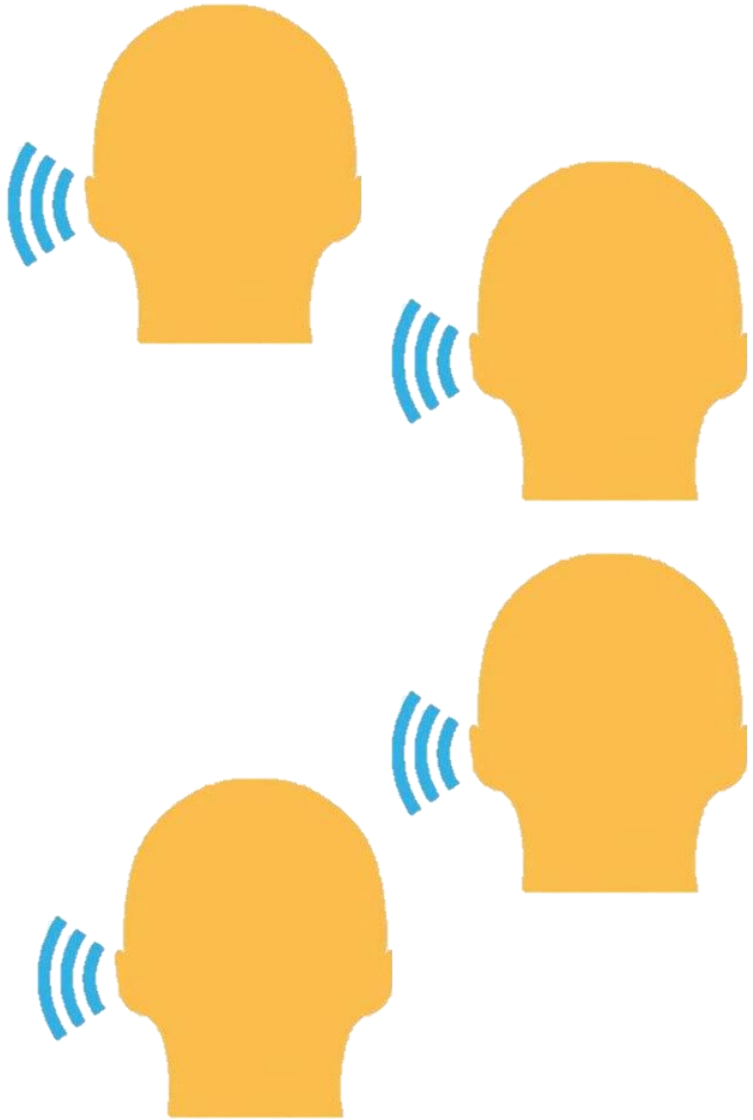
Byzantine Broadcast





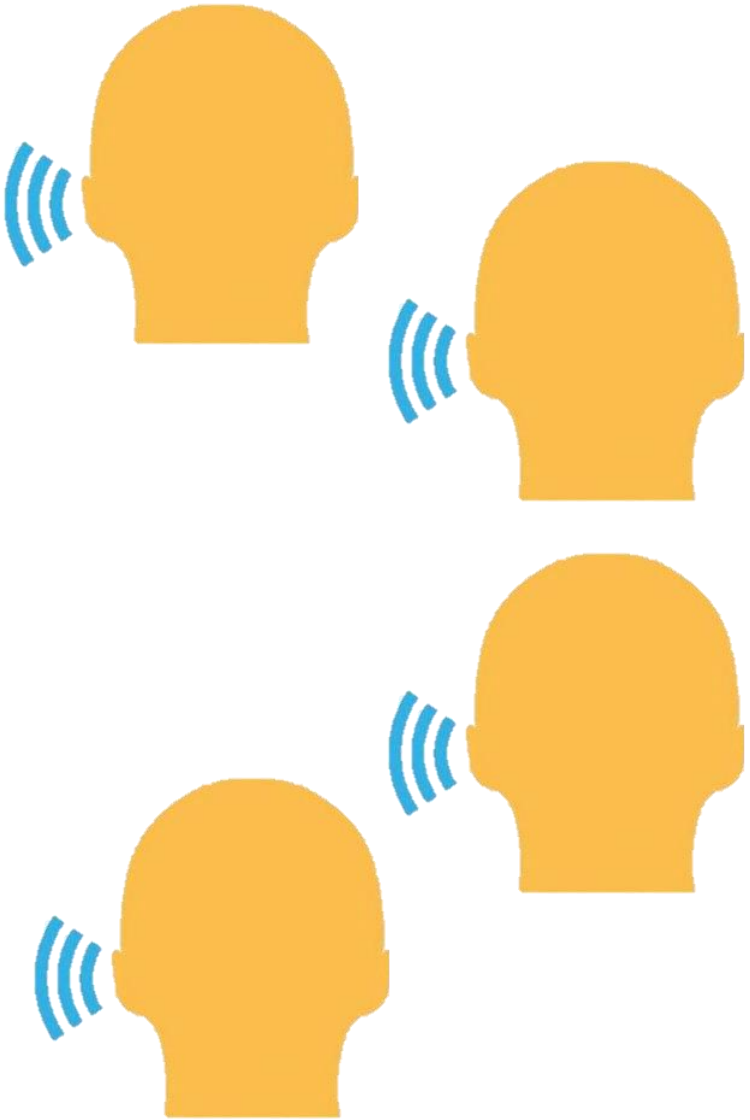
Byzantine Broadcast





Byzantine Broadcast

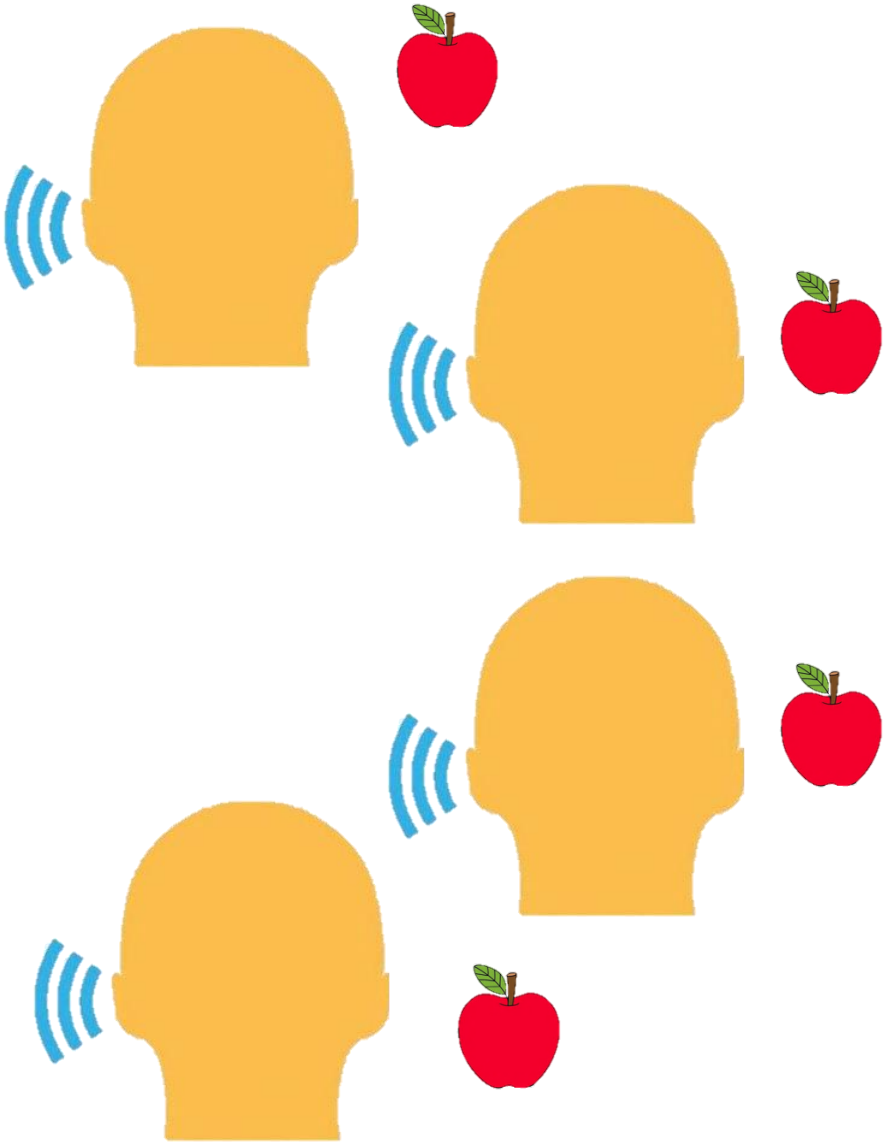




Agreement

Byzantine Broadcast

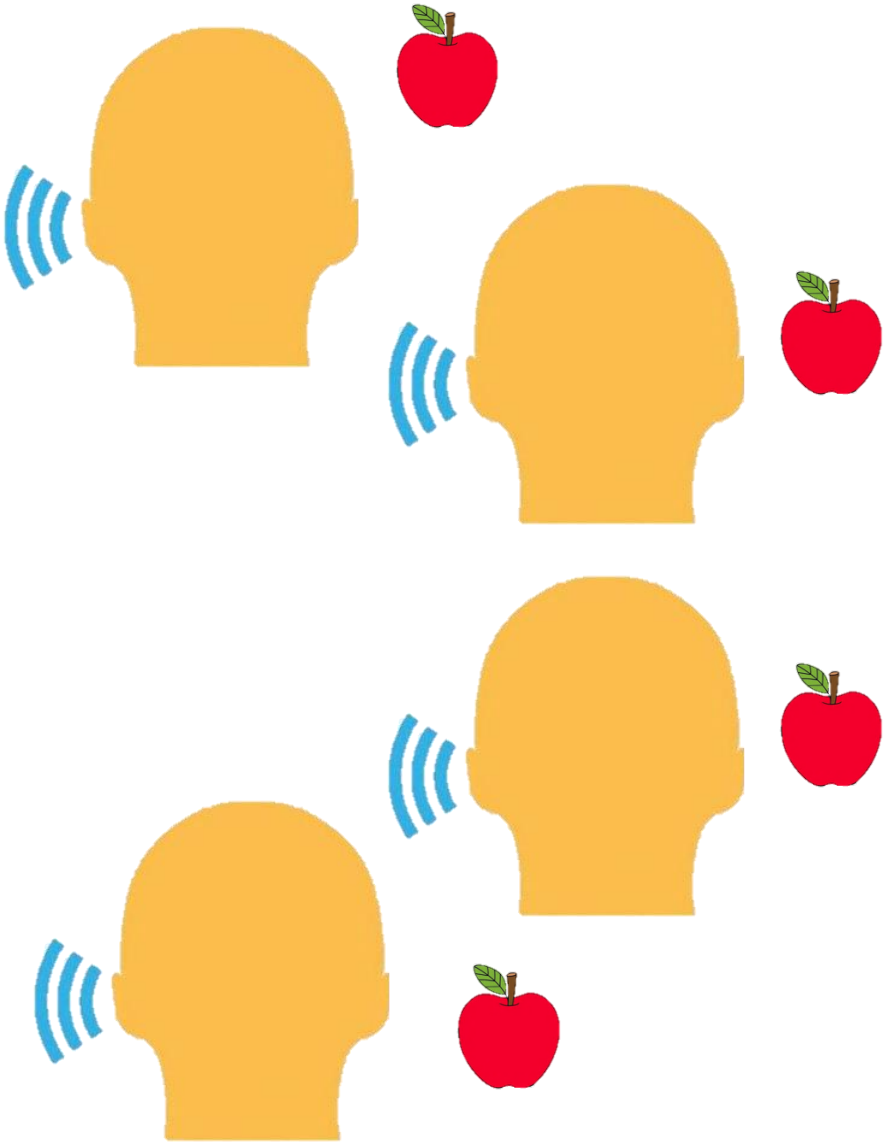




Agreement

Byzantine Broadcast

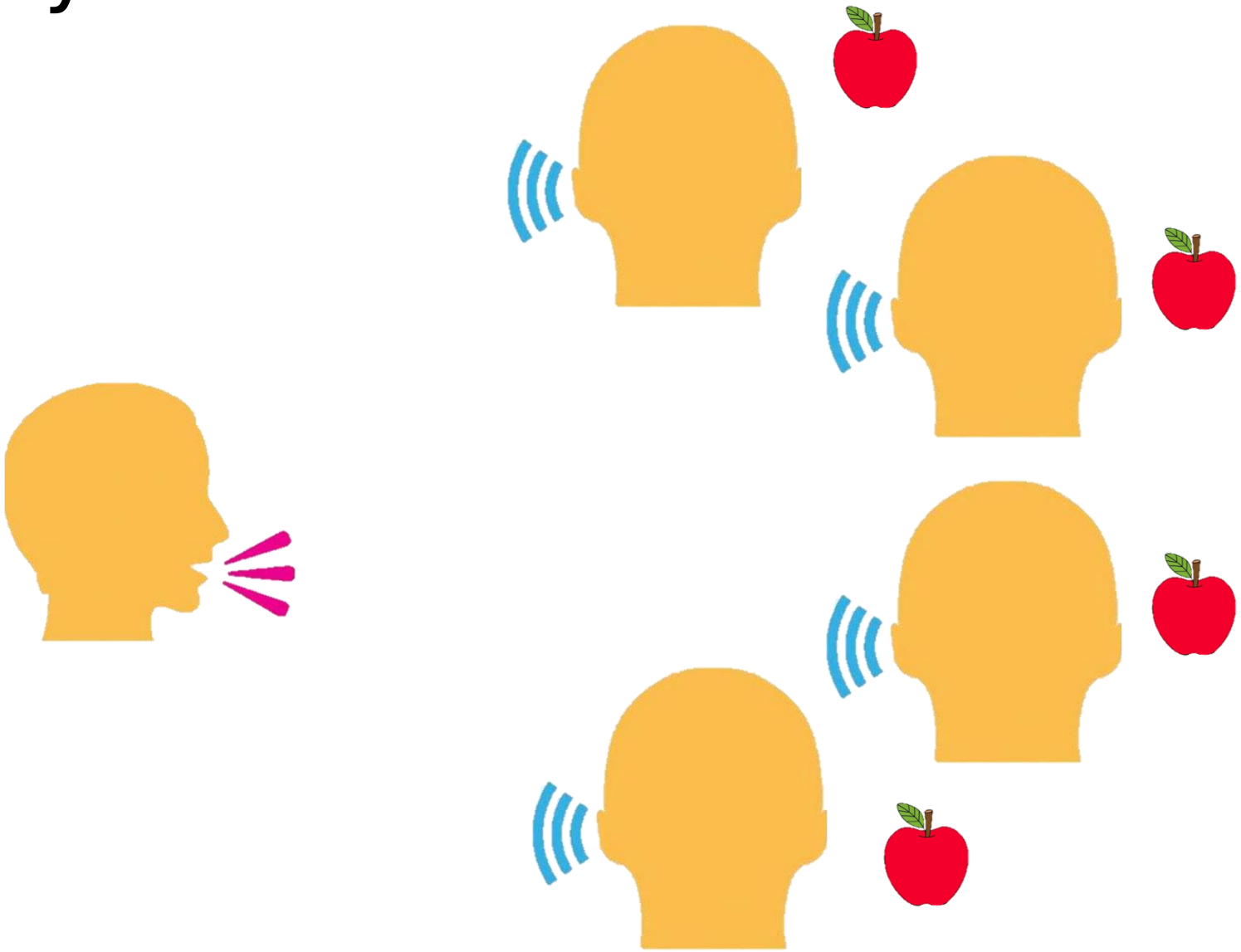




Agreement

Validity

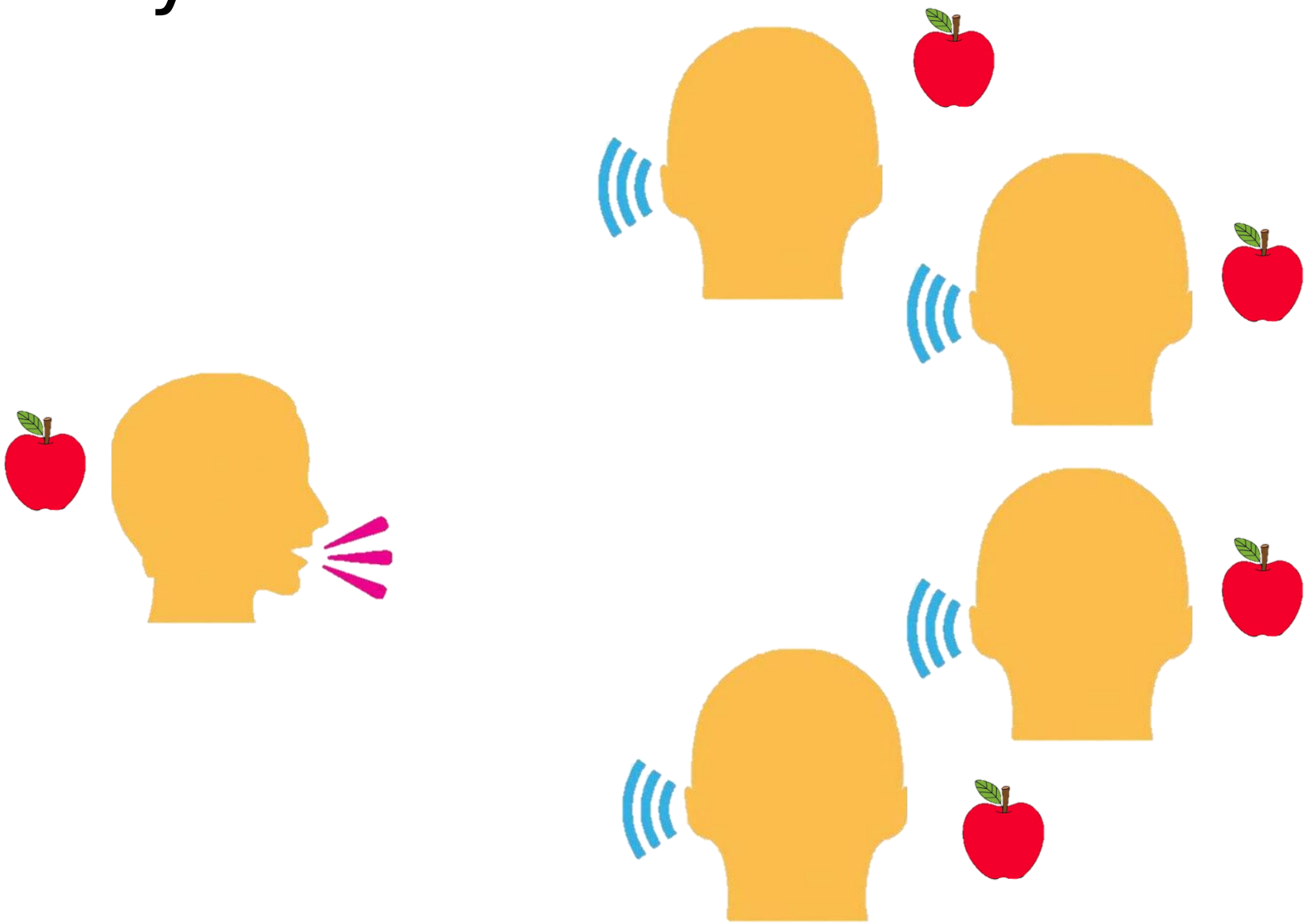
Byzantine Broadcast



Agreement

Validity

Byzantine Broadcast

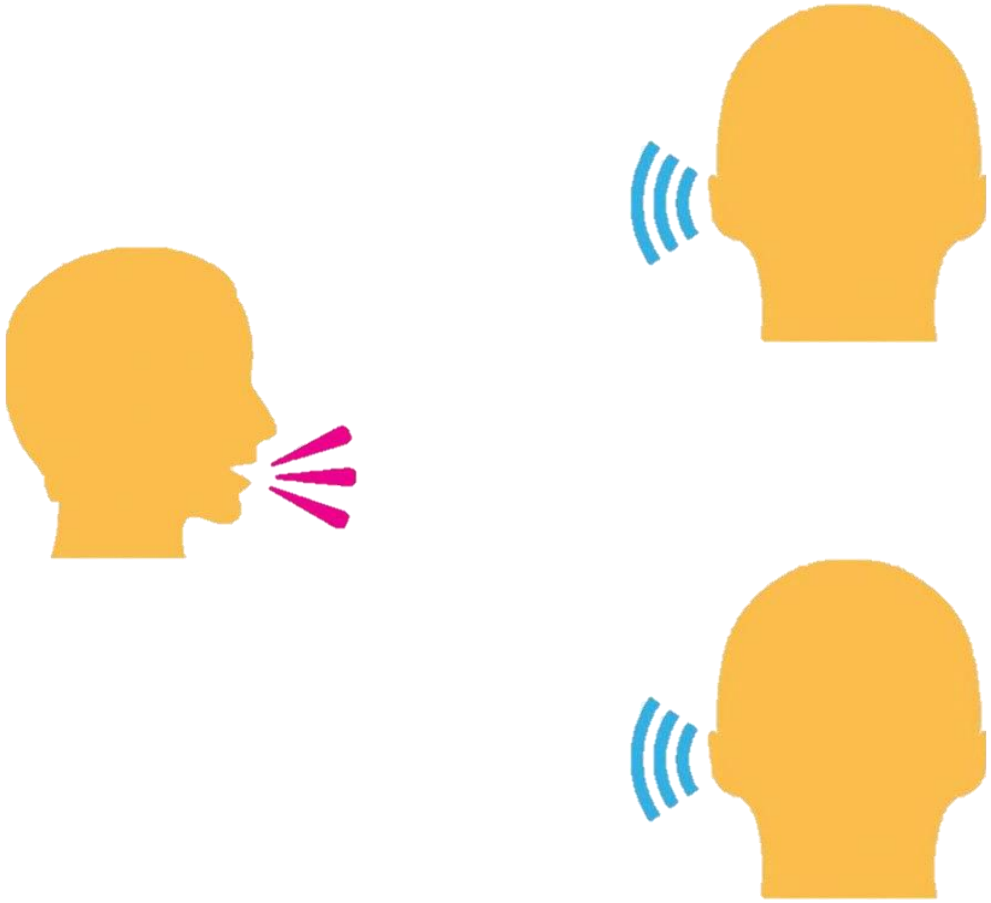


Agreement

Validity

Impossibility

Impossibility



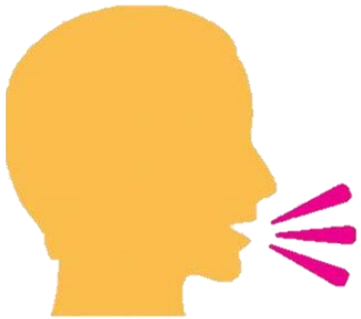
Impossibility







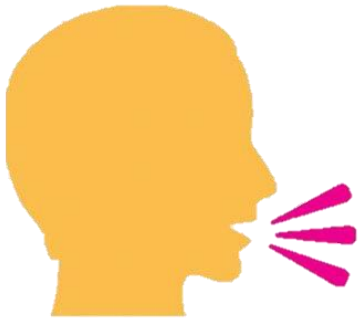
Impossibility







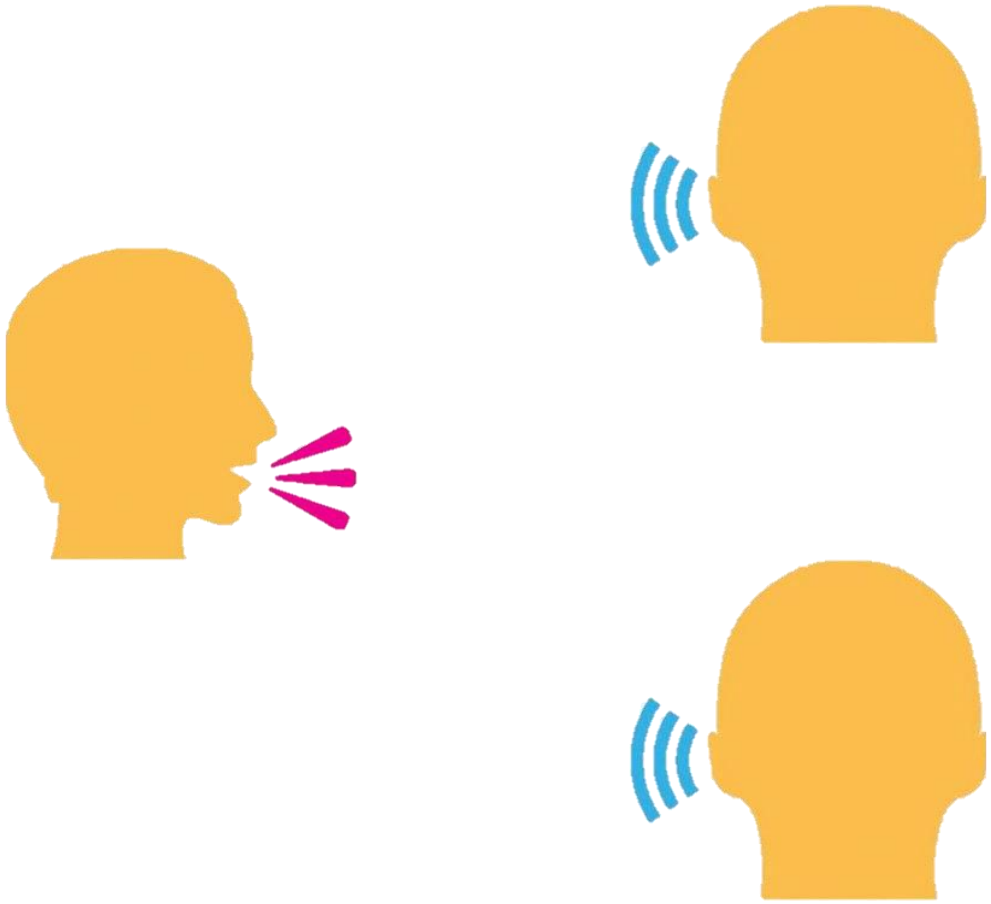
Impossibility







Impossibility



The Byzantine Generals Problem

LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International

Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach agreement. It is shown that, using only oral messages, this problem is solvable if and only if more than two-thirds of the generals are loyal; so a single traitor can confound two loyal generals. With unforgeable written messages, the problem is solvable for any number of generals and possible traitors. Applications of the solutions to reliable computer systems are then discussed.

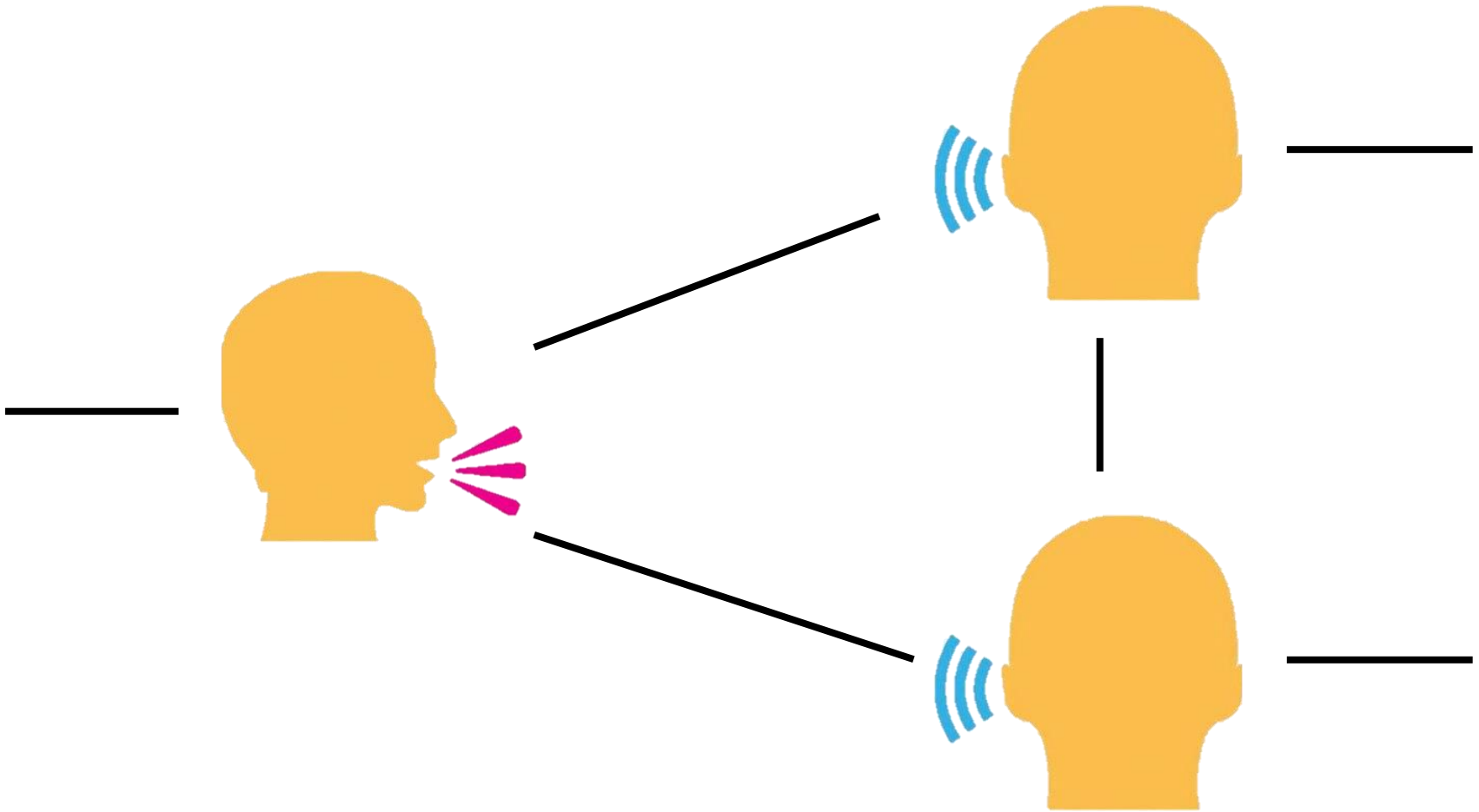
Categories and Subject Descriptors: C.2.4. [**Computer-Communication Networks**]: Distributed Systems—*network operating systems*; D.4.4 [**Operating Systems**]: Communications Management—*network communication*; D.4.5 [**Operating Systems**]: Reliability—*fault tolerance*

General Terms: Algorithms, Reliability

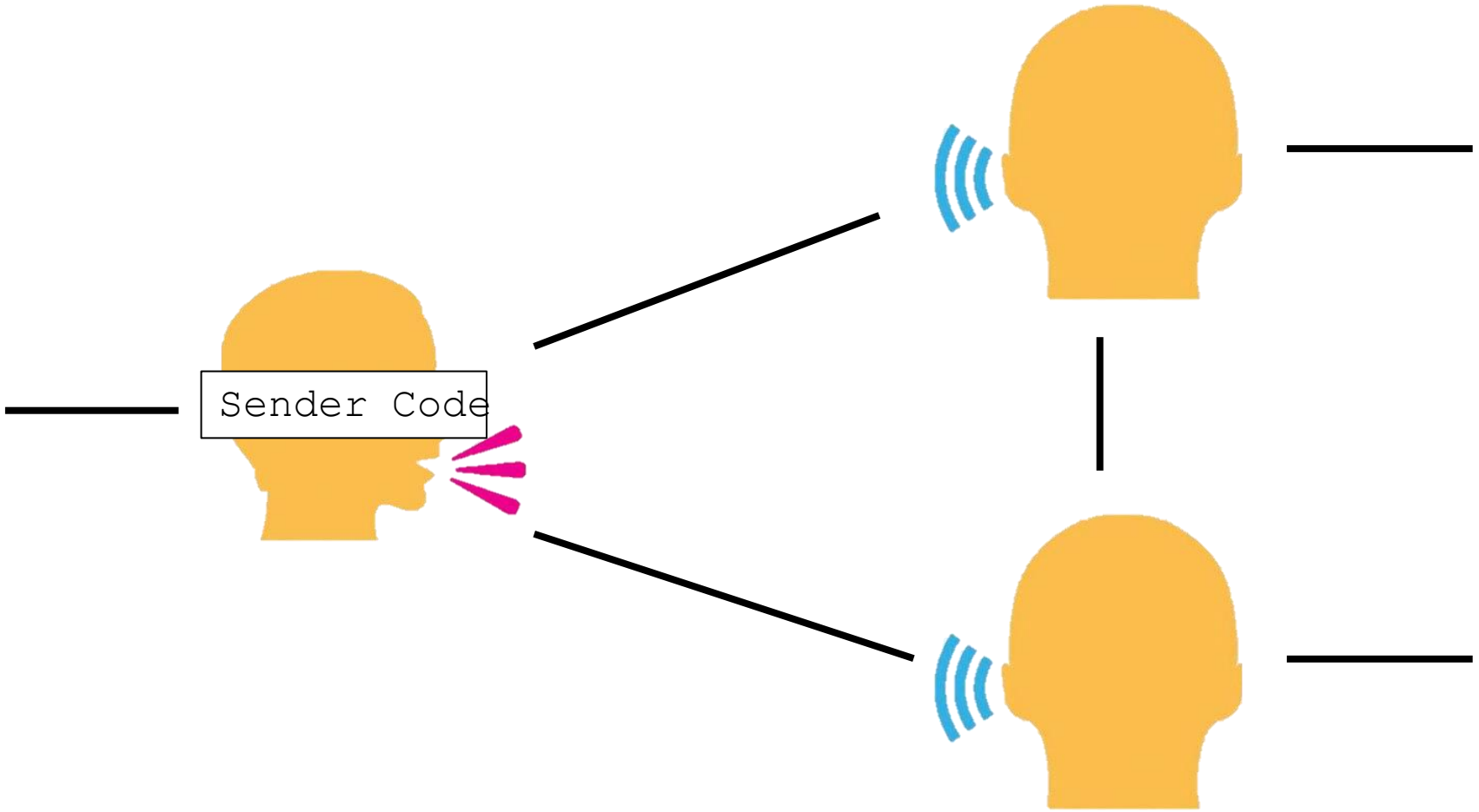
Additional Key Words and Phrases: Interactive consistency

Suppose it were possible…

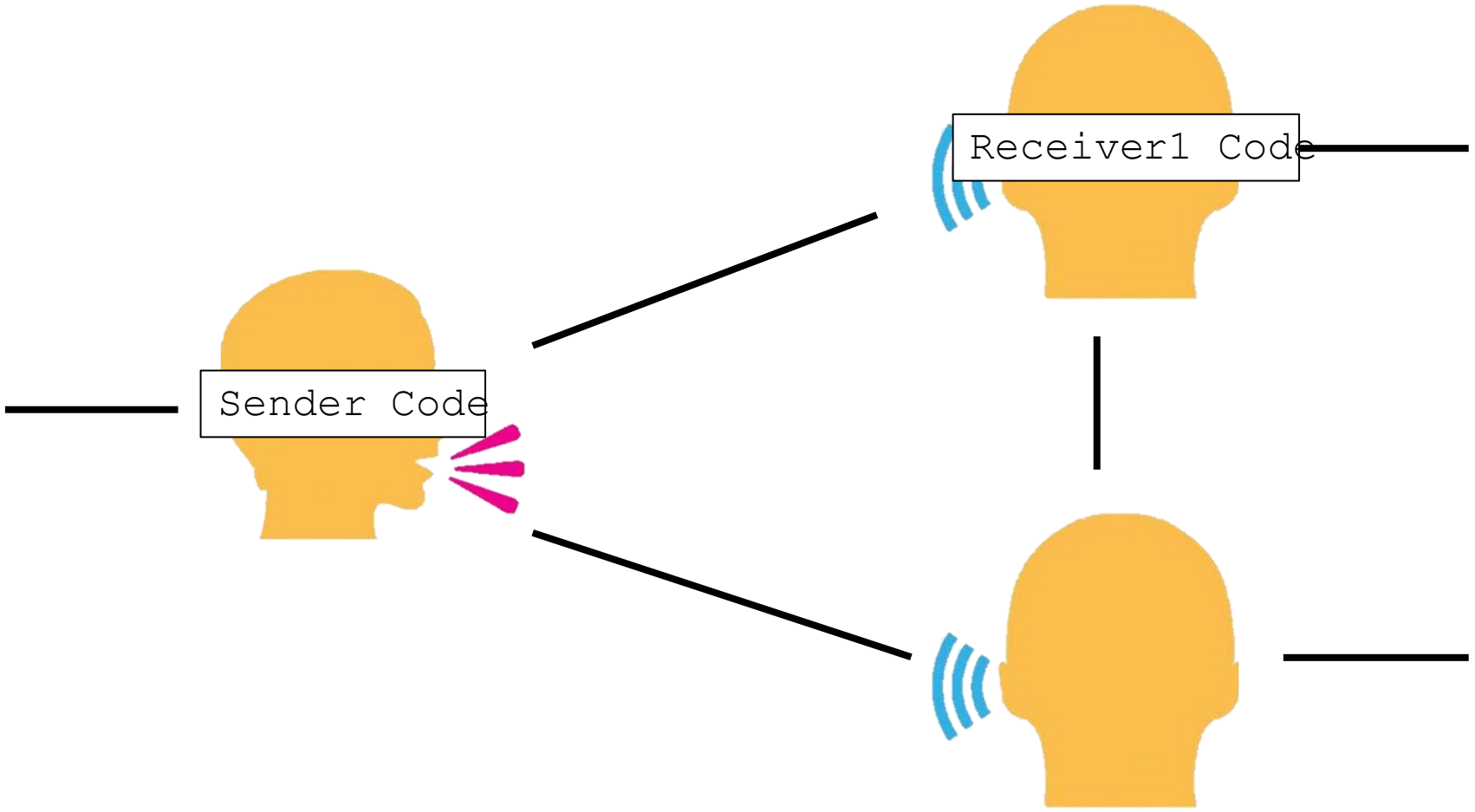
Suppose it were possible…



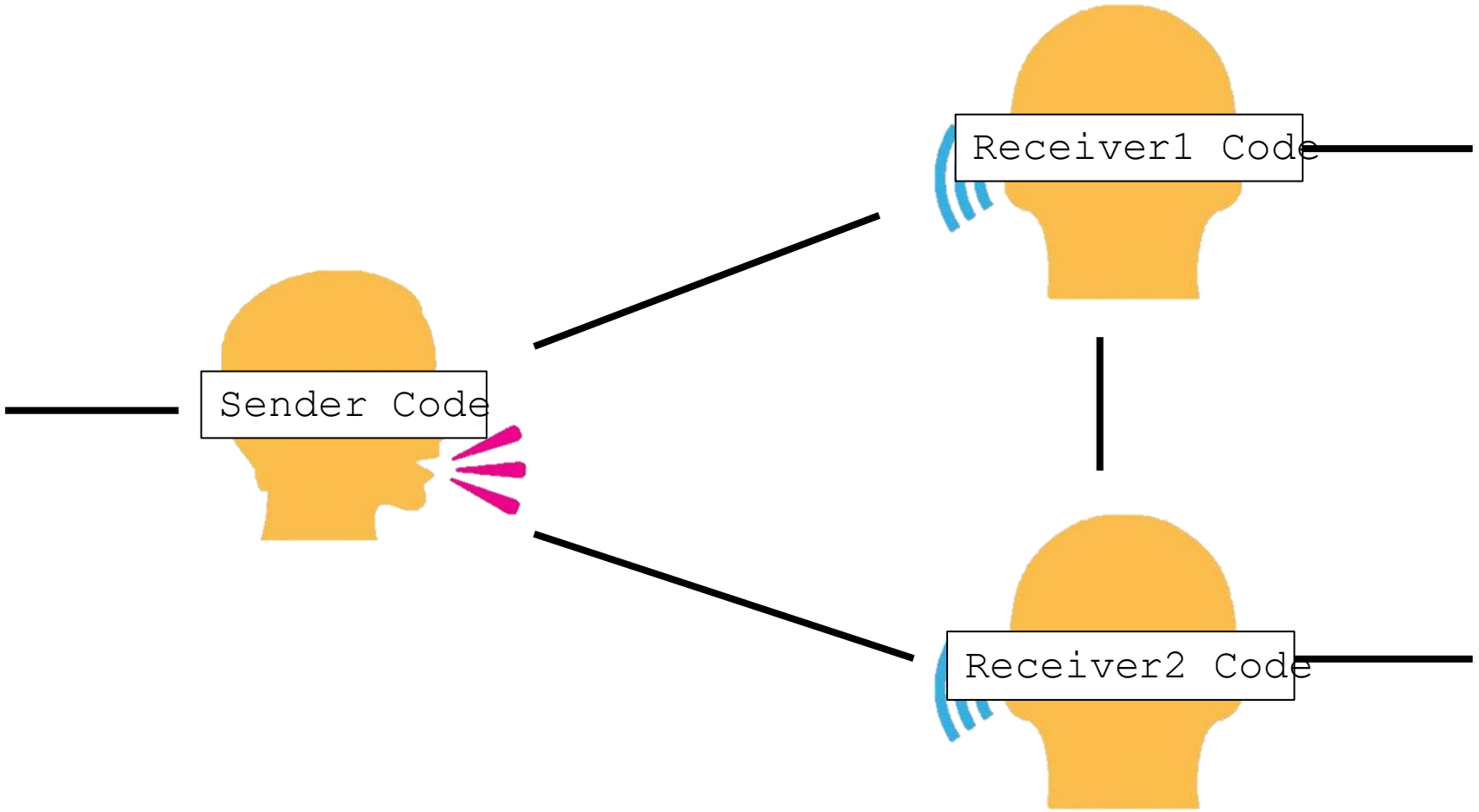
Suppose it were possible...



Suppose it were possible...

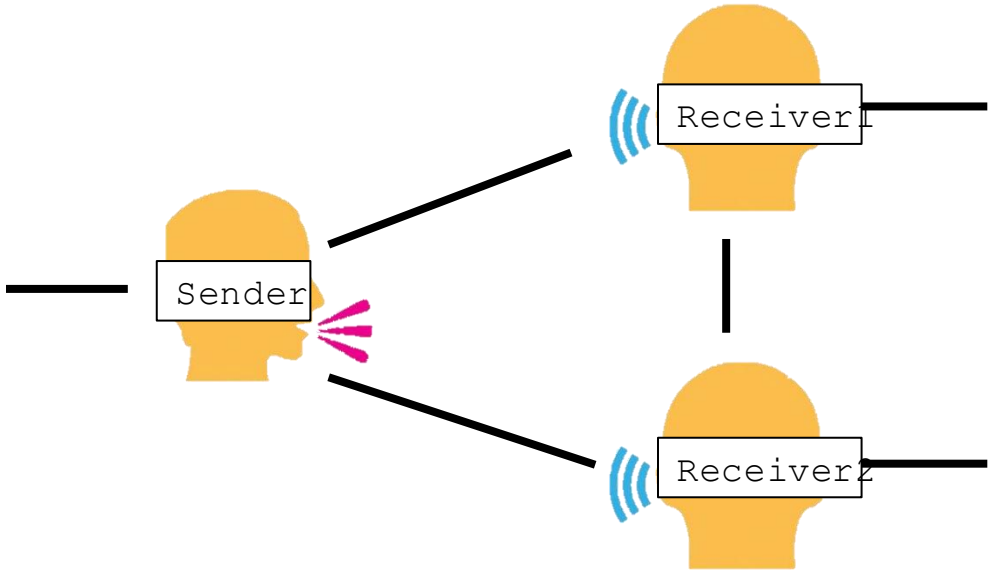


Suppose it were possible...

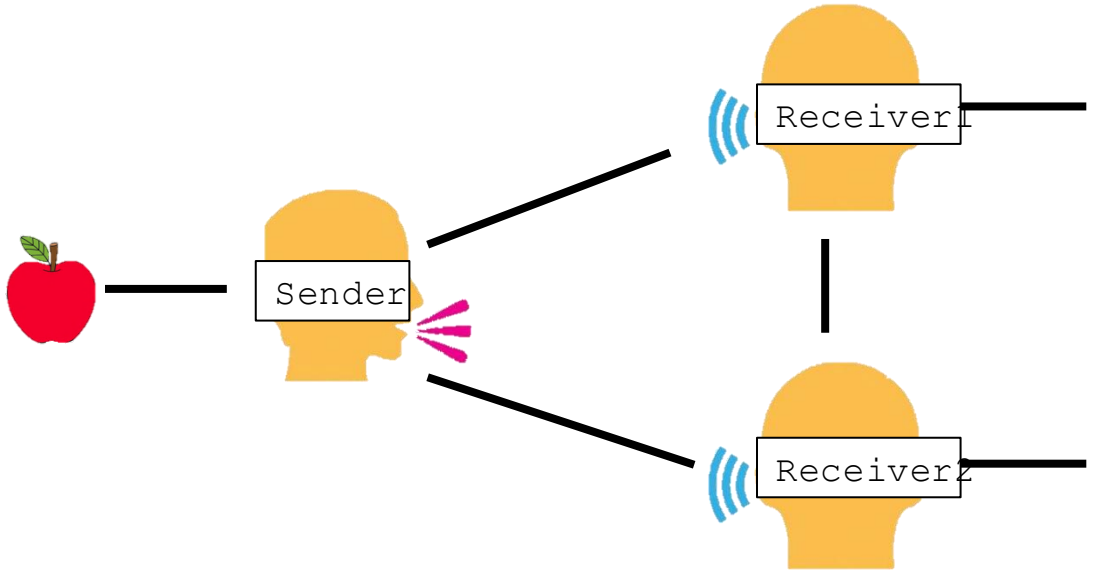


Validity

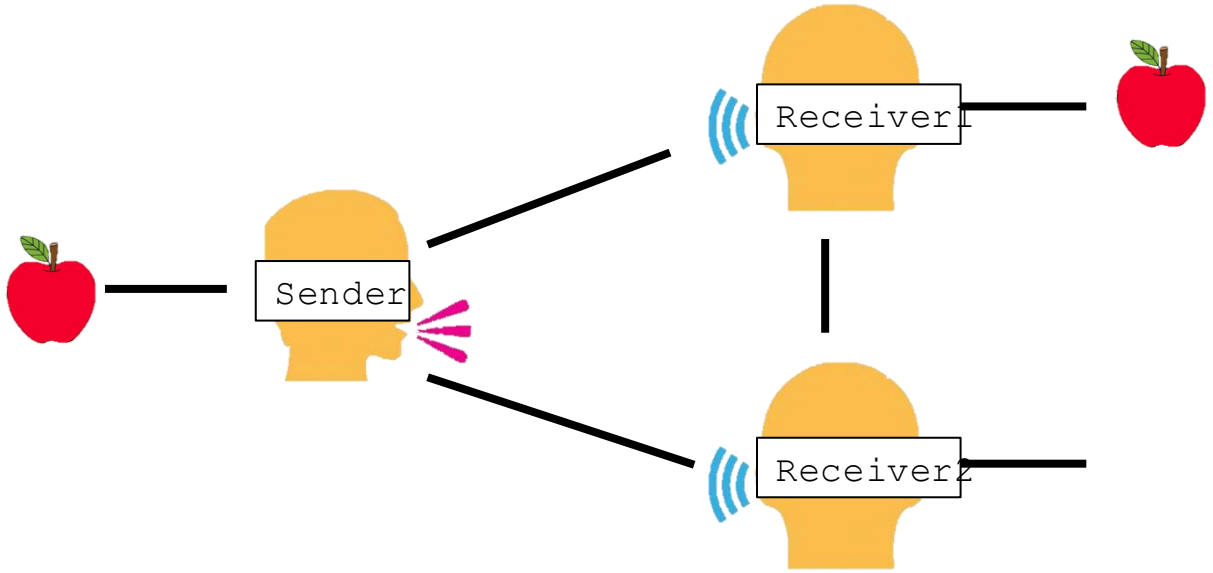
Validity



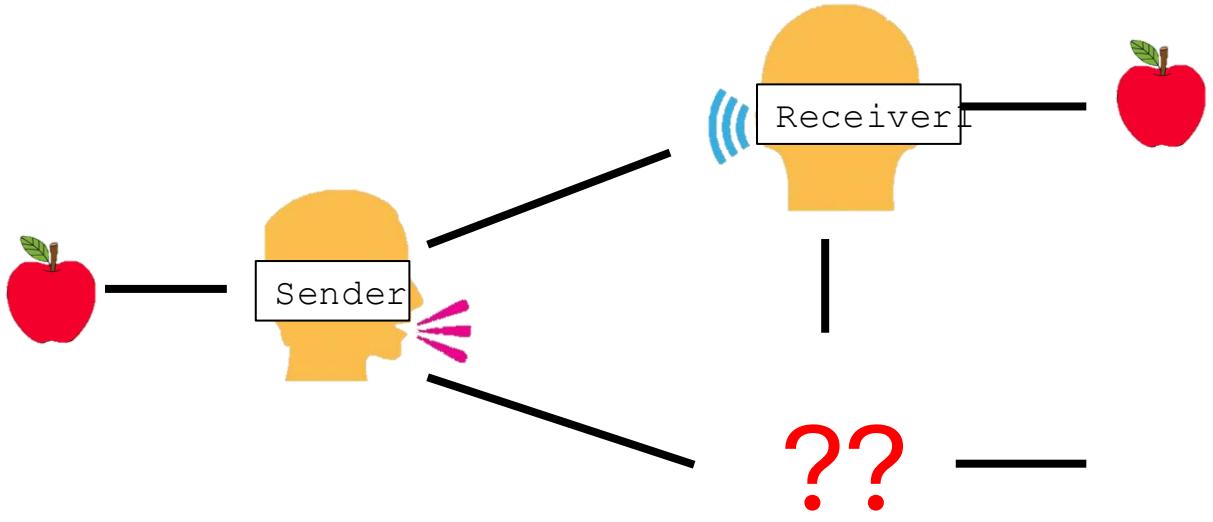
Validity



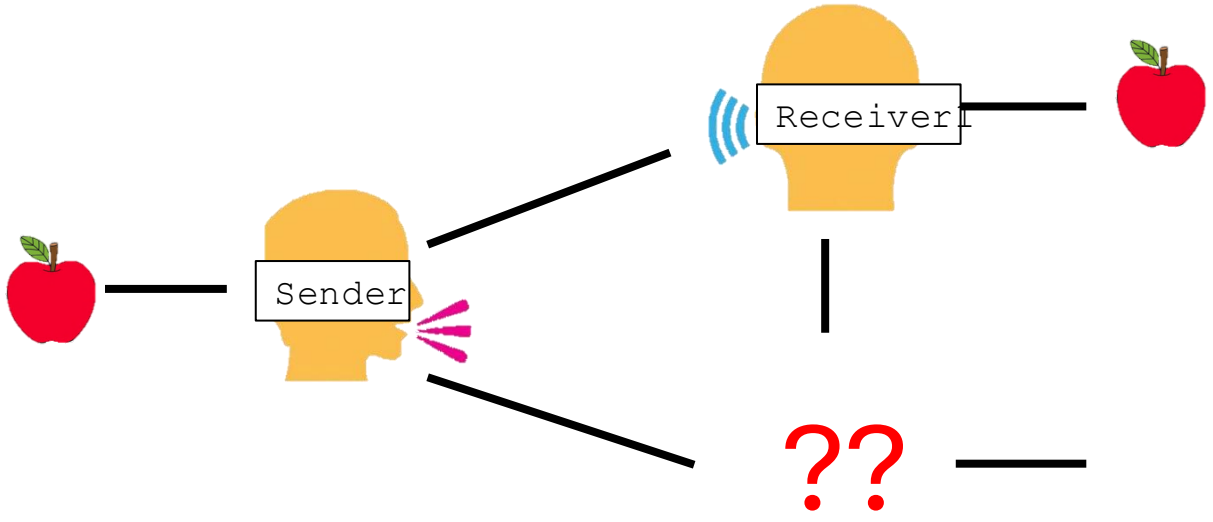
Validity

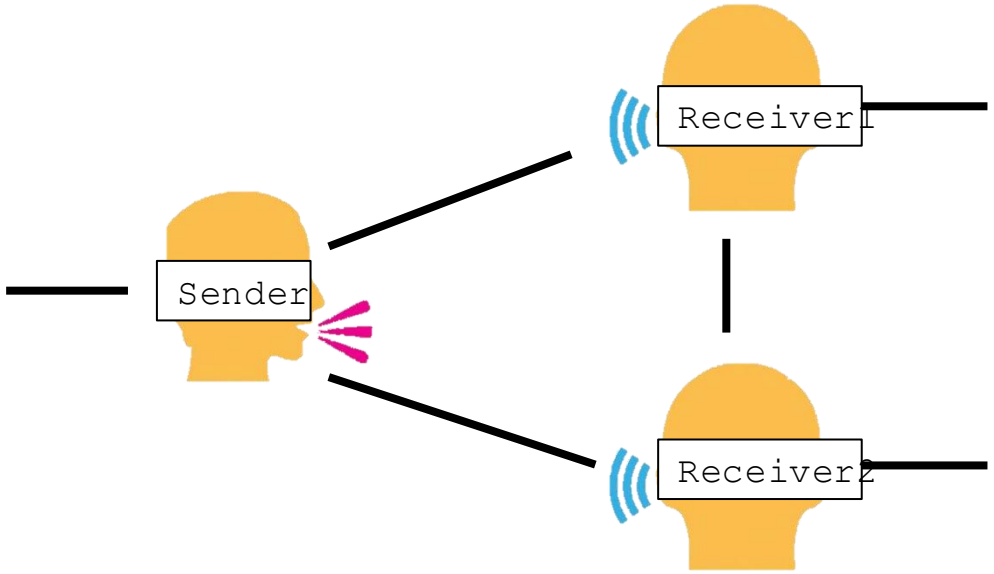


Validity

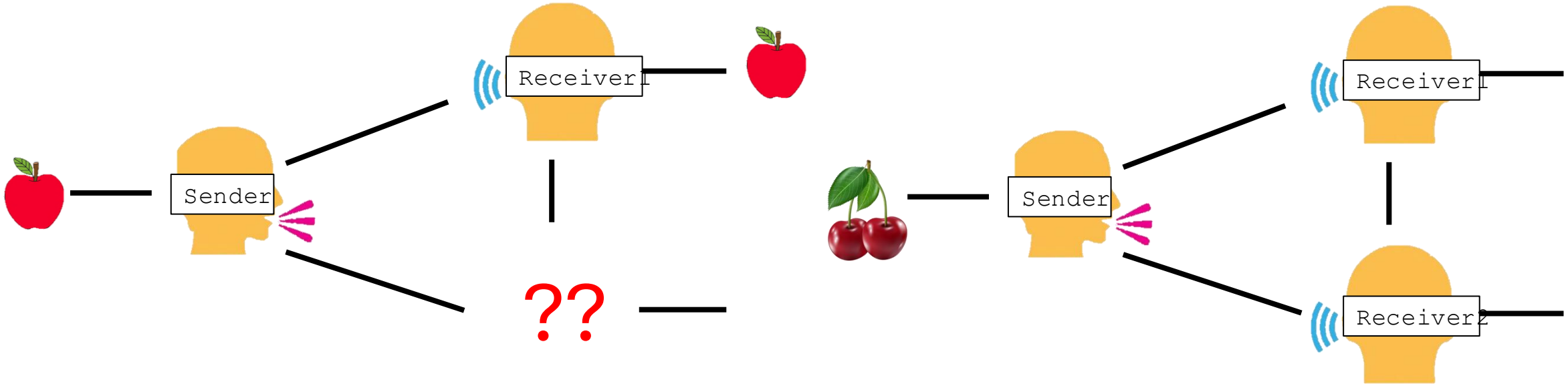


Validity

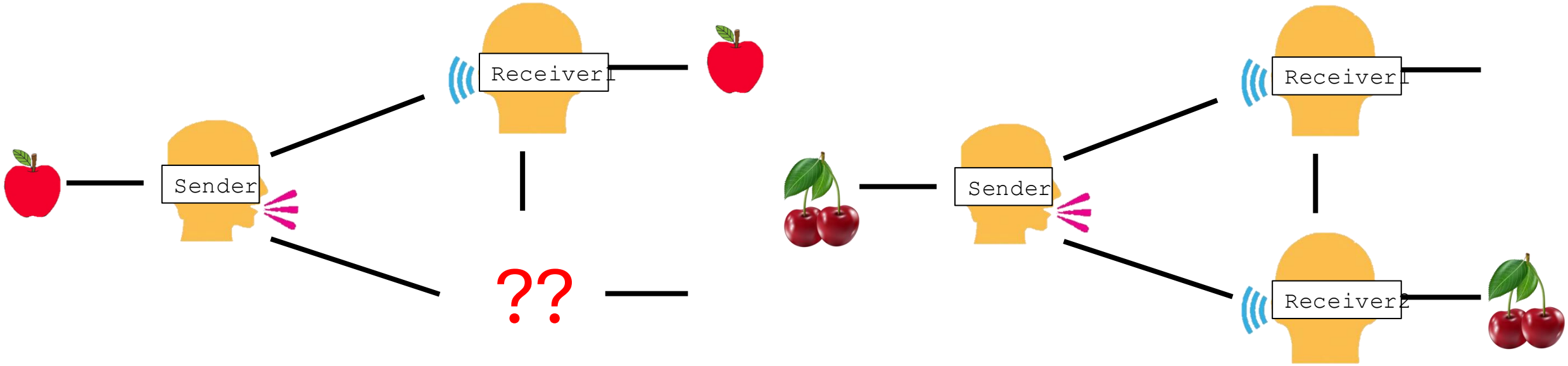




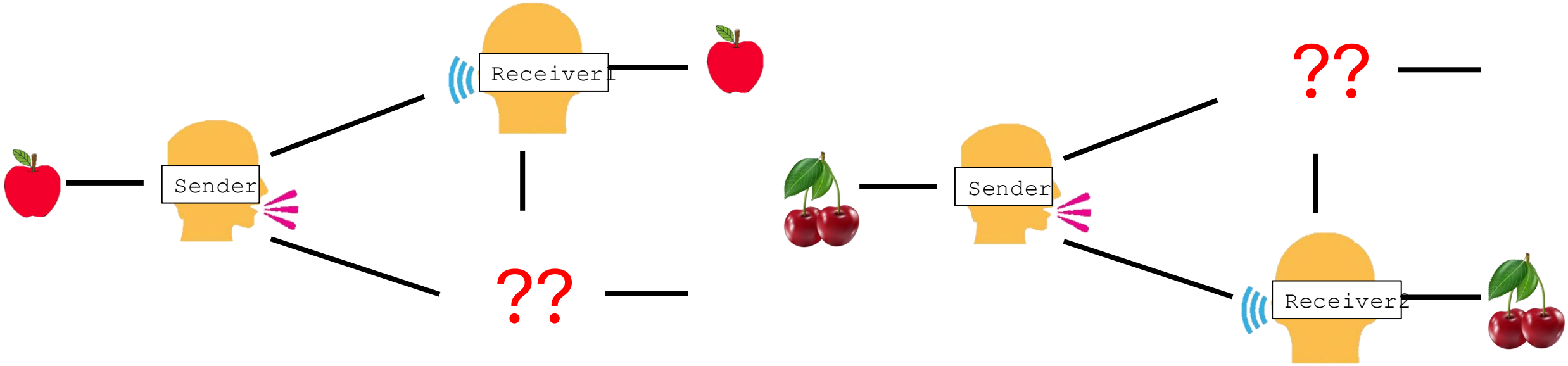
Validity



Validity

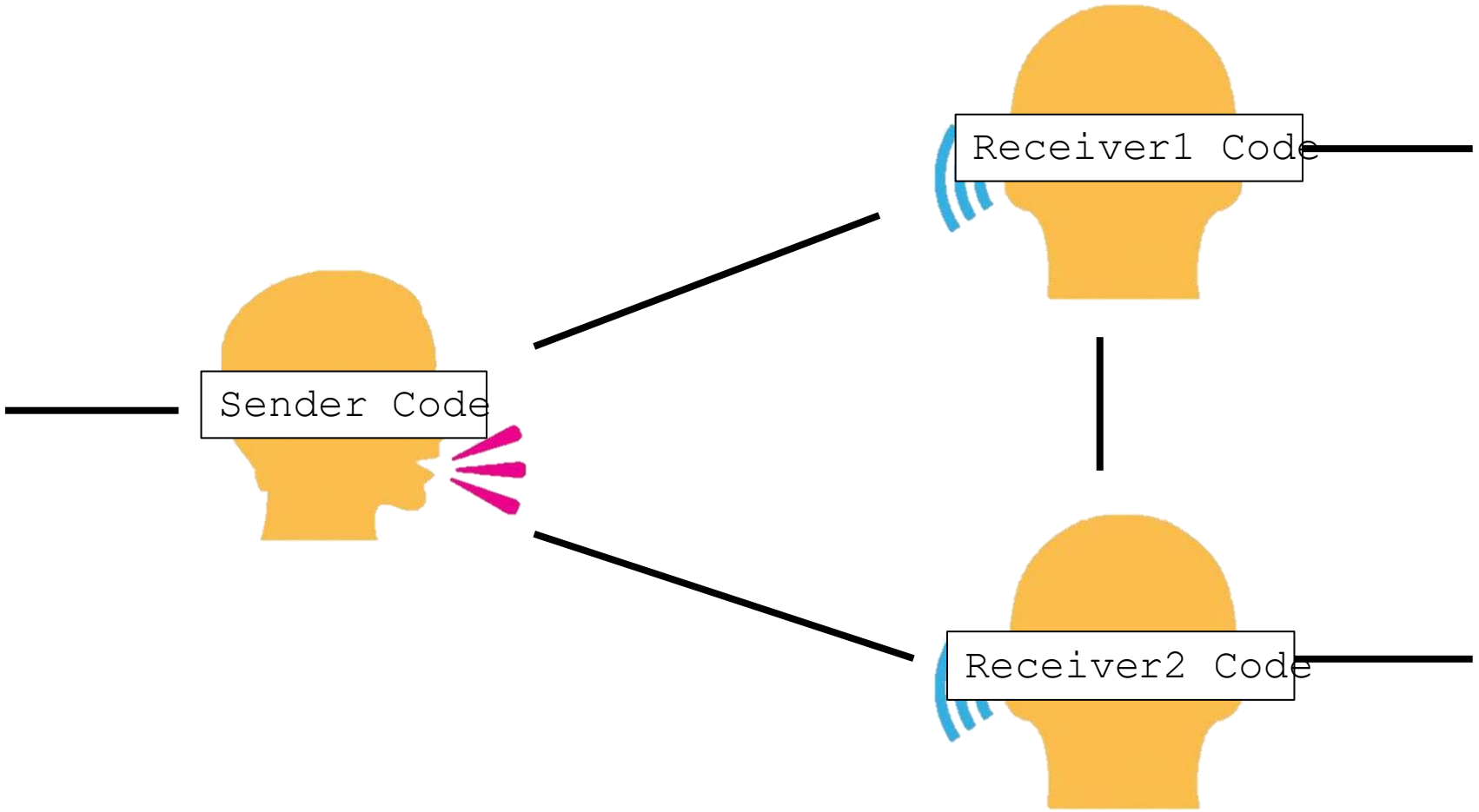


Validity

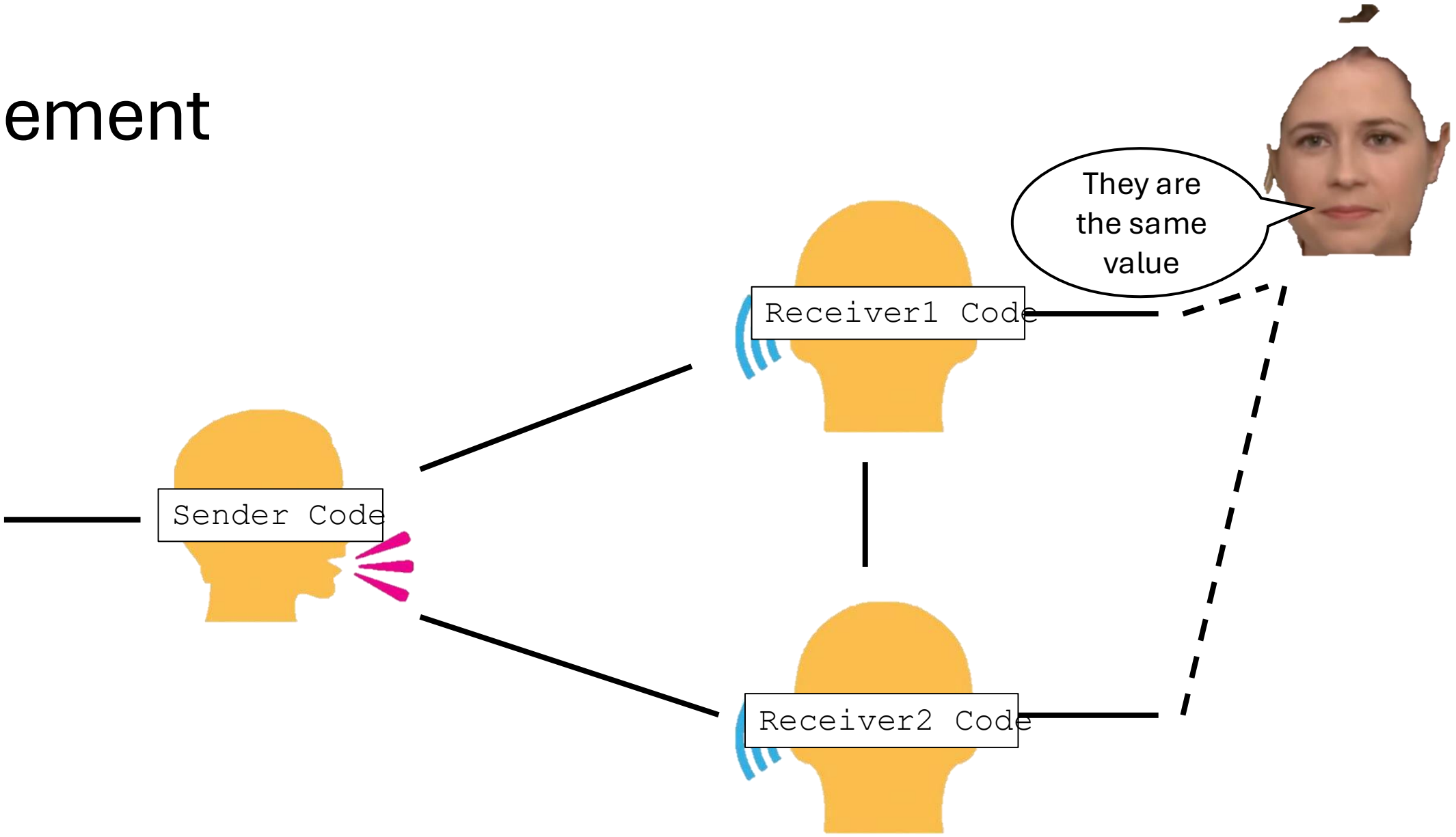


Agreement

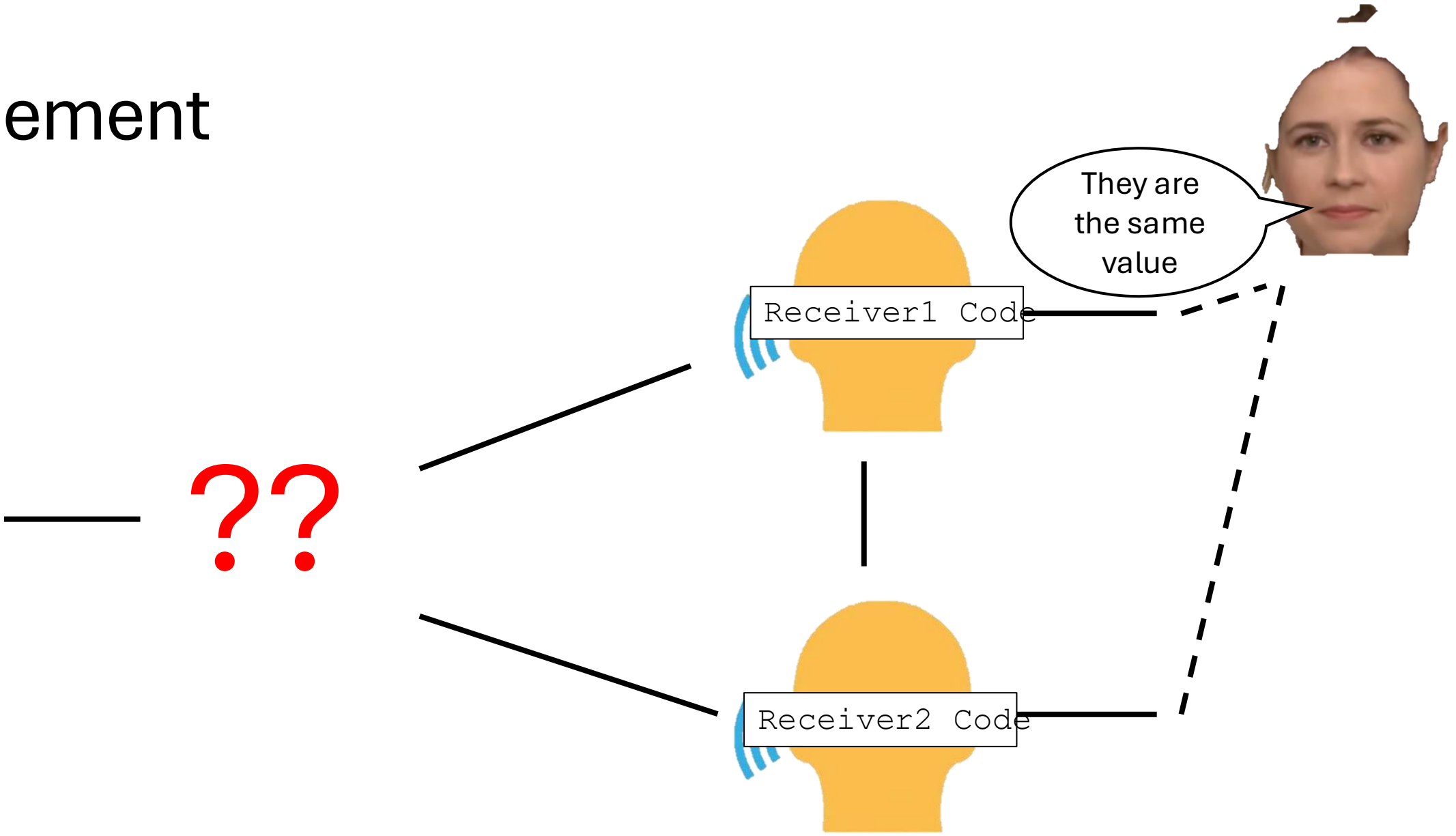
Agreement



Agreement

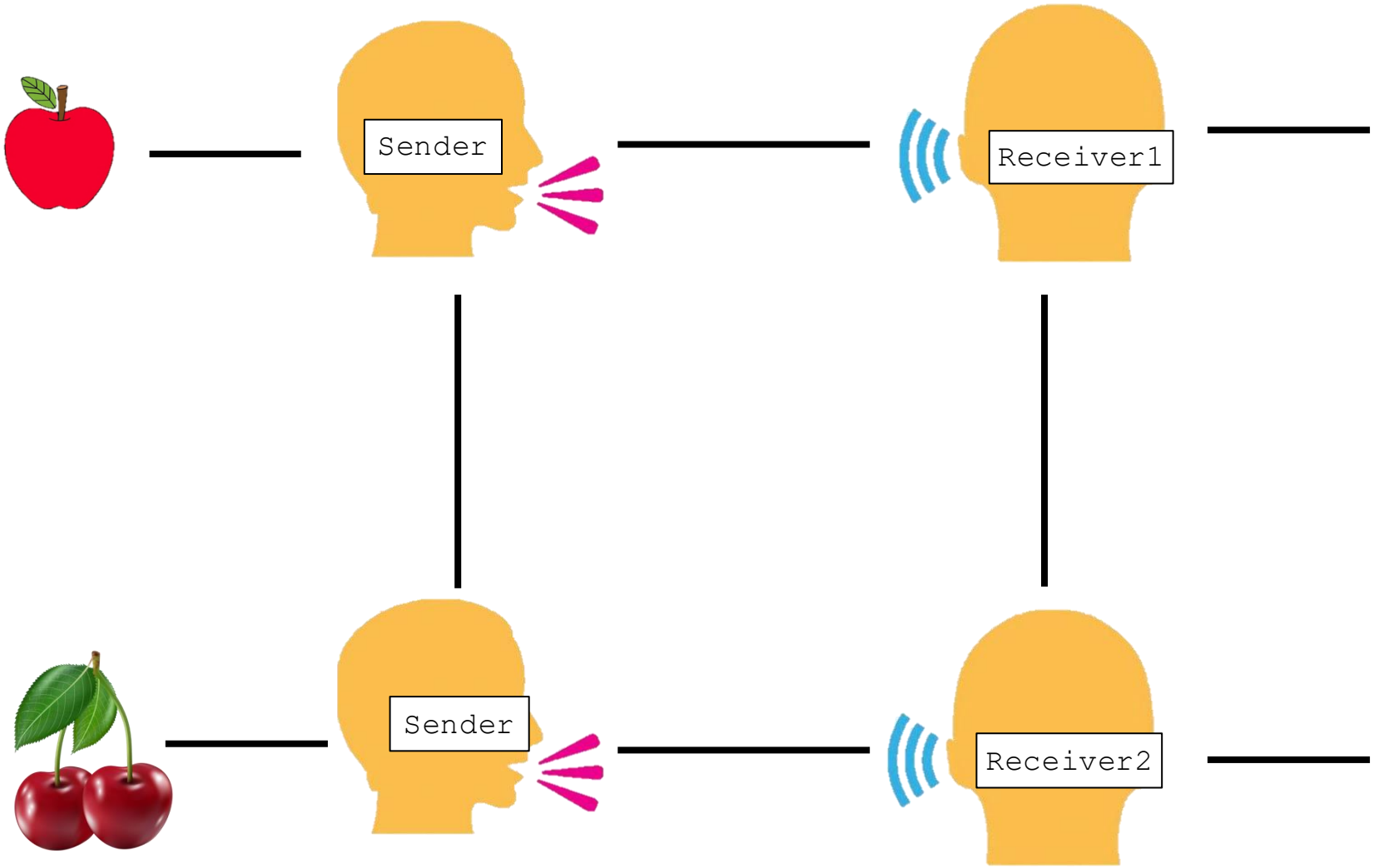

They are the same value
Receiver1 Code
Sender Code
Receiver2 Code

Agreement

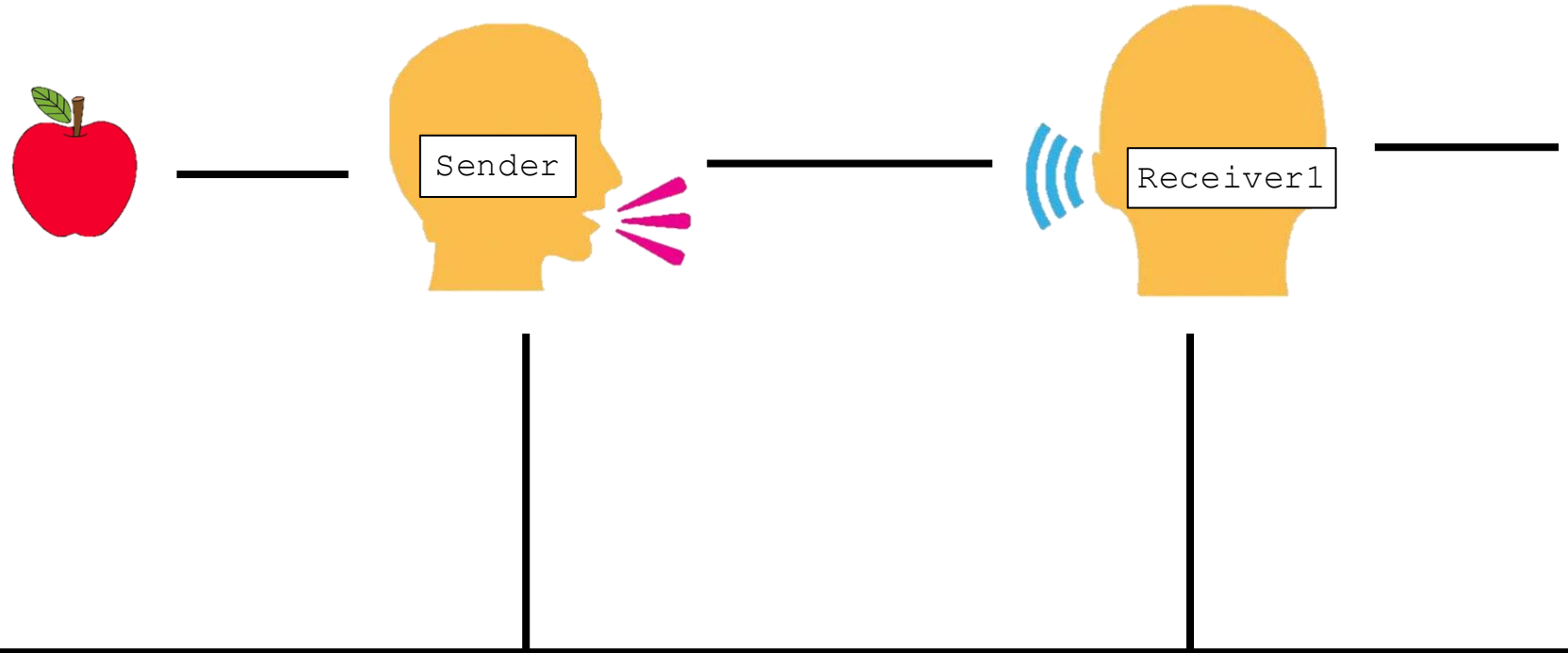


Wait.. That doesn’t make sense

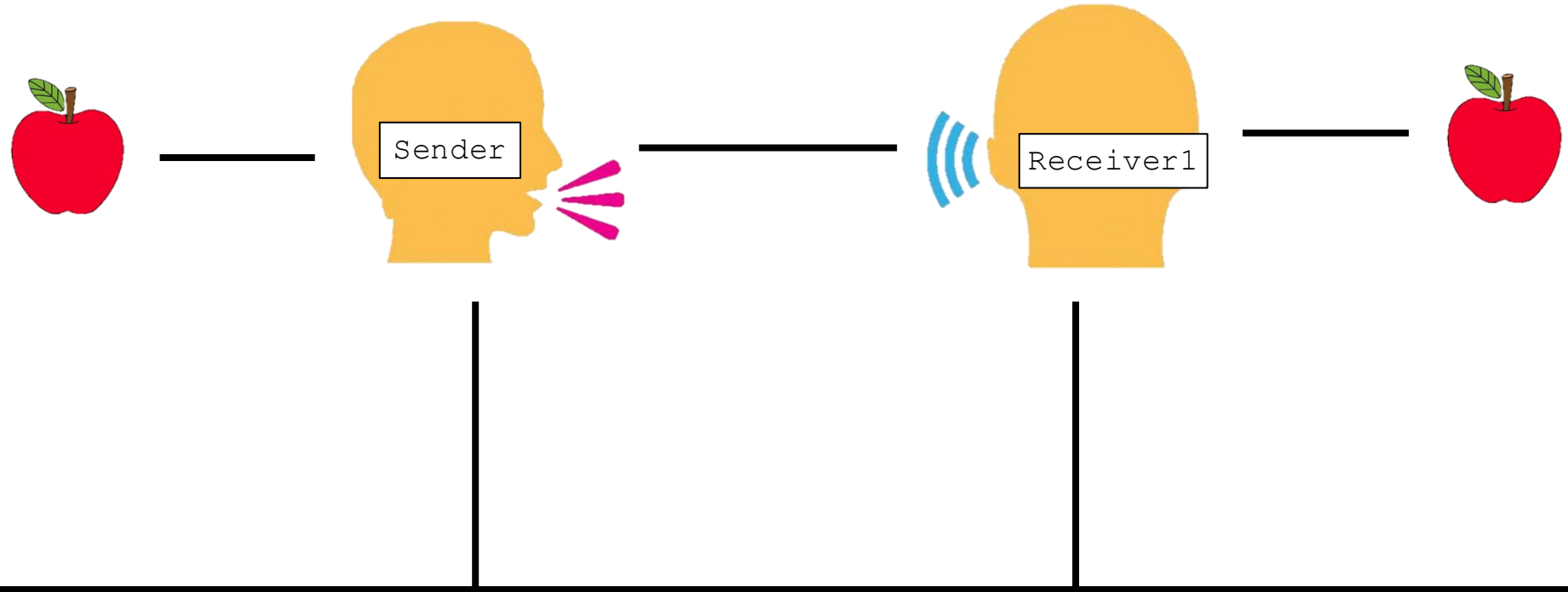
Wait.. That doesn’t make sense



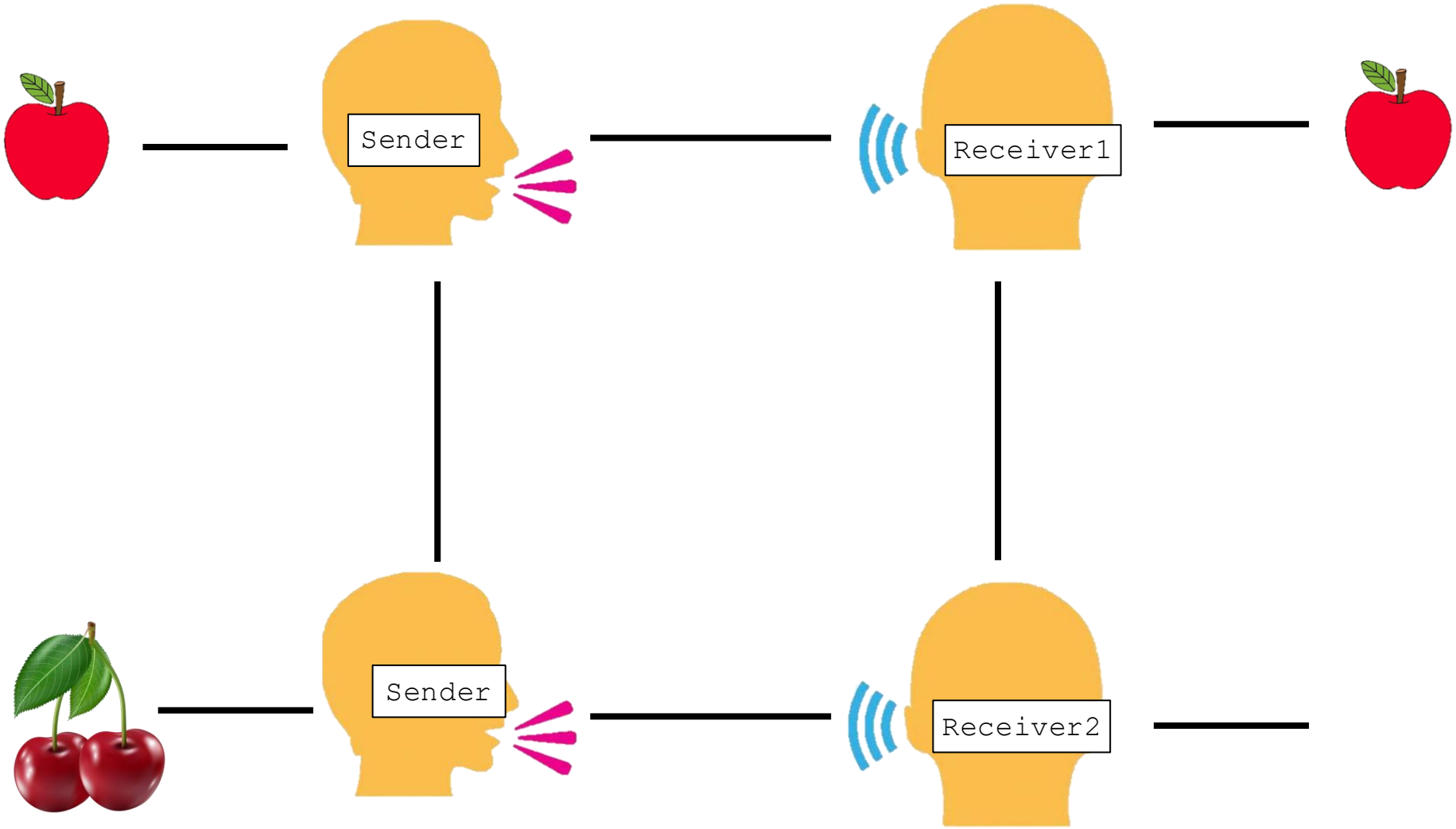
Wait.. That doesn’t make sense



Wait.. That doesn’t make sense

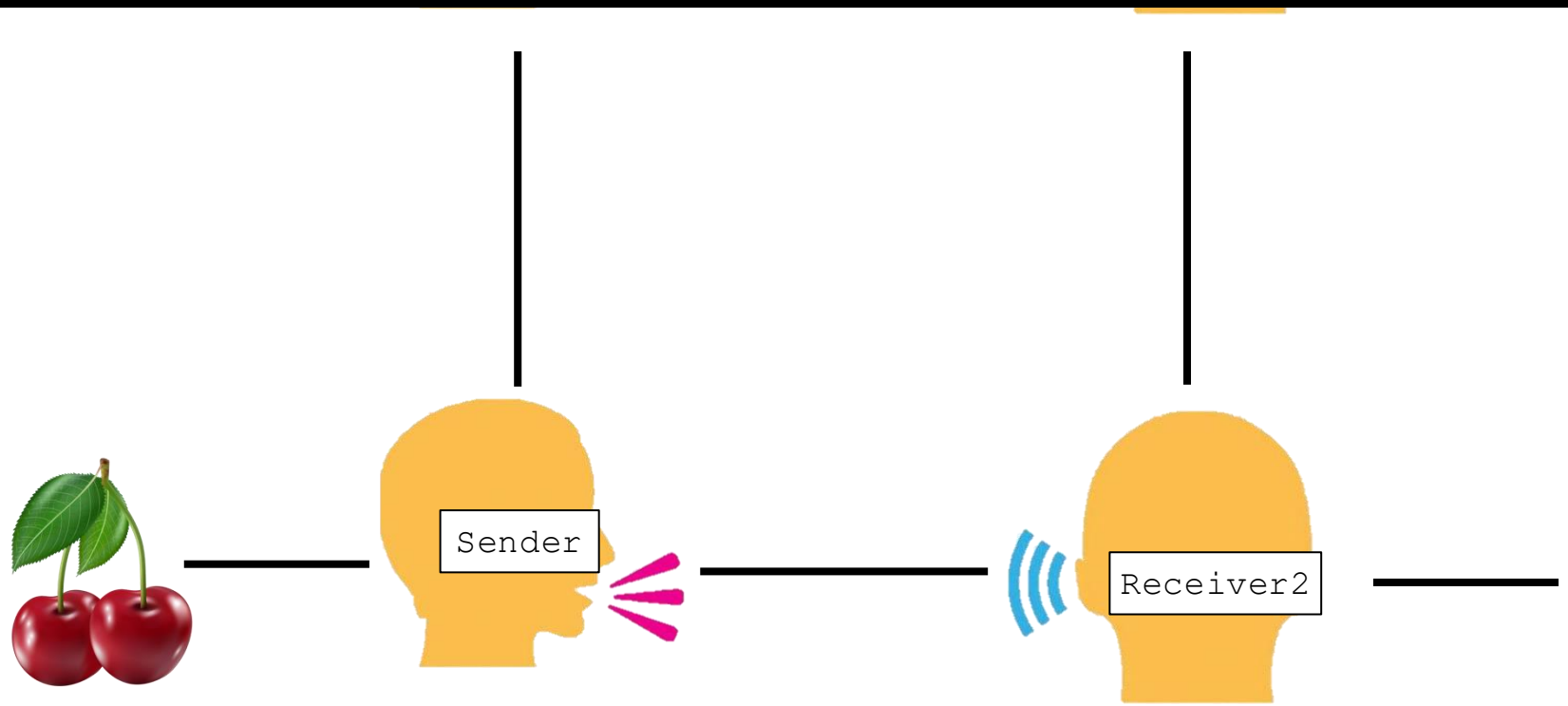


Wait.. That doesn’t make sense



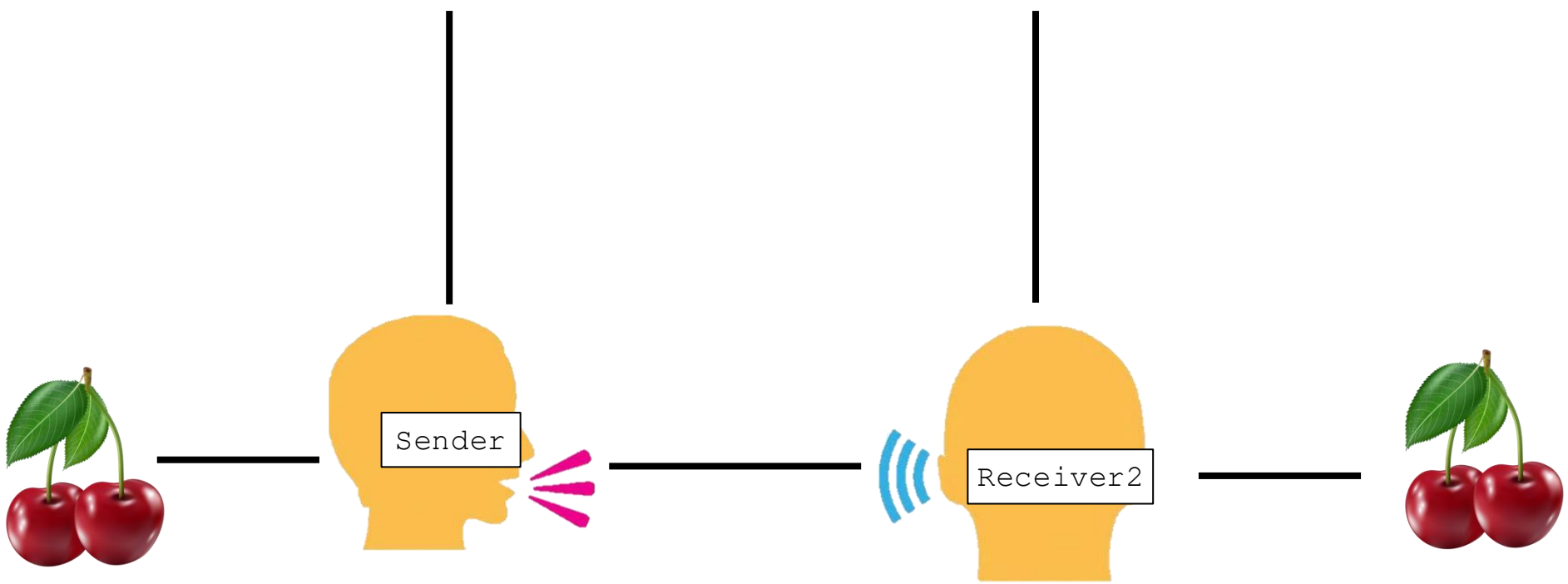
Wait.. That doesn’t make sense

??

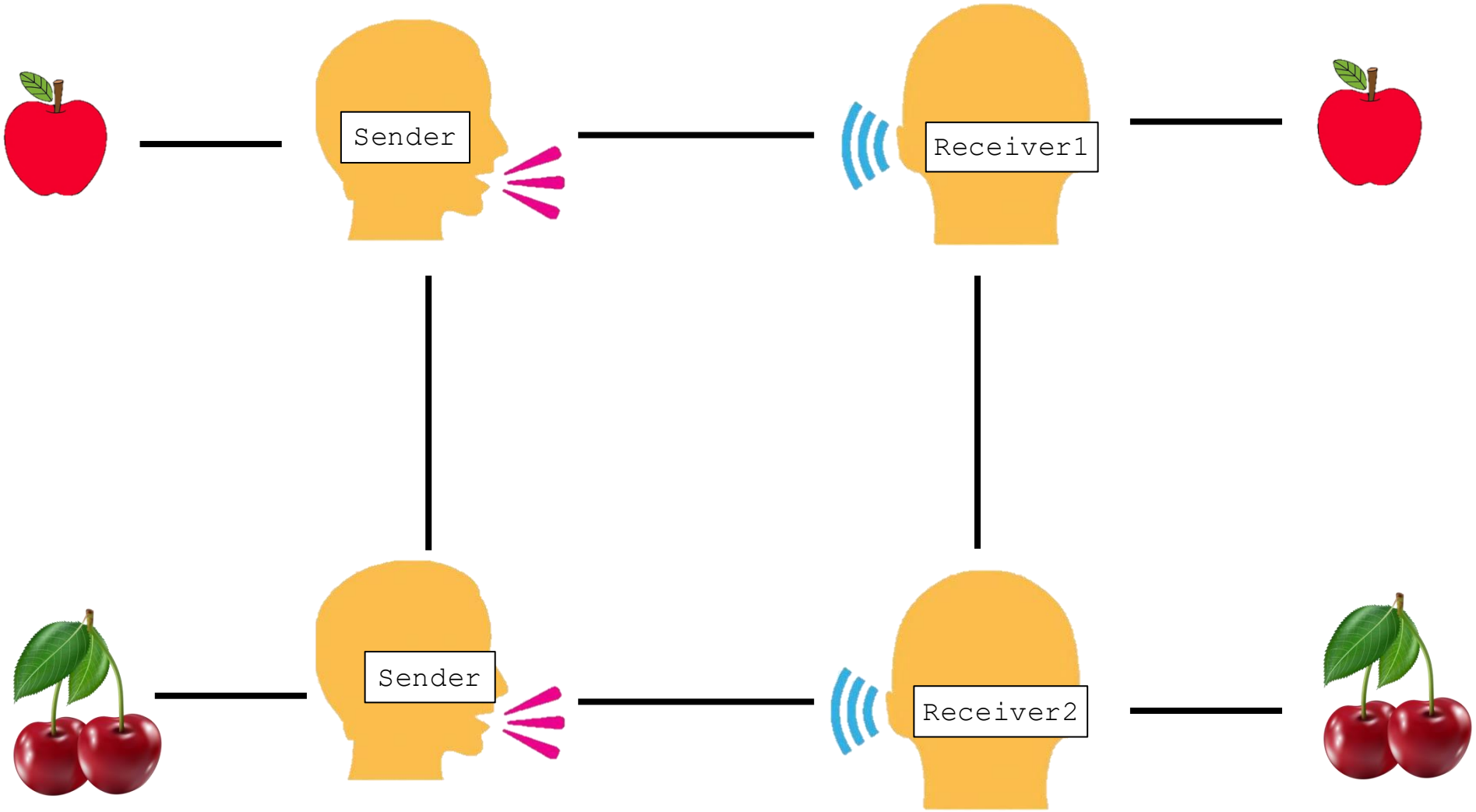


Wait.. That doesn’t make sense

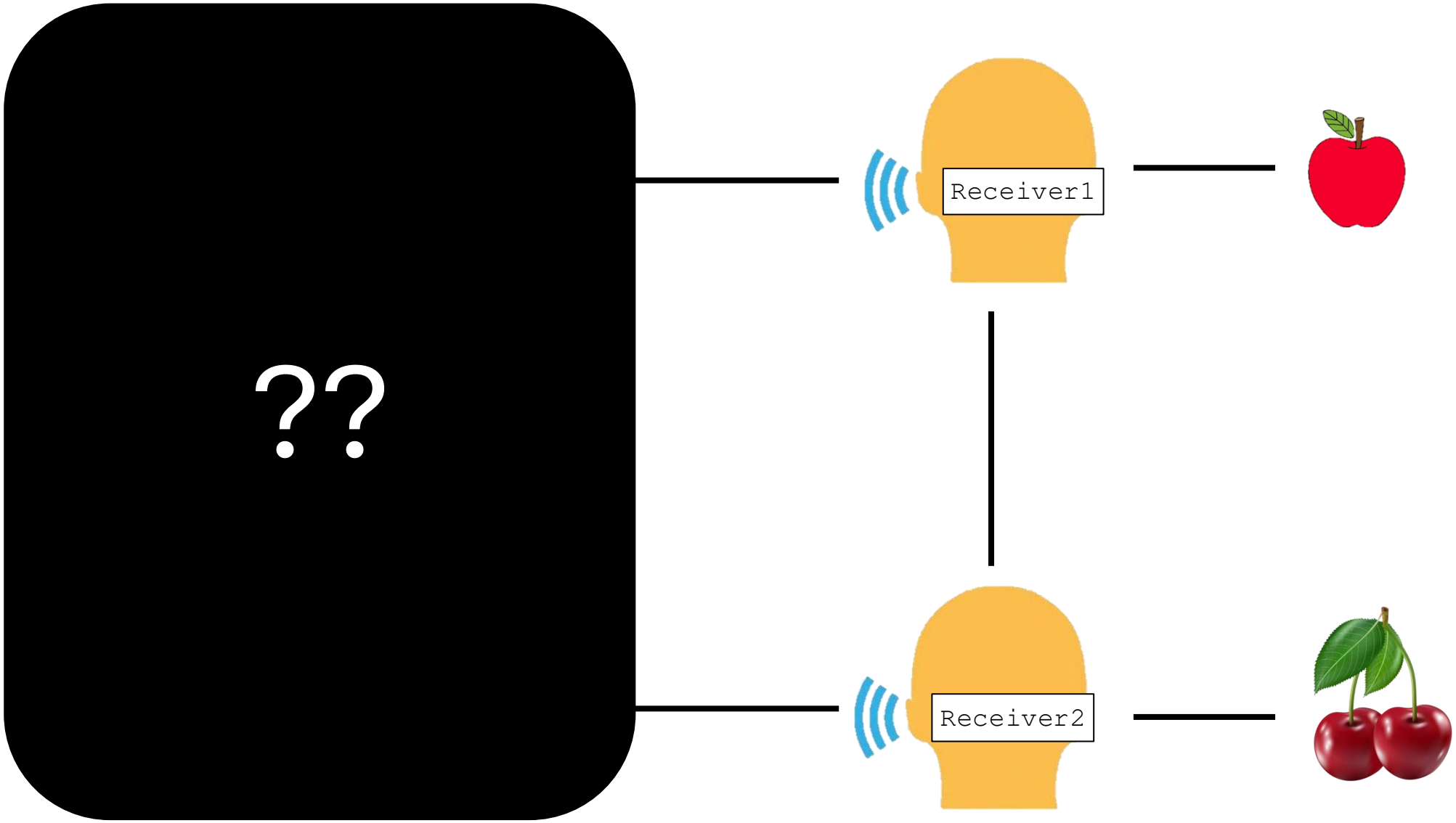
??



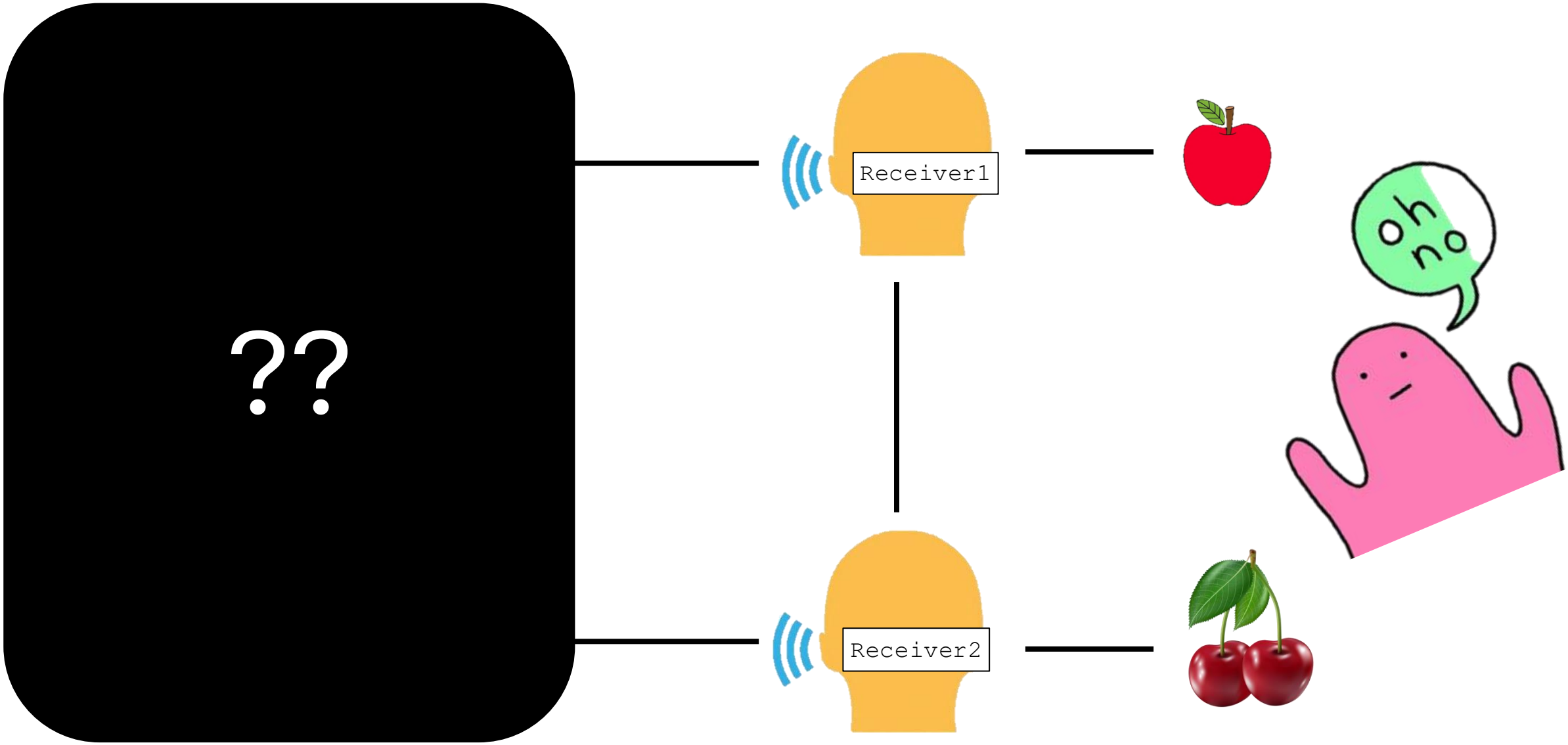
Wait.. That doesn’t make sense



Wait.. That doesn’t make sense



Wait.. That doesn’t make sense



Some History

Some History

Universally Composable Security

RAN CANETTI, Boston University

This work presents a general framework for describing cryptographic protocols and analyzing their security. The framework allows specifying the security requirements of practically any cryptographic task in a unified and systematic way. Furthermore, in this framework the security of protocols is preserved under a general composition operation, called universal composition. The proposed framework with its security-preserving composition operation allows for modular design and analysis of complex cryptographic protocols from simpler building blocks. Moreover, within this framework, protocols are guaranteed to maintain their security in any context, even in the presence of an unbounded number of arbitrary protocol sessions that run concurrently in an adversarially controlled manner. This is a useful guarantee, which allows arguing about the security of cryptographic protocols in complex and unpredictable environments such as modern communication networks.

CCS Concepts: • **Theory of computation → Computational complexity and cryptography**; **Cryptographic protocols**; *Concurrency*; **Distributed computing models**; **Process calculi**; • **Security and privacy → Formal methods and theory of security**;

Additional Key Words and Phrases: Security modeling, specification, and analysis, security-preserving composition, universal composition, modular security

ACM Reference format:
Ran Canetti. 2020. Universally Composable Security. *J. ACM* 67, 5, Article 28 (September 2020), 94 pages.
https://doi.org/10.1145/3402457

Some History

Universally Composable Security

RAN CANETTI, Boston University

This work presents a general framework for describing cry
The framework allows specifying the security requirement
and systematic way. Furthermore, in this framework the
composition operation, called universal composition. The
composition operation allows for modular design and anal
pler building blocks. Moreover, within this framework, pr
in any context, even in the presence of an unbounded nu
currently in an adversarially controlled manner. This is a
security of cryptographic protocols in complex and unpre
cation networks.

CCS Concepts: • **Theory of computation → Computational complexity and cryptography**; **Cryptographic protocols**; *Concurrency*; **Distributed computing models**; **Process calculi**; • **Security and privacy → Formal methods and theory of security**;

Additional Key Words and Phrases: Security modeling, specification, and analysis, security-preserving composition, universal composition, modular security

ACM Reference format:
Ran Canetti. 2020. Universally Composable Security. *J. ACM* 67, 5, Article 28 (September 2020), 94 pages.
https://doi.org/10.1145/3402457

I'm reading Canetti's original papers (UC with Global Setup, 2006 and UC Authentication and Key Exchange, 2014) and find it so hard to dig through the handwavy proofs and spaghetti code (written in English). I get the overall idea of simulation and encapsulating processes external to the protocol inside environment Z, but when I get to the nuances of how corruptions work and similar, there's a lot of things that are implicit and swept under the carpet. I can see that the idea is powerful, but the presentation certainly could be improved, and I was wondering if someone did improve it and any of you could direct me to such resource.

Summing it up…

Summing it up…

Systems consist of smaller components **composed** together.

Summing it up…

Systems consist of smaller components **composed** together.

- This paper proposes an **Algebraic Theory of Systems**

Summing it up…

Systems consist of smaller components **composed** together.

- This paper proposes an **Algebraic Theory of Systems**
- Formalizes commonly used composition operations, such as parallel and self composition

Summing it up…

Systems consist of smaller components **composed** together.

- This paper proposes an **Algebraic Theory of Systems**
- Formalizes commonly used composition operations, such as parallel and self composition
- Hierarchy of useful ‘models’ of this theory.

Summing it up…

Systems consist of smaller components **composed** together.

- This paper proposes an **Algebraic Theory of Systems**
- Formalizes commonly used composition operations, such as parallel and self composition
- Hierarchy of useful ‘models’ of this theory.
- The provided proof for Byzantine Broadcast impossibility argues for proofs at the highest possible level of abstraction.

Toward an Algebraic Theory of Systems

Christian Matt, Ueli Maurer, Christopher Portmann, Renato Renner, Björn Tackmann