1.S980: Applied Category Theory for Engineering Design (ACT4ED)

Lecture 1: Introduction, Logistics, Motivation, Compositionality

September 4, 2024

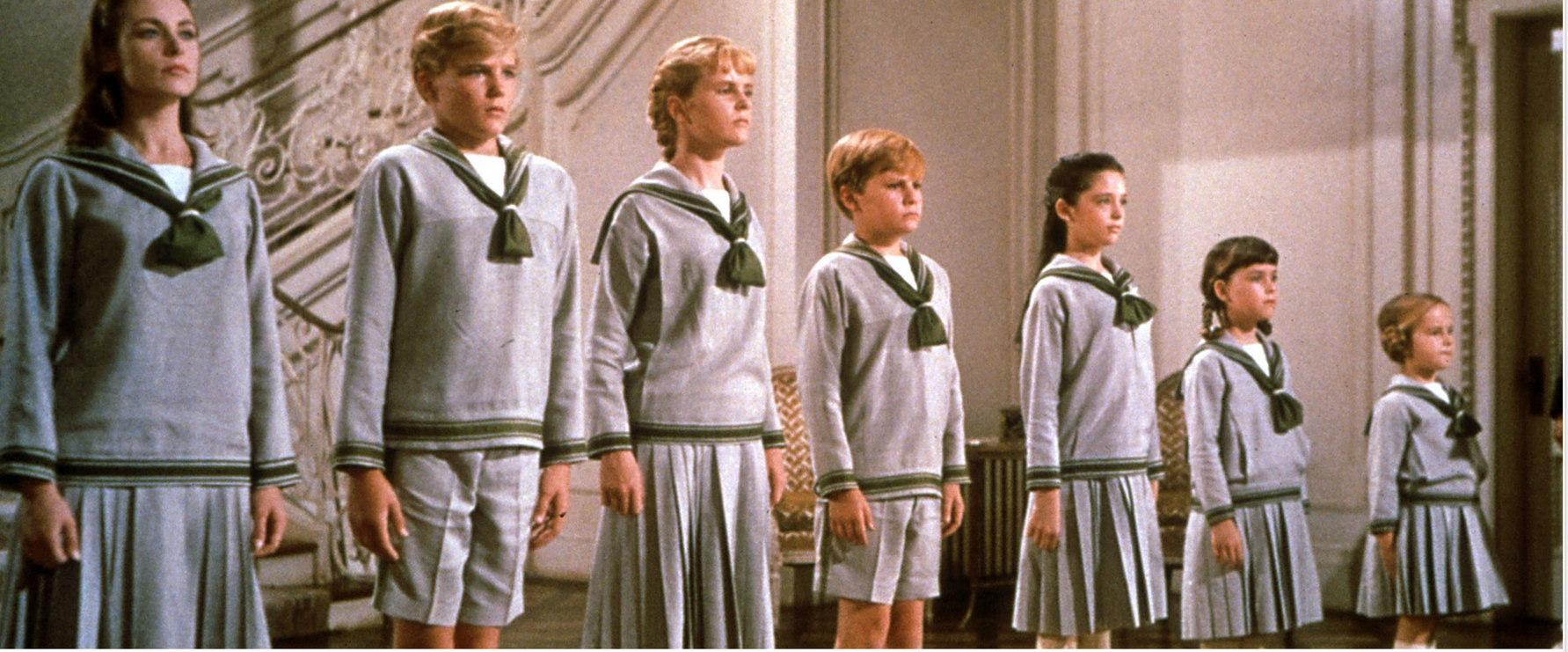
Gioele Zardini

Civil and Environmental Engineering

LIDS

IDSS
MIT INSTITUTE FOR DATA, SYSTEMS, AND SOCIETY

MIT



Introductions

Who we are

Instructor:

- Prof. **Gioele** Zardini
  - From Mendrisio, Switzerland
  - Ph.D. in Dynamic Systems and Control, ETH Zurich
  - Rudge (1948) and Nancy Allen Assistant Professor, MIT

TA:

- **Yujun** Huang
  - Graduate Student (CEE) in Prof. Zardini's group
  - From Wuxi, Jiangsu, China
  - MSc. in Robotics, Systems and Control from ETH Zurich

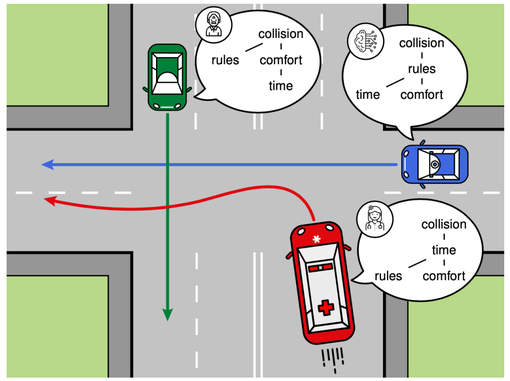
The Zardini Lab at MIT

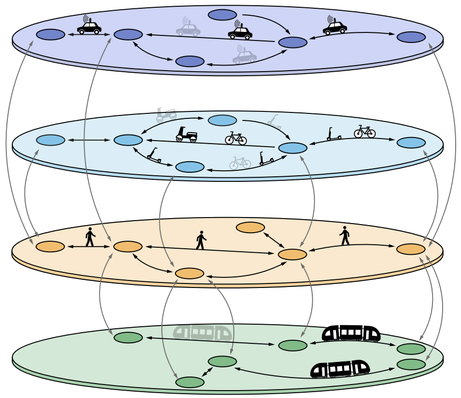
Driven by **societal challenges**, we develop **efficient computational tools**, and **algorithms** to formulate and solve complex, **interconnected system design** and **autonomous decision making problems**

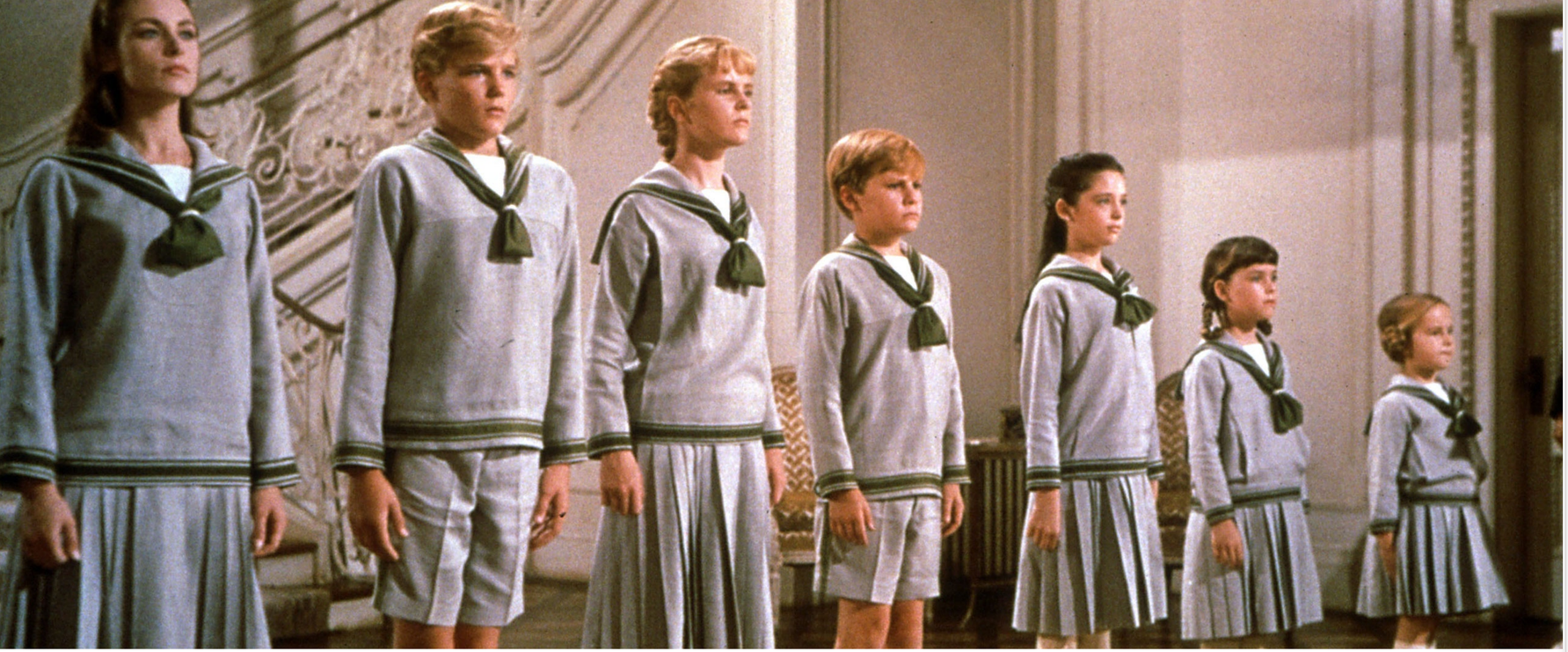
Modeling & Algorithmic Foundations

Societal Applications

User-friendly Tools







Who are you?

Logistics

Lecture Scheduling

- **Lectures**
  - **Mondays,** 11:00 - 12:30 pm (1-134)
  - **Wednesdays,** 11:00 - 12:30 pm (1-134)

- **Recitals**
  - **Fridays,** 1:00 - 2:00 pm (1-134)
  - This is practice for theory exercises, as well as Q&A.

- **Office hours**
  - By appointment

- See the official **website** for an up-to-date schedule.

https://zardini.mit.edu/act4ed/

Grading and Course Requirements

- **Problem sets**
  - **40%**
  - 1 every ~10 days, check the syllabus for important dates
  - Less than 24h late, graded 80% of the total score, 24h to 48h late, graded at 50%
- **Midterm**
  - **30%**
  - 10/21/24, during lecture time
- **Final project**
  - **30%**
  - Final presentations on 12/4/24 and 12/6/24, final report

Der Erziehungsrat
des
Kantons Aargau
urkundet hiemit:

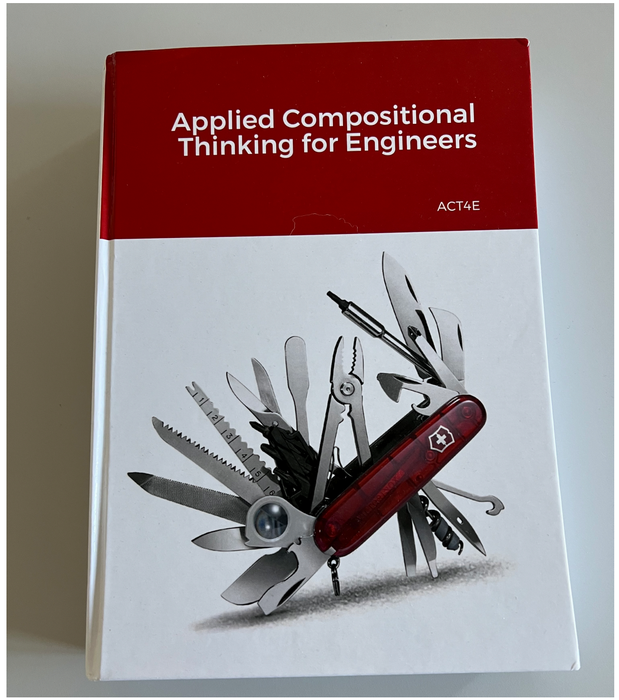
Herr Albert Einstein von Ulm,
geboren den 14. März 1879,
besuchte die aargauische Hauptnormalschule d. h. nur die III. & IV. Klasse
der Gewerbeschule.
Nach abgelegter schriftl. & mündl. Maturitätsprüfung am 18., 19. & 21.
September, sowie am 30. September 1896, erhielt derselbe folgende Noten:

1. Deutsche Sprache und Literatur	5
2. Französisch	5
3. Englisch	—
4. Italienisch	5
5. Geschichte	6
6. Geographie	4
7. Algebra	6
8. Geometrie (Planimetrie, Trigonometrie, Stereometrie & analytische Geometrie)	6
9. Darstellende Geometrie	6
10. Physik	6
11. Chemie	5

Course Materials

- **CANVAS** will contain:
  - Annotated slides (regular slides posted just before the lecture)
  - Extra reading, when needed

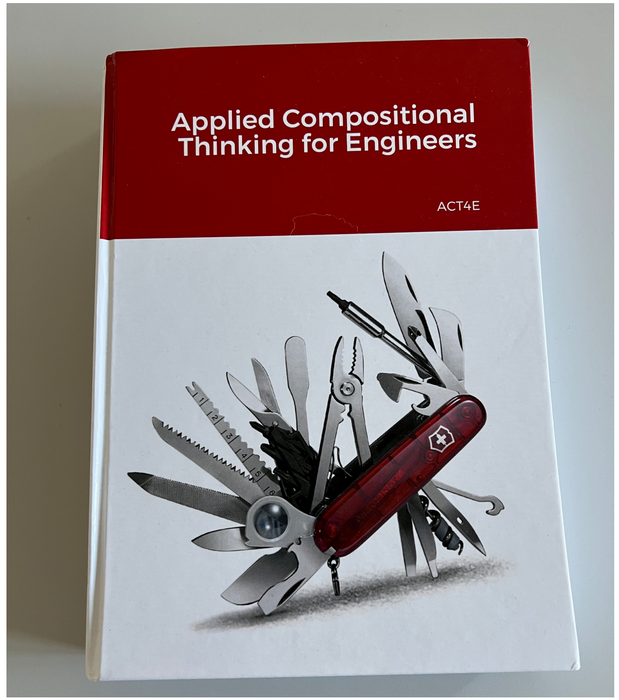
- The **book** is the main source of material.
  - Find the link on CANVAS or website
  - Please report typos via GitHub!



Notes about the book

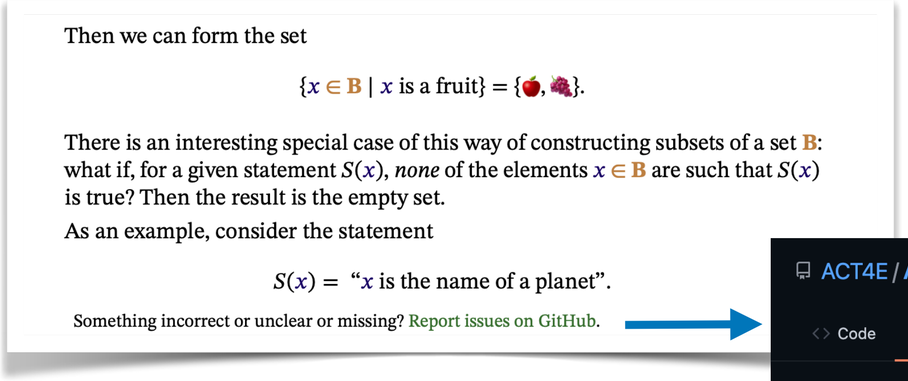
- **The book covers more than this class**
  - There are a lot of theory exercises in the book: but not all are necessary!
  - There are a lot of code exercises in the book: not covered yet here!
  - There are many solved exercises in the book:
    the level of formality is what we look for in the problem sets.

- The book is **updated periodically**:
  download it again, and you will see changes.
- We will tell you when large changes are posted.



Reporting problems in the book

- At the bottom of each page, there is a link that allows to make an issue on the repo.



Then we can form the set

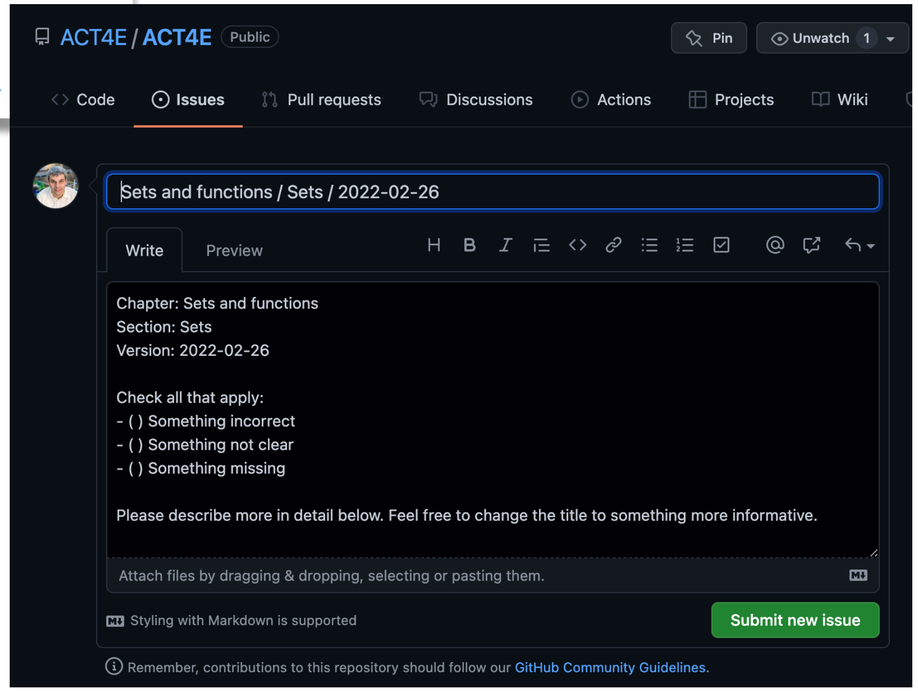
$$\{x \in \mathbf{B} \mid x \text{ is a fruit}\} = \{🍎, 🍇\}.$$

There is an interesting special case of this way of constructing subsets of a set $\mathbf{B}$: what if, for a given statement $S(x)$, *none* of the elements $x \in \mathbf{B}$ are such that $S(x)$ is true? Then the result is the empty set.

As an example, consider the statement

$$S(x) = \text{“}x \text{ is the name of a planet”}.$$

Something incorrect or unclear or missing? Report issues on GitHub.



- The repo ACT4E/ACT4E is public - you can fork and do pull requests.

Why are we here?

Compositionality in engineering

- Because the human brain has limited capacity,
  we **need to rely on abstractions and decomposition**.

- **We need to decompose systems into components.**
  What is a system? What are the components?

> **"A system is composed of components;**
> **a component is something you understand."**
>
> — Howard Aiken (1900-1973)

- What types of **decomposition** are there?
- What is **abstraction**, exactly?
- Can we talk about these things generally, without reference to a particular **domain**?

Applied Compositional Thinking

Applied Category Theory

What Applied Category Theory is for an engineer

- **You, as an engineer, have developed very good mental models** of systems and components in your domain.
  - Some models are formal, some models are "informal".
  - Some models are computational, some are not.

- **You use the more formal models to design, simulate, analyze your systems**.
  - For example: $\dot{x} = f(x)$

- The formal models also **help you communicate and share your ideas**.

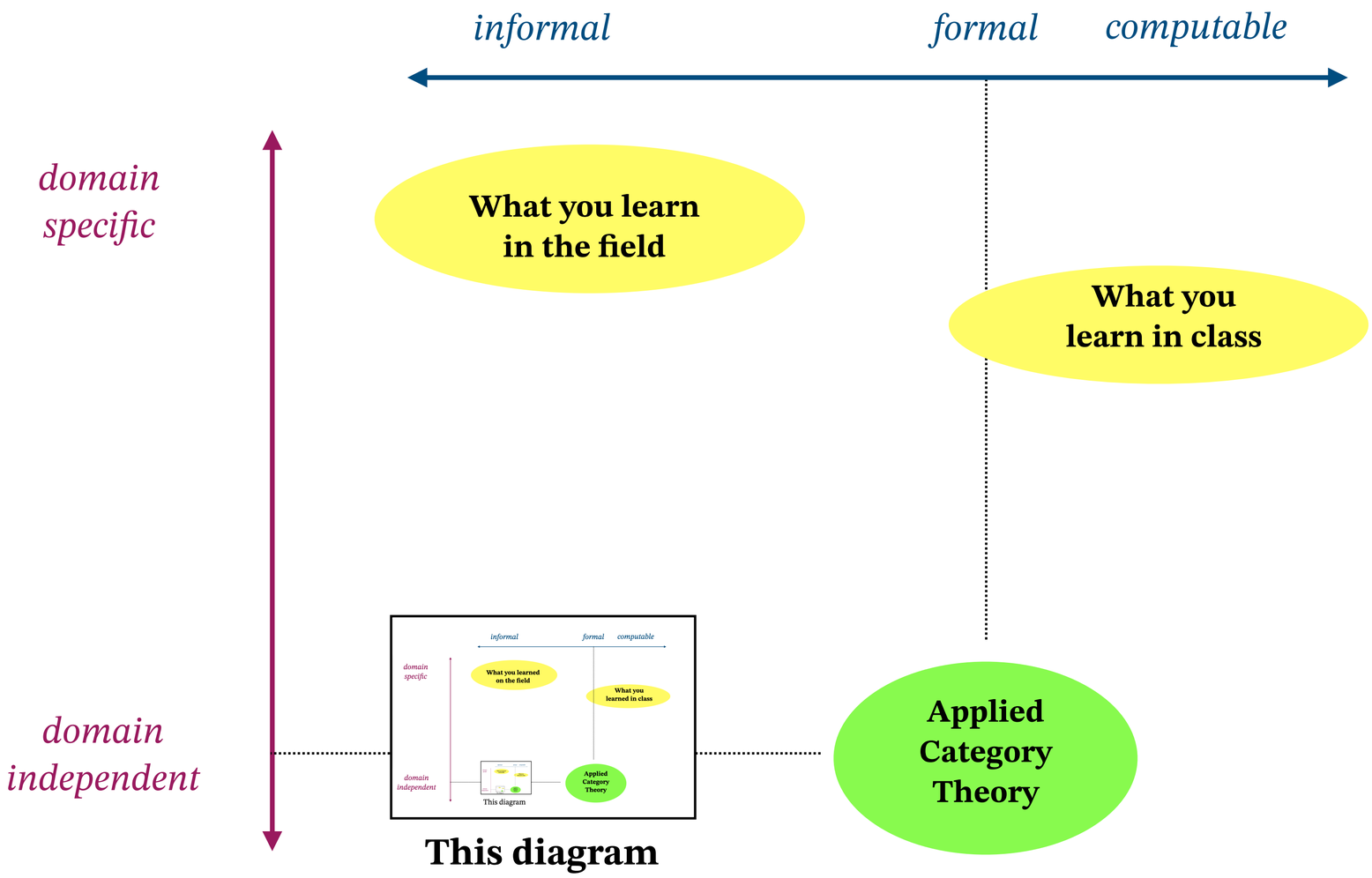
- Think of the **mathematics** you already know as **a set of patterns** to describe the world.

- Now, unfortunately you are an engineer **in the 21st century:**
  - The engineers before us solved all the easy problems!
  - The new problems to solve **have to deal with very complex systems.**

- **Applied Category Theory is an updated set of mathematical patterns that is particularly suited to describe large systems** with complex interaction/interconnection/decomposition features.

The case for Applied Category Theory

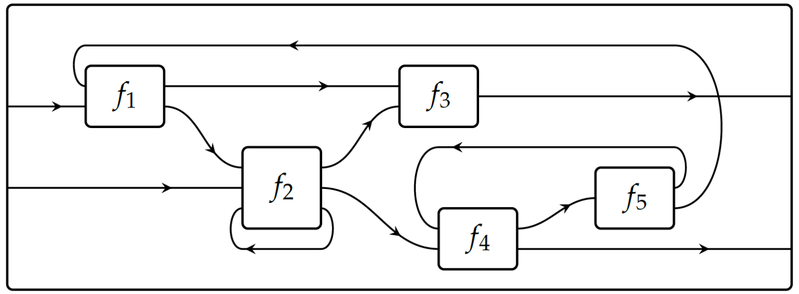
- Applied Category Theory is **formal** but **domain-independent**.



The case for Applied Category Theory

- **What can Applied Category Theory do for you:**
  - It is a **mathematical theory** equipped with theorems, methods, and **"libraries" of re-usable concepts/connections.**
  - It s a **formal and visual language** for reasoning/thinking and design.

- **Example**: These **"boxes and wires" diagrams** (representing something called "*traced monoidal categories*") have wildly different semantics according to the domain.

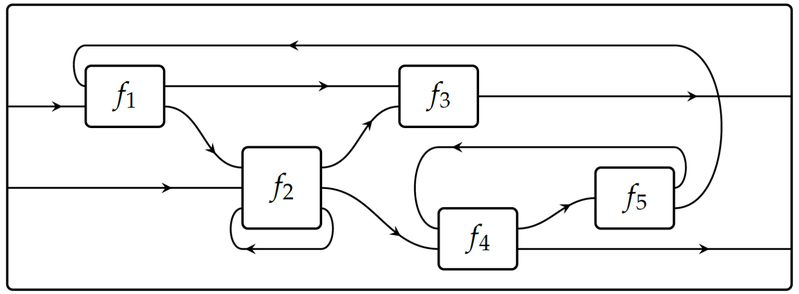


  - However: **in many cases, the rules about composition are always the same**.
  - We can study the concept and its implications abstractly, then re-particularize the conclusions to the domain of interest.

The case for Applied Category Theory

- **What Applied Category Theory cannot do:**
  - It cannot (yet), by itself, solve domain specific problems.
  - However, it can help you *formulate* the problems well.

- **The catch:**
  - You will need to **invest *more* work at the beginning** in the definitions, formalization, and modeling choices for your problem, but…
  - **The payoff comes later** in that things "work out" and that useful properties can be proven easily/quickly (they come for "cheap" or "free").



Obstacles to learning ACT as engineers

- While it is not particularly difficult, the **math can be unfamiliar** to engineers.
  - Engineering background is heavy on continuous math (analysis, ...), very light on discrete mathematics (algebra, ...) that are prerequisites for applied category theory.

- The **materials** you can currently find are **not particularly tailored for engineers**.
  - Typical audiences:

    Mathematicians

    Computer scientists

    Software engineers

- **No in-depth discussion of actual case studies** relevant to engineers.

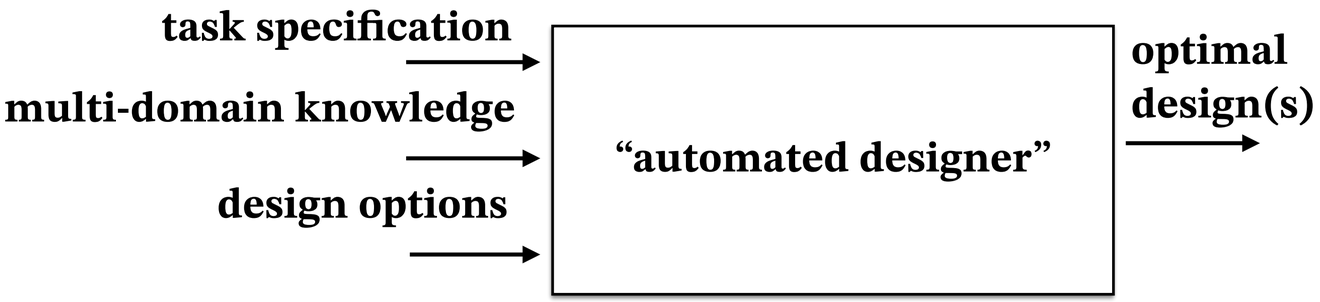
- The typical **curriculum** is at the same time **too broad** and **not broad enough**.
  - There are basic topics in ACT that engineers do not need.
  - There are topics that mathematicians consider very niche that have strong appeal for engineers.

- **We started the class to fix the situation.**

The vision for systems co-design

minimize
(resources usage)

subject to
(functionality constraints)



The vision for systems co-design

minimize
(resources usage)

subject to
(functionality constraints)

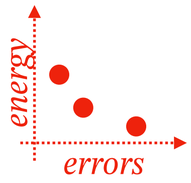
task specification →
multi-domain knowledge →
design options →

"automated designer"

→ **optimal design(s)**

Autonomy co-design

task →
robot autonomy, physics →
components, algorithms →



→

The vision for systems co-design

minimize
(resources usage)

subject to
(functionality constraints)

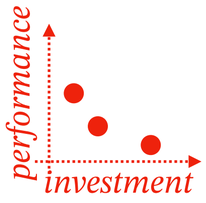
task specification →
multi-domain knowledge →
design options →

"automated designer"

→ **optimal design(s)**

Mobility co-design

demand →
networks, operations, infrastructure →
mobility services, policies →



Autonomy as the frontier of complexity for the co-design of complex systems

A fleet of autonomous robots

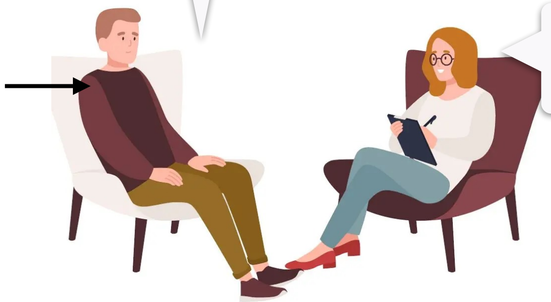
= **actuation** **hardware** **software** **behavior** **coordination** **localization** **planning** **sensing** **interaction** **liability** **control** **computation** **mapping** **learning** **perception** **energetics** **communication** **regulation** …

So many **components** (hardware, software, …),

We forget why we made some **choices**, and we are afraid to make **changes**…

These computers are not helping us that much when designing complex systems..

anthropomorphization of 21st century engineering malaise



"My dear, it's simple: you lack

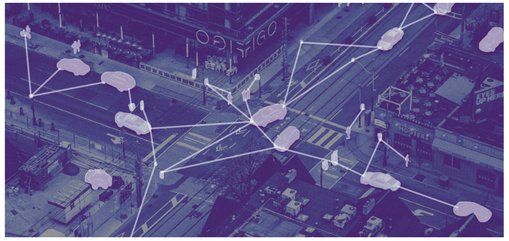
Co-design across fields and scales

"Your system is just a component in another person's system"

City level

Infrastructural choices

Service level

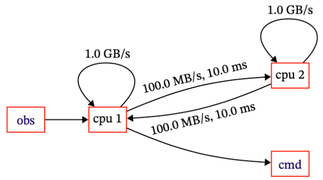


Deployment

Platform level

Autonomy stack

Subsystem level



Resource allocation

Mobility systems as a proxy for sociotechnical systems

“Co”-design challenges and desiderata

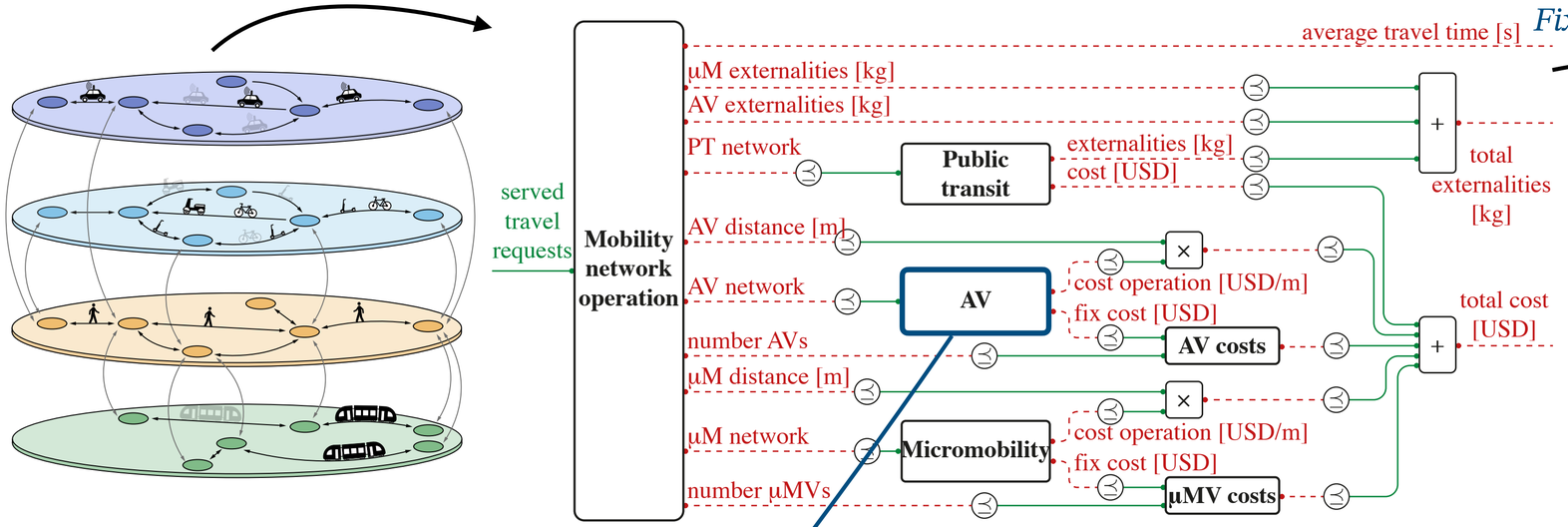
Complexity when designing complex systems

Large systems
- Many components
- Heterogeneous natures
- Multiple objectives

Strategic interactions
- Many agents
- Heterogeneous interactions
- Conflicts/collaborations

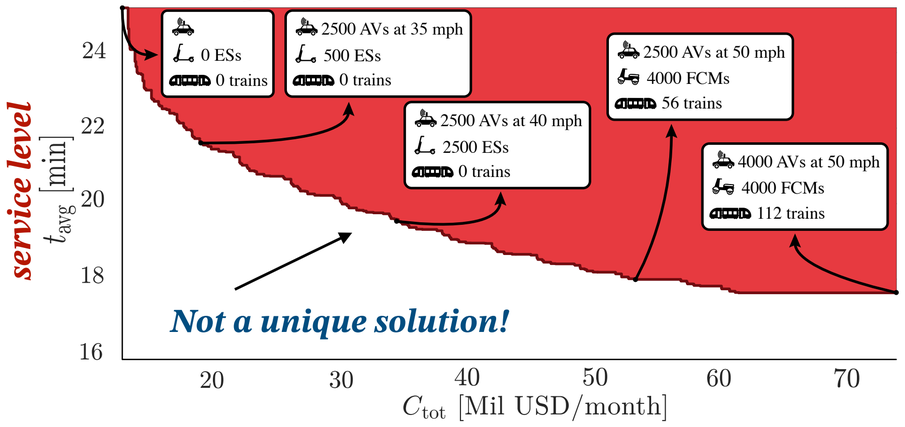
- **Computational**
  - Let the machine help us!
- **Compositional, hierarchical**
  - My system is a component of somebody else’s system.
- **Collaborative, intellectually tractable**
  - Pooling knowledge from experts across fields.
- **Continuous**
  - Design is not static: it should be reactive to changes in goals and contexts.

Co-design to enable user-friendly tools to assess the impact of future mobility solutions



Fix demand, minimize costs

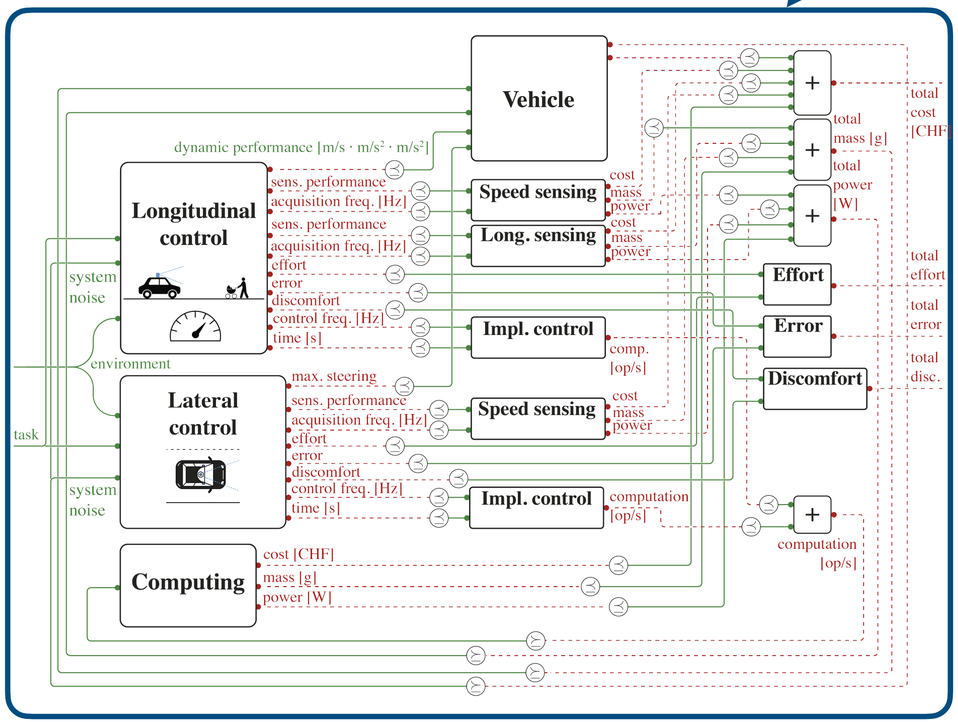
Washington D.C. in 2030



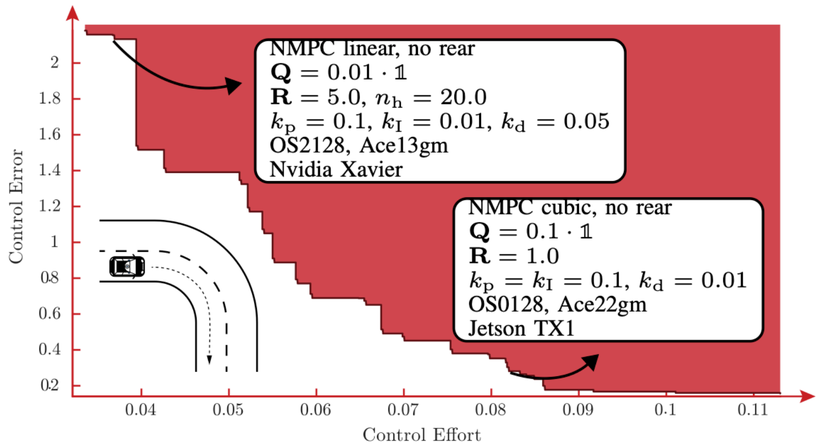
Not a unique solution!

investments

compositionality (vertical composition)



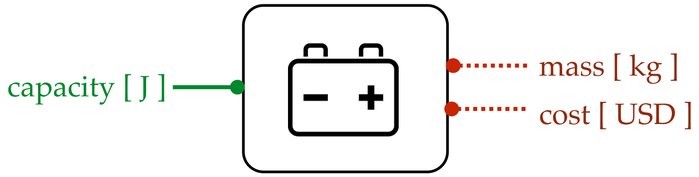
Fix environment, task



Details about software and hardware implementations, in a way that was not possible before

Co-design language

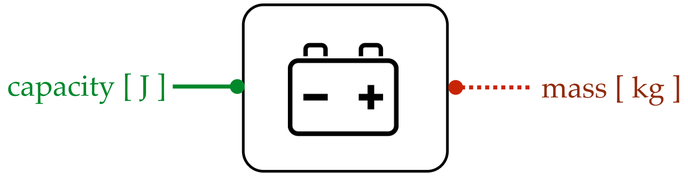
- "**Catalogues**": already available designs




```
catalogue {
    provides capacity [J]
    requires mass [g]
    requires cost [USD]

    500 kWh ⟷ model1 ⟼ 100 g, 10 USD
    600 kWh ⟷ model2 ⟼ 200 g, 200 USD
    600 kWh ⟷ model3 ⟼ 250 g, 150 USD
    700 kWh ⟷ model4 ⟼ 400 g, 400 USD
}
```

- "**First-principles**": analytical relations.




```
mcdp {
    provides capacity [J]
    requires mass [kg]

    specific_energy_Li_Ion = 500 Wh / kg

    required mass >= provided capacity / specific_energy_Li_Ion
}
```

```
mcdp {
    provides lift [N]
    requires power [W]
    c = 10.0 W/N²
    required power ≥ c · provided lift²
}
```

Overview of the syllabus

Overview of the syllabus

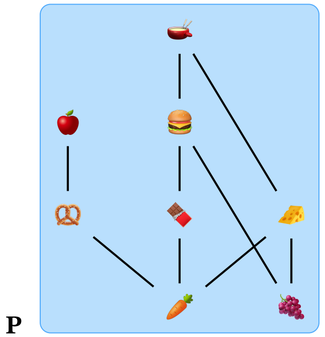
When	Day	Session	Topic	Homework
09/04	W	L1	*Introduction: Logistics, Motivation, Compositionality*	
09/06	F	R1		
09/09	M	L2	*Sets, functions, relations*	PS 1 out
09/11	W	L3	*Algebra*	
09/13	F	R2		
09/16	M	L4	*Trade-offs, Monotonicity*	PS 2 out
09/18	W	L5	*Categories*	
09/20	F		HOLIDAY – NO RECITATION	
09/23	M	L6	*Categorical constructions, constructing categories*	PS 3 out, PS 1 in
09/25	W	L7	*Actions*	
09/27	F	R3		PS 2 in
09/30	M	L8	*Functors*	PS 4 out
10/2	W	L9	*Co-design*	
10/4	F	R4		PS3 in
10/7	M	L10	*Natural transformations*	PS 5 out
10/9	W	L11	*Parallel composition (monoidal categories)*	
10/11	F	R5		PS4 in
10/14	M		HOLIDAY – NO CLASS	
10/16	W	L12	*Feedback (traced monoidal categories)*	
10/18	F	R6		PS 5 in
10/21	M		**Midterm**	

Overview of the syllabus

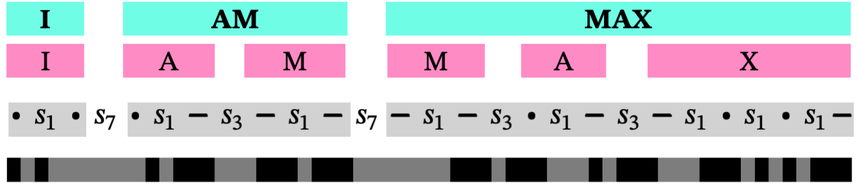
10/23	W	L13	*Modeling co-design problems, projects kick-off*	
10/25	F	R7		
10/28	M	L14	*Functorial computation*	PS 6 out
10/30	W	L15	*Monads*	
11/1	F	R8		
11/4	M	L16	*Universal constructions*	
11/6	W	L17	*Midterm presentations for projects*	
11/8	F	R9		PS6 in
11/11	M		HOLIDAY – NO CLASS	
11/13	W	L18	*Enriched category theory*	
11/15	F	R10		
11/18	M	L19	*Negative category theory*	
11/20	W	L20	*Special topics in applied category theory*	
11/22	F	R11		
11/25	M	L21	*Special topics in applied category theory*	
11/27	W	L22	*Special topics in applied category theory*	
11/29	F	R12		
12/2	M	L23	*Special topics in applied category theory*	
12/4	W	L24	*Final Presentations*	
12/6	F	R13	*Final Presentations*	

Syllabus overview / 1

- We recall some **simple algebraic structures** that you might have seen elsewhere:
  - **Semigroups, Monoids, Groups, group actions**
  - **Partially ordered sets & monotone functions**
- We do this because then we will show how everything is a particular case of a category.



$$\log(a \cdot b) = \log(a) + \log(b)$$

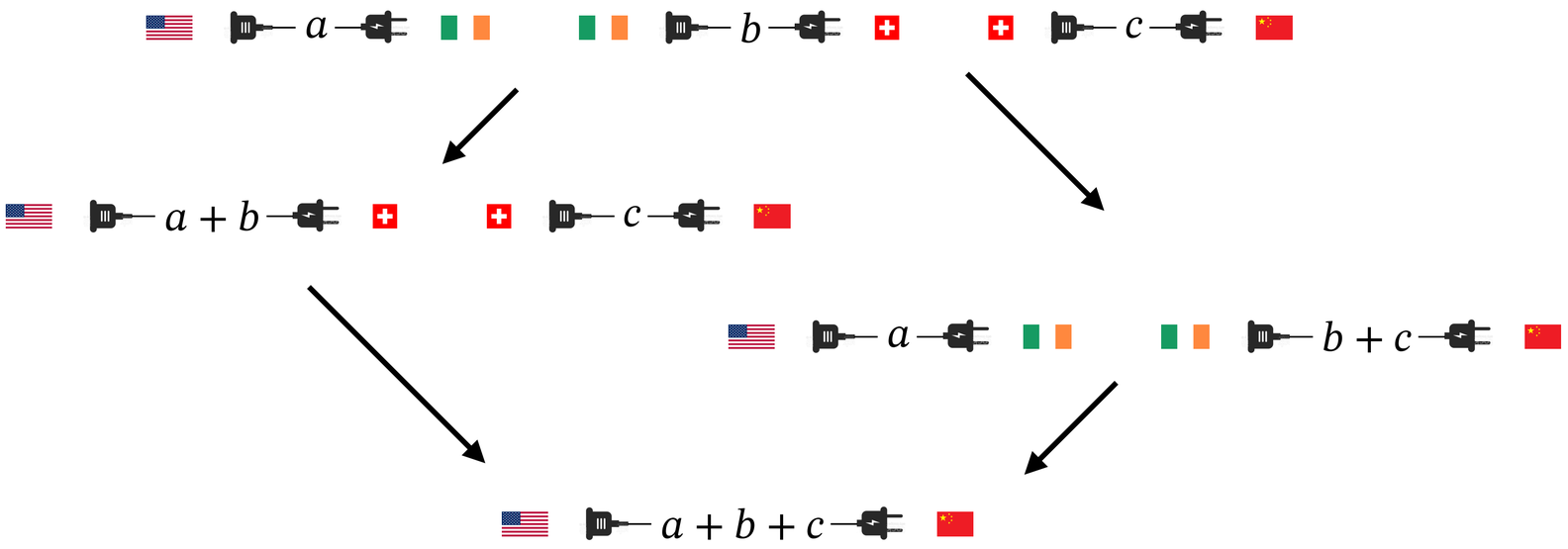


$$\begin{tikzcd}
\langle \mathbb{R}_{>0}, \cdot \rangle \arrow[loop left, "\log \fatsemi \exp = \mathrm{id}_{\mathbb{R}_{>0}}"] \arrow[r, bend left, "\log : \mathbb{R}_{>0} \to \mathbb{R}"] & \langle \mathbb{R}, + \rangle \arrow[loop right, "\exp \fatsemi \log = \mathrm{id}_{\mathbb{R}}"] \arrow[l, bend left, "\exp : \mathbb{R} \to \mathbb{R}_{>0}"]
\end{tikzcd}$$

$$\exp(c + d) = \exp(c) \cdot \exp(d)$$

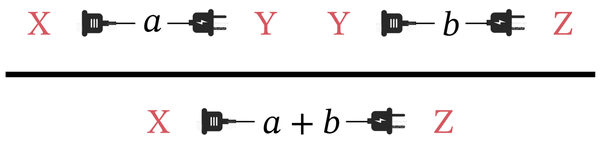
Syllabus overview / 2: categories

- We define **categories** as "associative compositional structures with interfaces".
- We talk about how to construct new categories from existing ones.

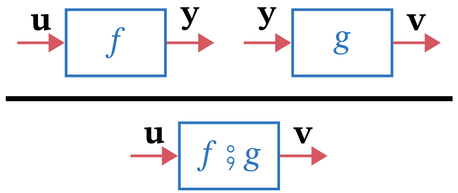


$$\frac{f : X \to Y \quad g : Y \to Z \quad h : Z \to U}{(f \fatsemi g) \fatsemi h = f \fatsemi (g \fatsemi h)}$$

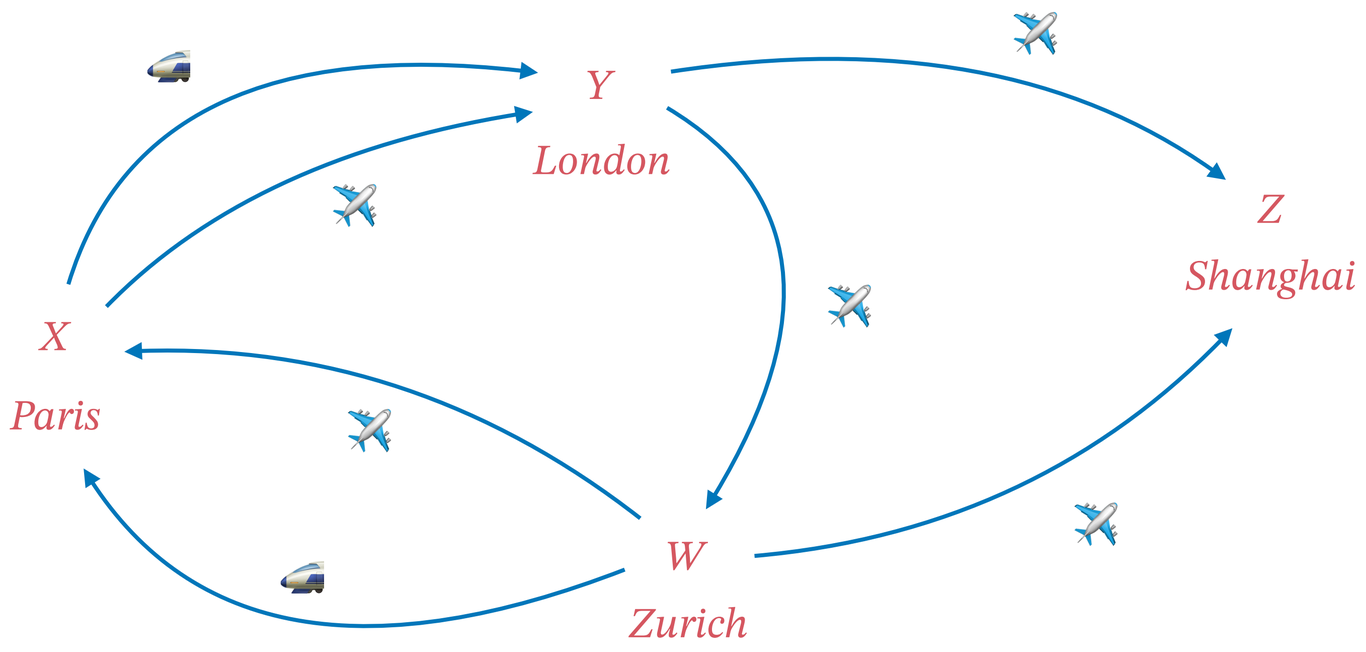
Syllabus overview / 2: categories

Electrical cords		*Sets and functions*
		$\dfrac{f : X \to Y \quad g : Y \to Z}{(f \fatsemi g) : X \to Z}$
Socket/plug shapes	***Objects***	Sets
Extension cords	*Arrows* ***Morphisms***	Functions
Plugging together	***Composition***	Function composition

Syllabus overview / 2: categories

Real matrices		*dynamical systems*
$\mathbb{R}^l \xrightarrow{\mathbf{A}} \mathbb{R}^m \quad \mathbb{R}^m \xrightarrow{\mathbf{B}} \mathbb{R}^n$ $\mathbb{R}^l \xrightarrow{\mathbf{B}\cdot\mathbf{A}} \mathbb{R}^n$		
$n \in \mathbb{N}$	***Objects***	Input/Output spaces
Real matrices $\mathbf{A} \in \mathbb{R}^{n\times m}$	*Arrows* ***Morphisms***	dynamical system
Matrix multiplication	***Composition***	Composing systems

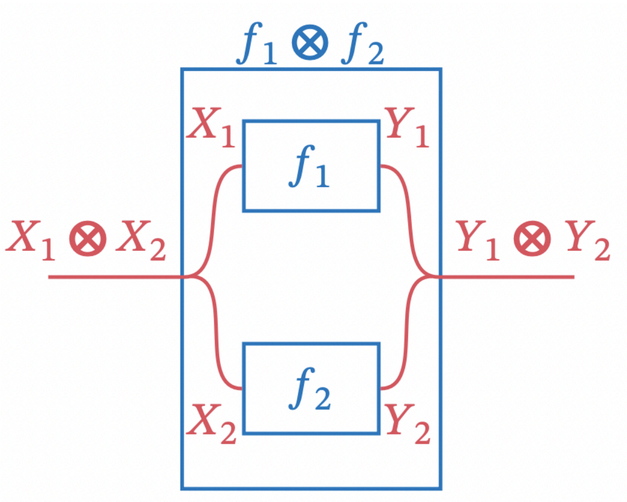
Syllabus overview / 2: categories

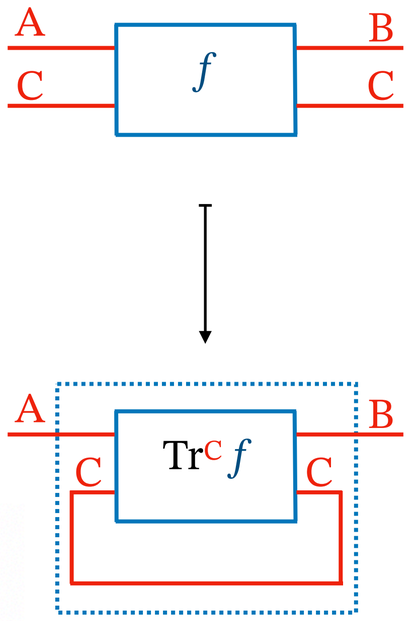


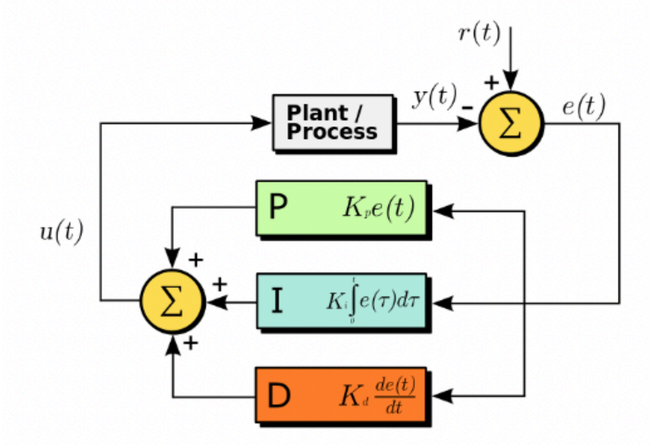
Objects	{London, Paris, Shanghai, Zurich}
Morphisms	All possible **paths** in the graph
Composition	Concatenation of paths in the graph

Syllabus overview / 3

- We talk about **categories with special structures**.
  - monoidal categories (**parallel** connection)
  - traced monoidal categories (**feedback** connections)
  - locally posetal categories (**and/or** connection)

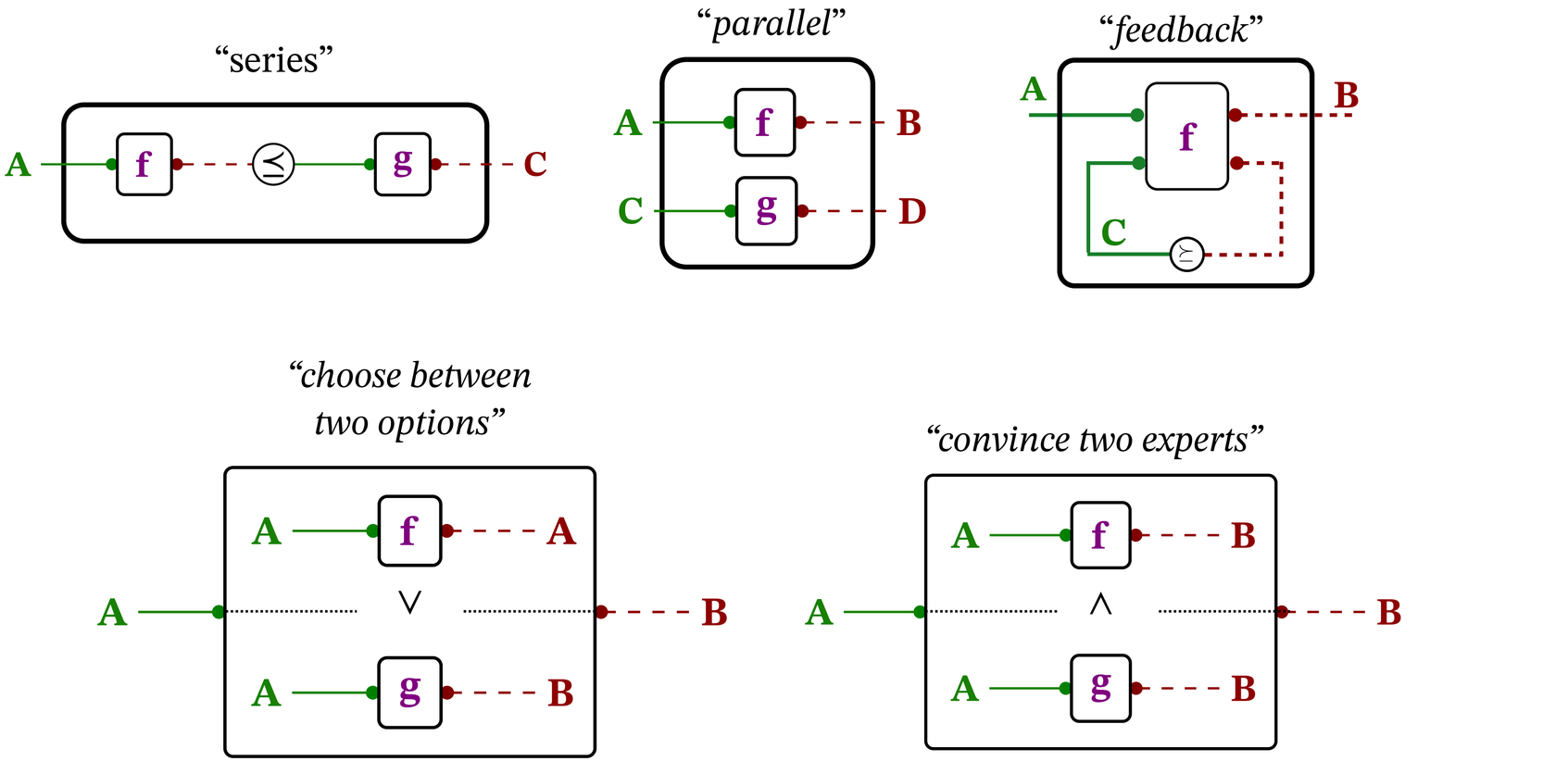


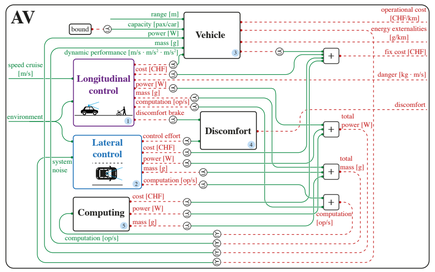




Syllabus overview / 4

- We talk about **monotone co-design problems** as an example of a rich category with more structure and more ways to compose morphisms.





Syllabus overview / 5

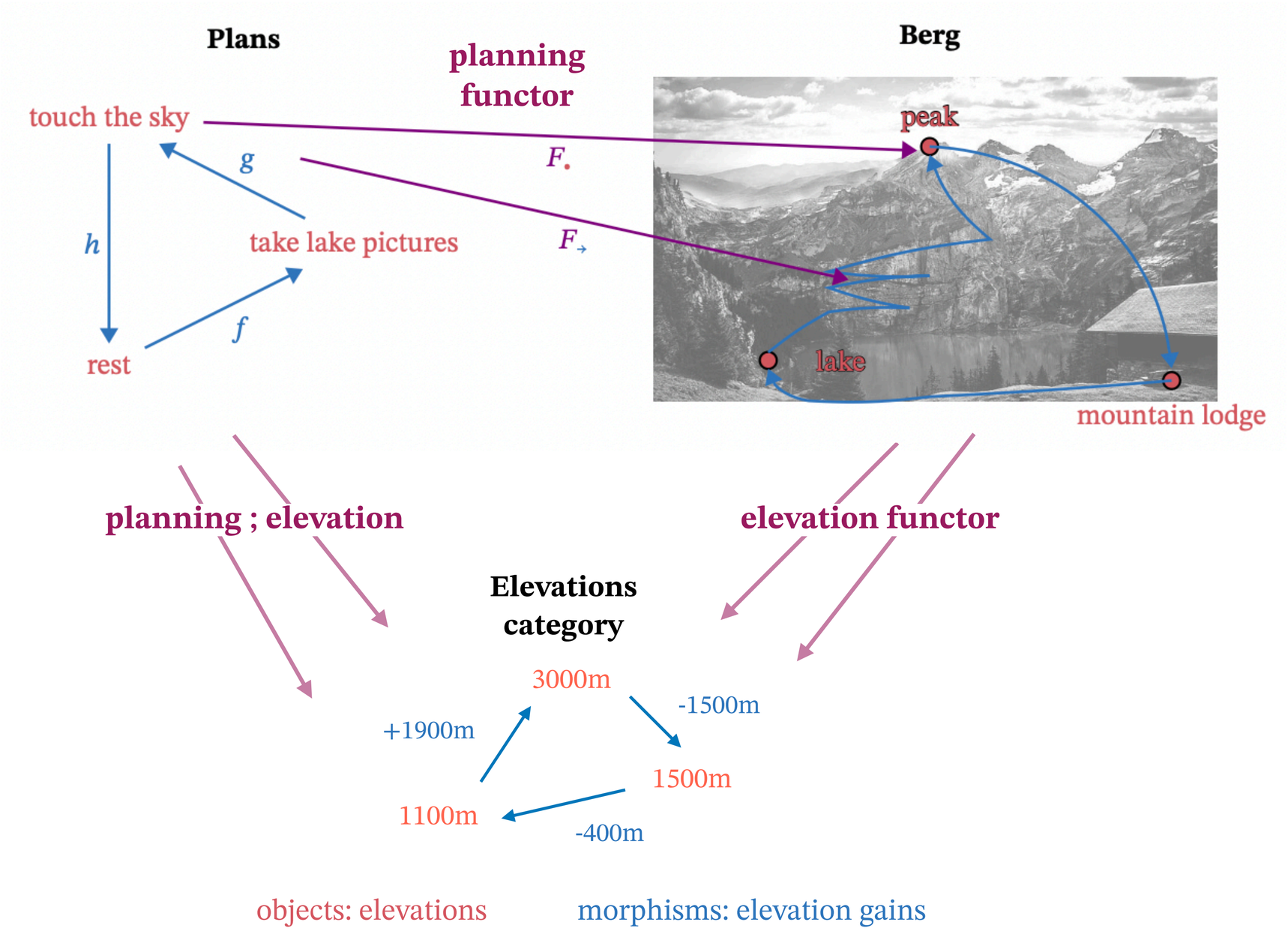
- We talk about **functors**: arrows between categories.
- Functors "**translate**" between concepts in different categories.

$$\frac{f : X \to_{\mathbf{C}} Y \qquad g : Y \to_{\mathbf{C}} Z}{F_{\to}(f \fatsemi_{\mathbf{C}} g) = F_{\to}(f) \fatsemi_{\mathbf{D}} F_{\to}(g)}$$

$$\begin{tikzcd}
 & Y \arrow[dr, "g"] & \\
X \arrow[ur, "f"] \arrow[rr, "f \fatsemi g"'] & & Z \\
\mathbf{C} & & \\
 & F_{\bullet}(Y) \arrow[dr, "F_{\to}(g)"] & \\
F_{\bullet}(X) \arrow[ur, "F_{\to}(f)"] \arrow[rr, "F_{\to}(f \fatsemi g)"'] & & F_{\bullet}(Z) \\
\mathbf{D} & &
\end{tikzcd}$$

$$X \xrightarrow{F} F_{\bullet}(X)$$

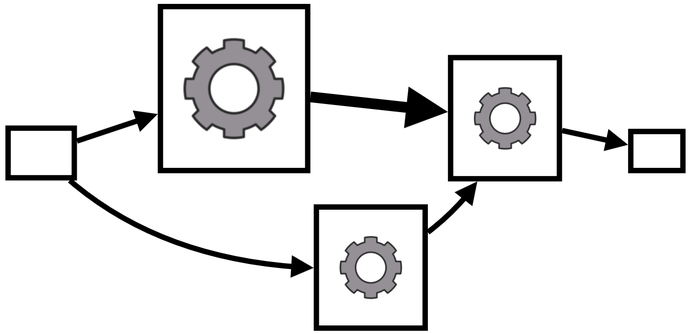
Functors: arrows between categories



Deployment of a computation graph to a resource graph

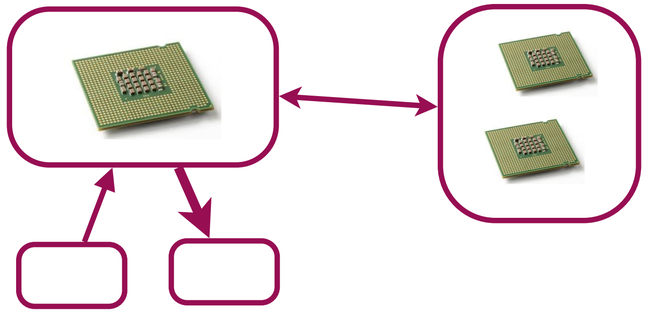
Computation Graph

- Nodes: **processes**
- Weight: **flops** required
- Edges: **signals**
- Weight: **bandwidth** needed

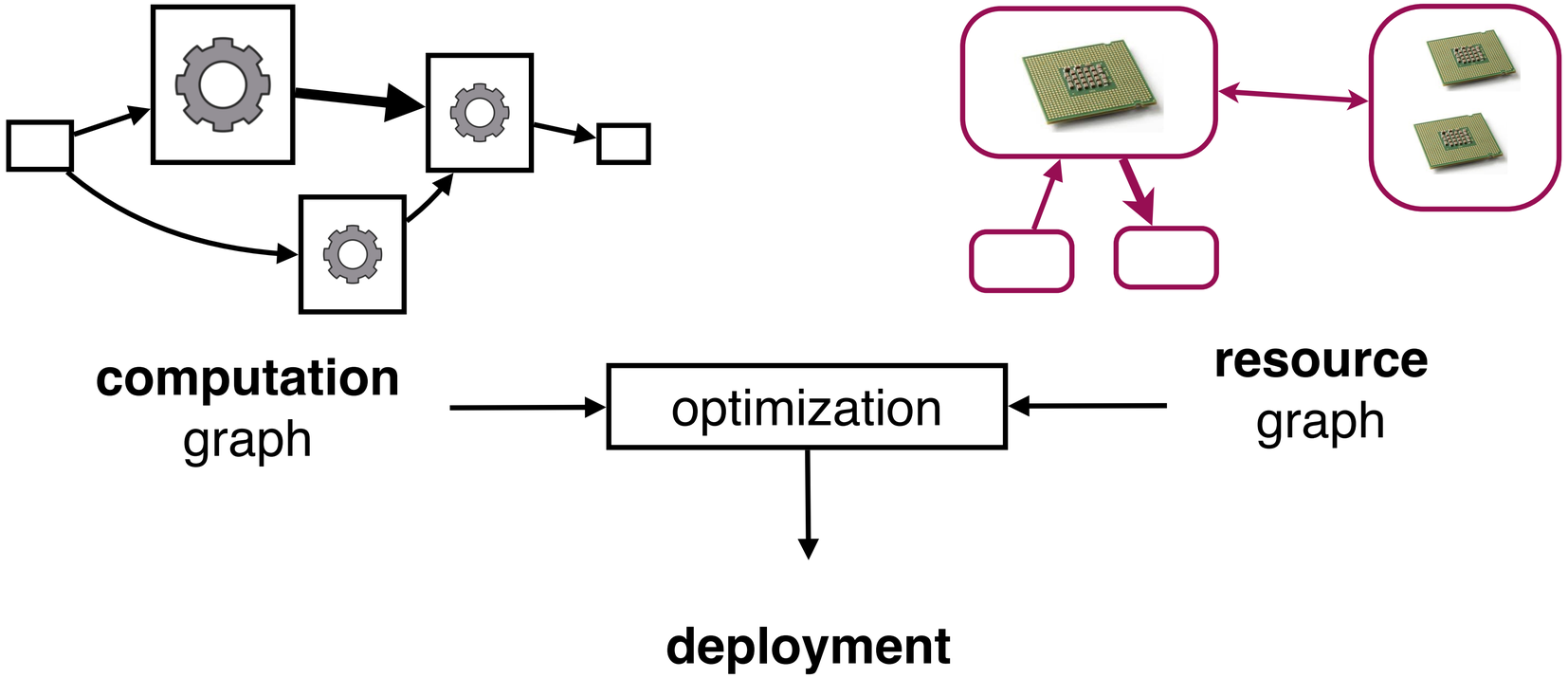


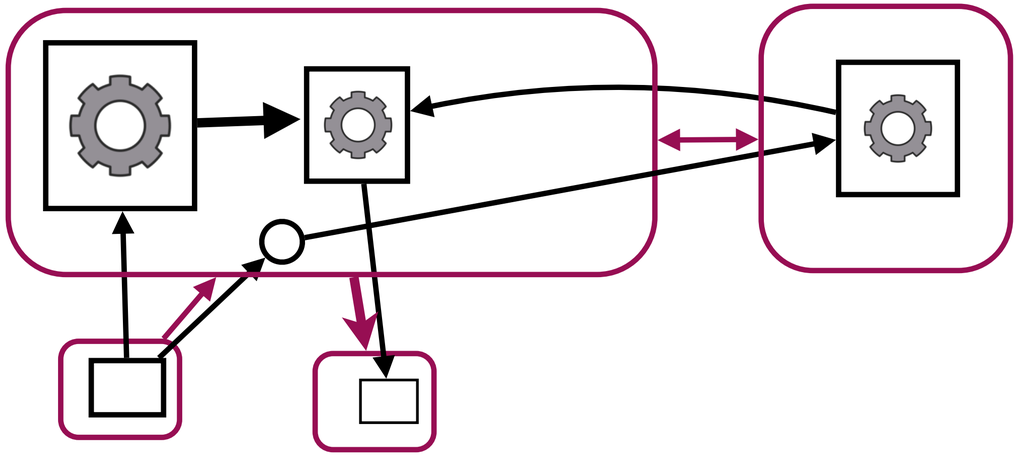
Resource Graph

- Nodes: **processors**
- Weight: **computation** available
- Edges: **network links**
- Weight: **bandwidth** available



Deployment of a computation graph to a resource graph





Functors in dynamical systems

laplace transform

LTI system represented by impulse response

composition = convolution

$$(f * g)(t) = \int_0^t f(\tau)g(t-\tau)\mathrm{d}\tau$$

inverse

LTI system represented by Laplace transform

composition = multiplication

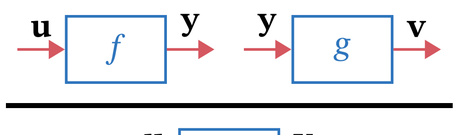
$$F(s) \cdot G(s)$$

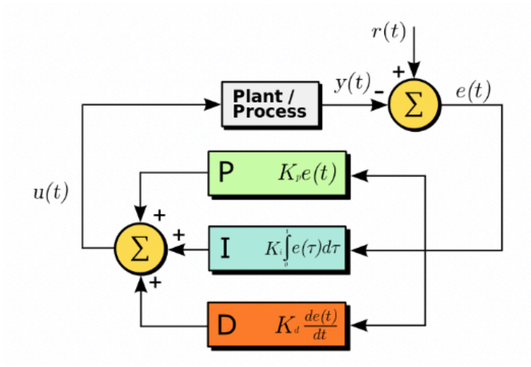
Common Laplace Transform Pairs

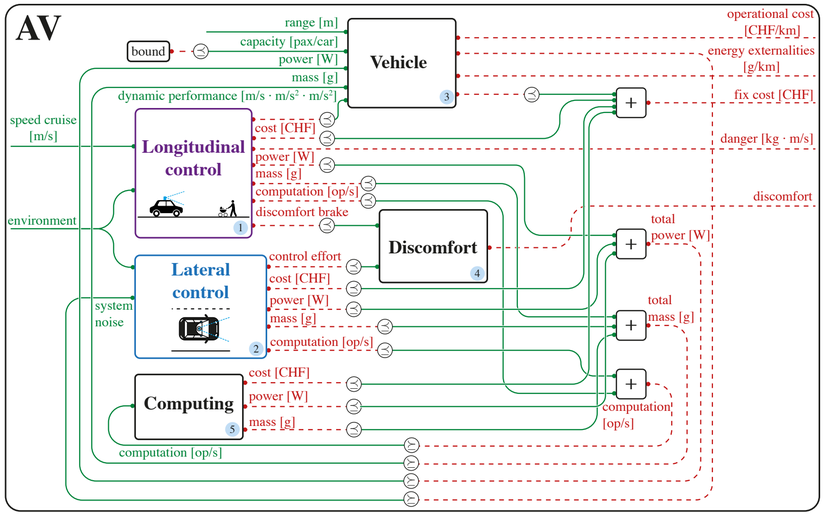
Time Domain Function		Laplace Domain Function
Name	Definition*	
Unit Impulse	$\delta(t)$	1
Unit Step	$\gamma(t)$ †	$\frac{1}{s}$
Unit Ramp	t	$\frac{1}{s^2}$
Parabola	t^2	$\frac{2}{s^3}$
Exponential	e^{-at}	$\frac{1}{s+a}$
Asymptotic Exponential	$\frac{1}{a}(1-e^{-at})$	$\frac{1}{s(s+a)}$
Dual Exponential	$\frac{1}{b-a}\left(e^{-at}-e^{-bt}\right)$	$\frac{1}{(s+a)(s+b)}$
Asymptotic Dual Exponential	$\frac{1}{ab}\left[1+\frac{1}{a-b}\left(be^{-at}-ae^{-bt}\right)\right]$	$\frac{1}{s(s+a)(s+b)}$
Time multiplied Exponential	te^{-at}	$\frac{1}{(s+a)^2}$
Sine	$\sin(\omega_0 t)$	$\frac{\omega_0}{s^2+\omega_0^2}$

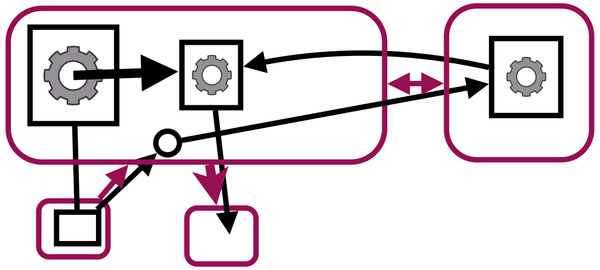
Syllabus overview / 5

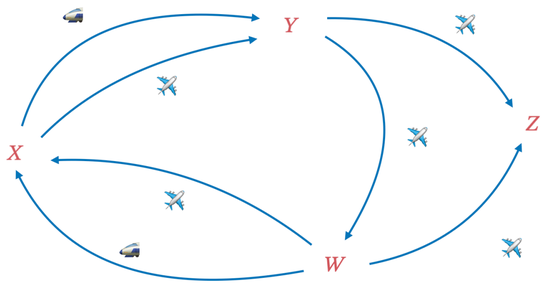
- Most examples are centered in these areas:
  - **dynamical systems** and similar widgets which have a time evolution
  - **monotone co-design problems** (our research in systems optimization)
  - **robotics** & **mobility**









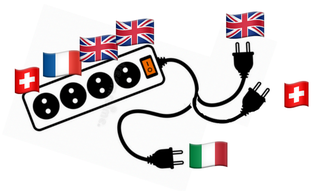


Beyond the basics

- ACT4E II, taught in the Fall, explores beyond the basics:
  1. **Generalizations**: structures of which categories are special cases;
  2. **"Up the ladder of abstraction"**: from functors to natural transformations, etc.
  3. **Richer structure**: categories with more constructors and properties;

basic category theory

generalization



richer structure

linear logic
nategories
...

up the ladder of abstraction

$$\begin{tikzcd} \mathbf{C} \arrow[r, bend left=60, "H"] \arrow[r, bend right=60, "F"'] \arrow[r, "\alpha"] & \mathbf{D} \arrow[r, "G"] & \mathbf{E} \end{tikzcd}$$

Example of generalizations: polycategories

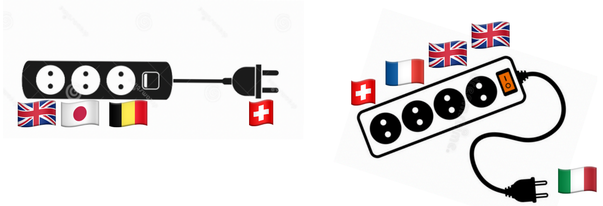
- **Polycategories** are a generalization of multicategories where each morphism has **zero or more source objects** and **zero or more target objects**.

Monoids
(Category with one object)

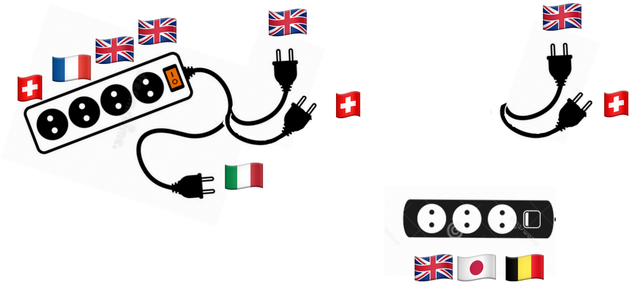
Categories

Multicategory with one object
(operad)

Multicategories



Polycategories

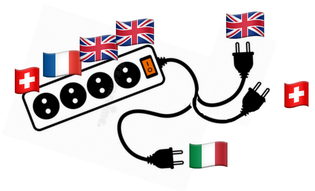


Beyond the basics

- ACT4E II, taught in the Fall, explores beyond the basics:
  1. **Generalizations**: structures of which categories are special cases;
  2. **"Up the ladder of abstraction"**: from functors to natural transformations, etc.
  3. **Richer structure**: categories with more constructors and properties;

basic category theory

generalization



richer structure

linear logic
nategories
...

up the ladder of abstraction

$$\begin{tikzcd} \mathbf{C} \arrow[r, bend left=60, "H"] \arrow[r, bend right=60, "F"'] \arrow[r, "\alpha"] & \mathbf{D} \arrow[r, "G"] & \mathbf{E} \end{tikzcd}$$

Composition

Oct. 24, 1961 G. K. CHRISTIANSEN 3,005,282

TOY BUILDING BRICK

Filed July 28, 1958 2 Sheets-Sheet 1

FIG. 1.

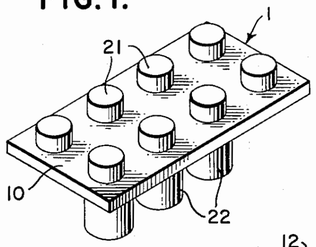


FIG. 2.

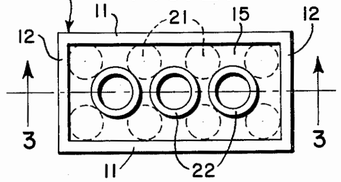


FIG. 3.

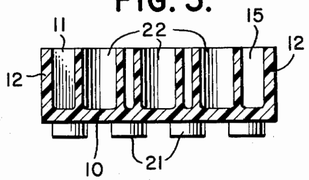


FIG. 4.

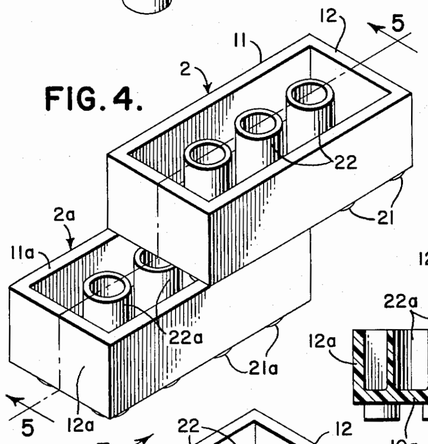


FIG. 5.

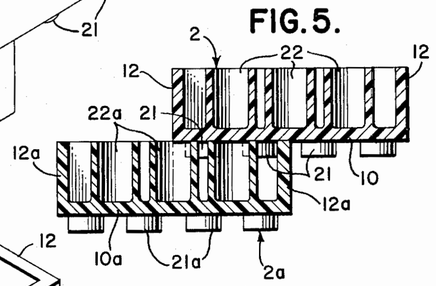
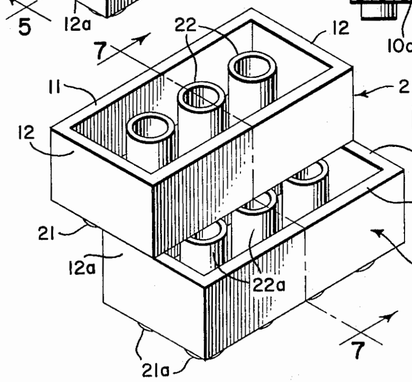


FIG. 6.



INVENTOR

Godtfred Kirk Christiansen

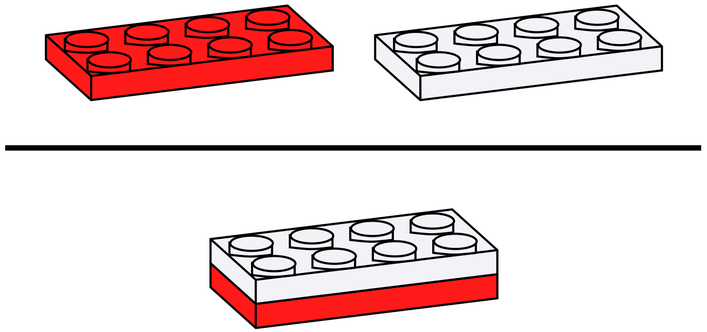
BY Stevens, Davis, Miller & Mosher
ATTORNEYS

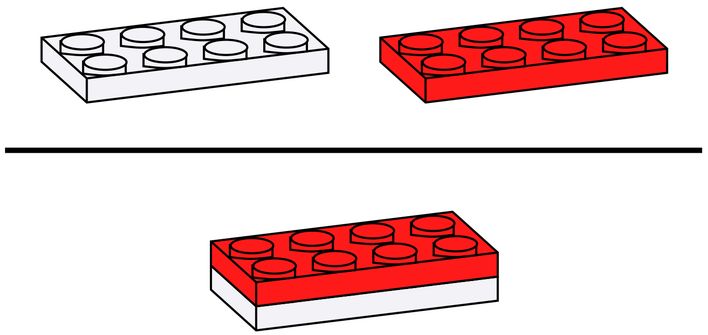
Composition recipes

ingredient ingredient ingredient

---

result result



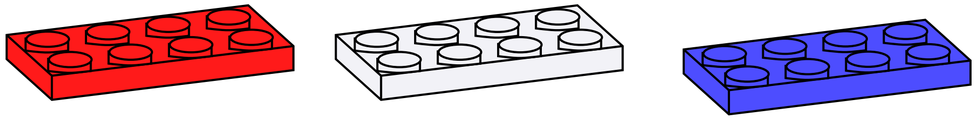


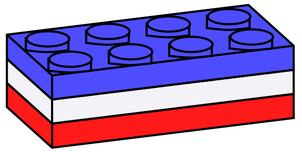
Composition recipes

ingredient ingredient ingredient

---

result result



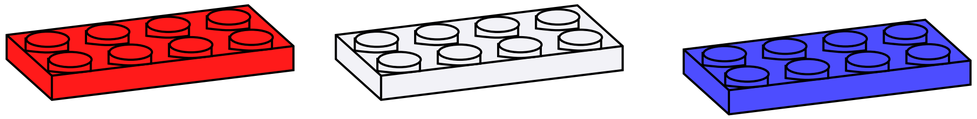


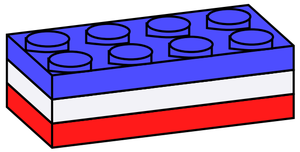
Composition recipes

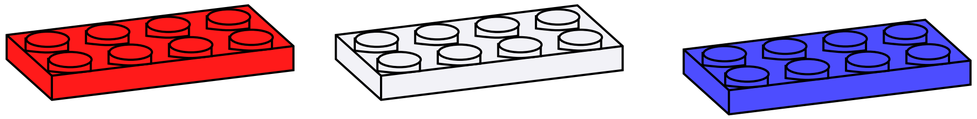
ingredient ingredient ingredient

---

result result





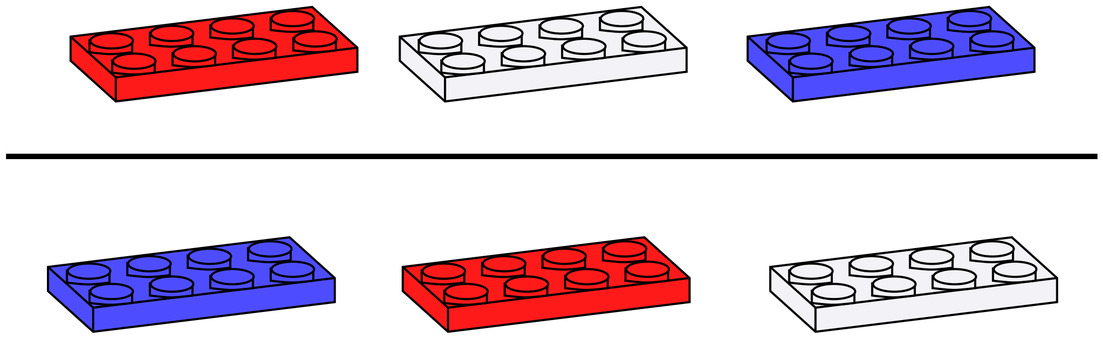


Composition recipes

ingredient ingredient ingredient

---

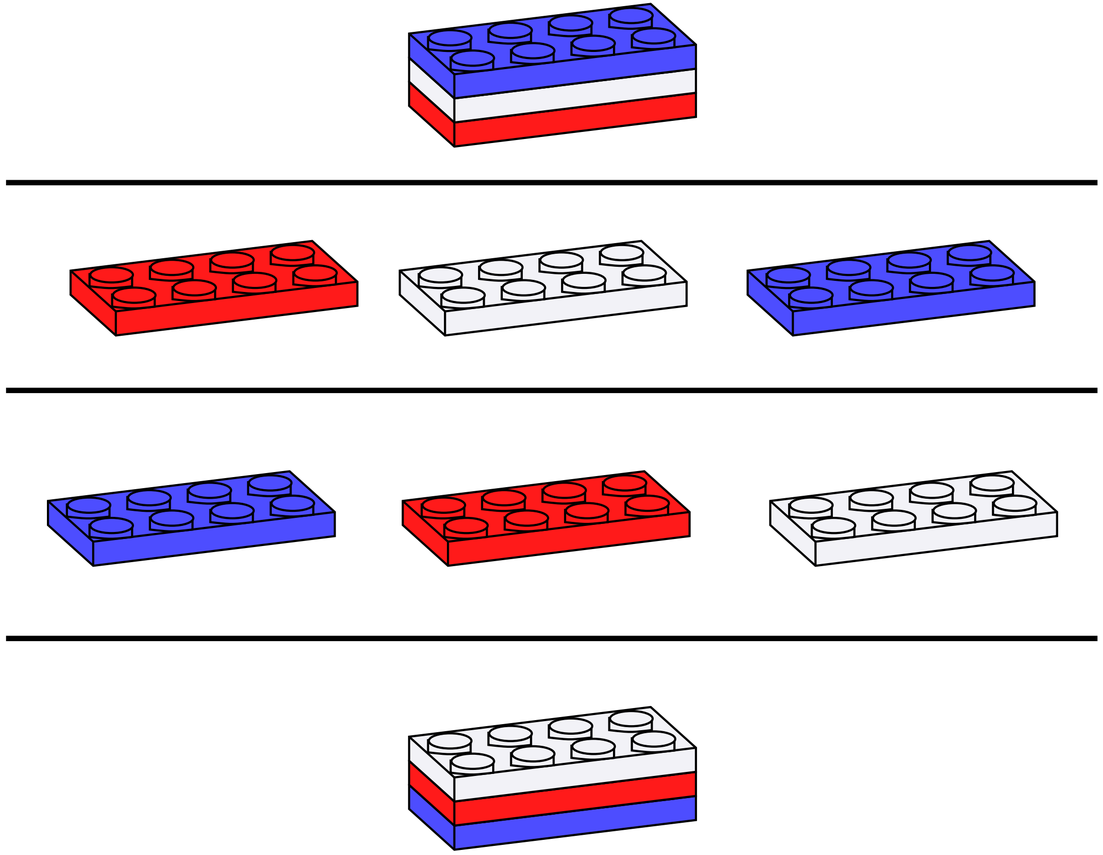
result result



Composition recipes

ingredient ingredient ingredient

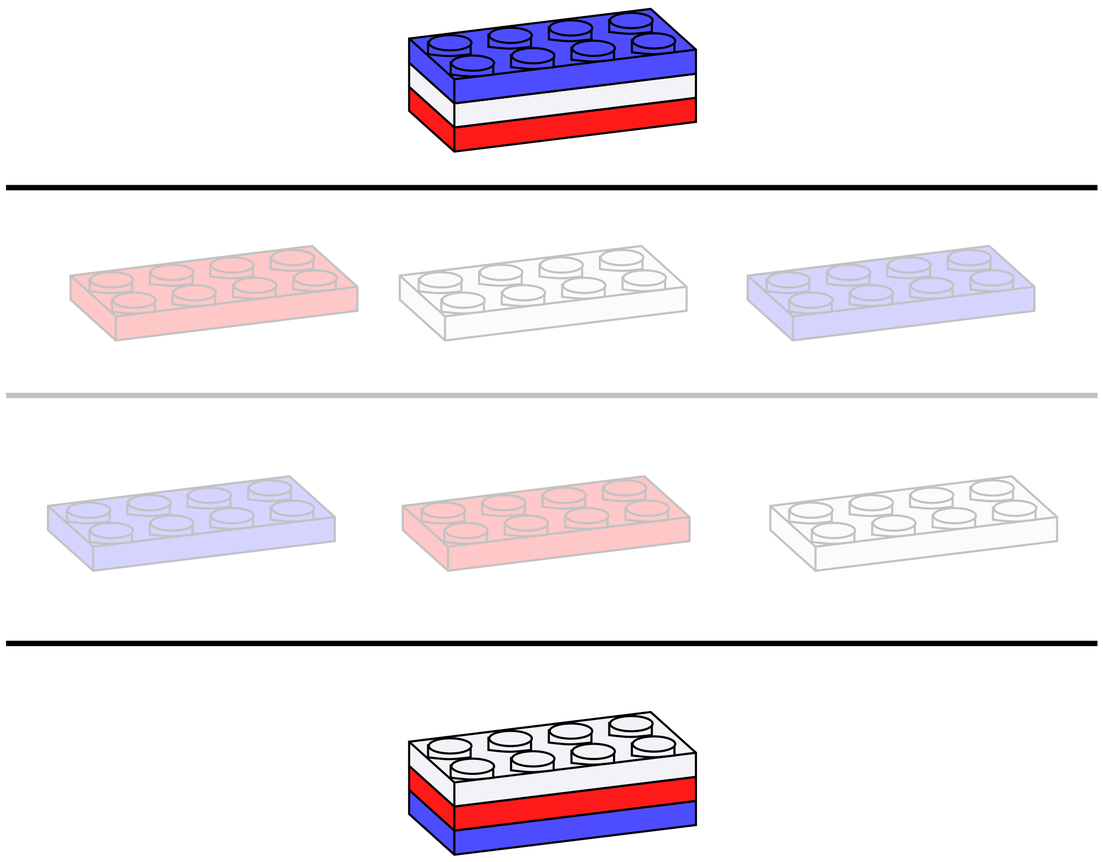
result result



Composition recipes

ingredient ingredient ingredient

result result

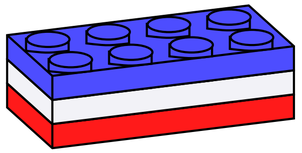


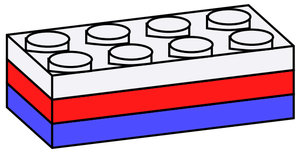
Composition recipes

ingredient ingredient ingredient

---

result result



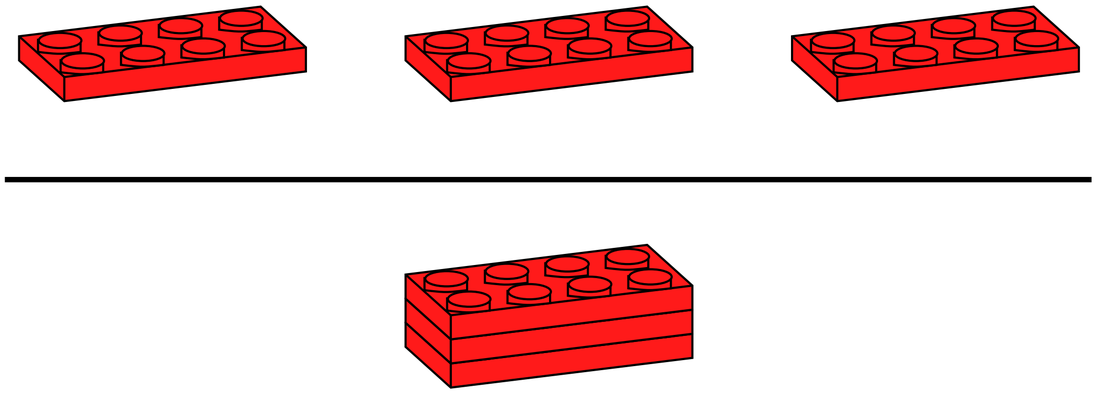


Composition recipes

ingredient ingredient ingredient

---

result result

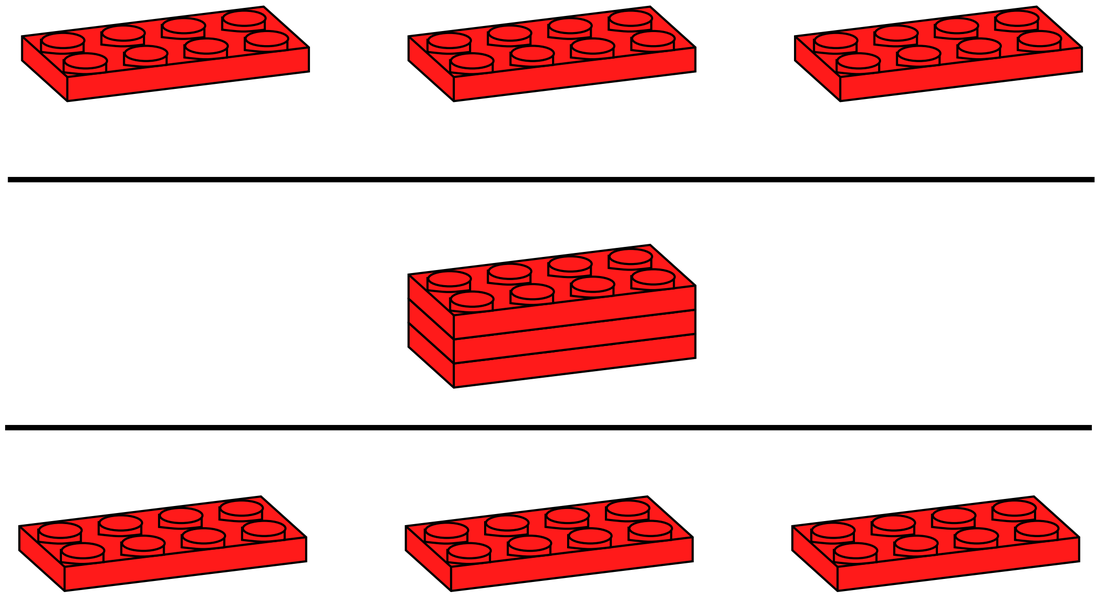


Composition recipes

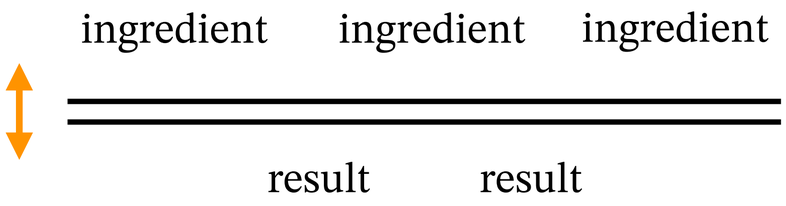
ingredient ingredient ingredient

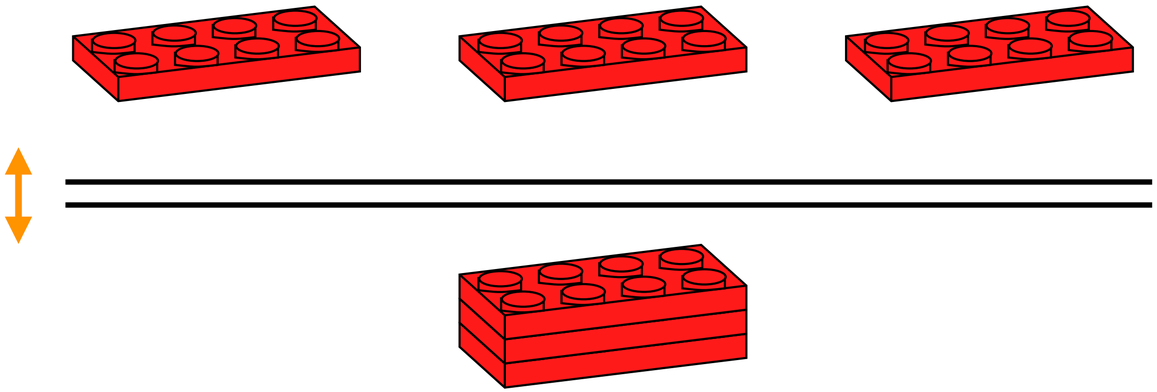
---

result result



Composition recipes



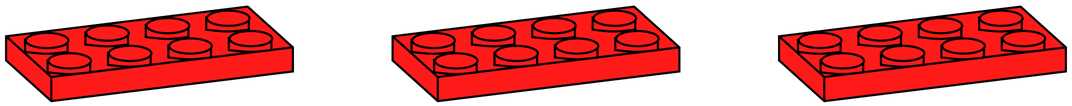


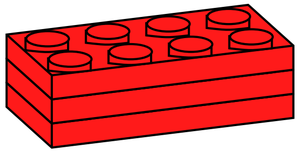
Composition recipes

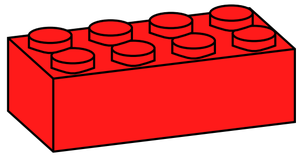
ingredient ingredient ingredient

---

result result





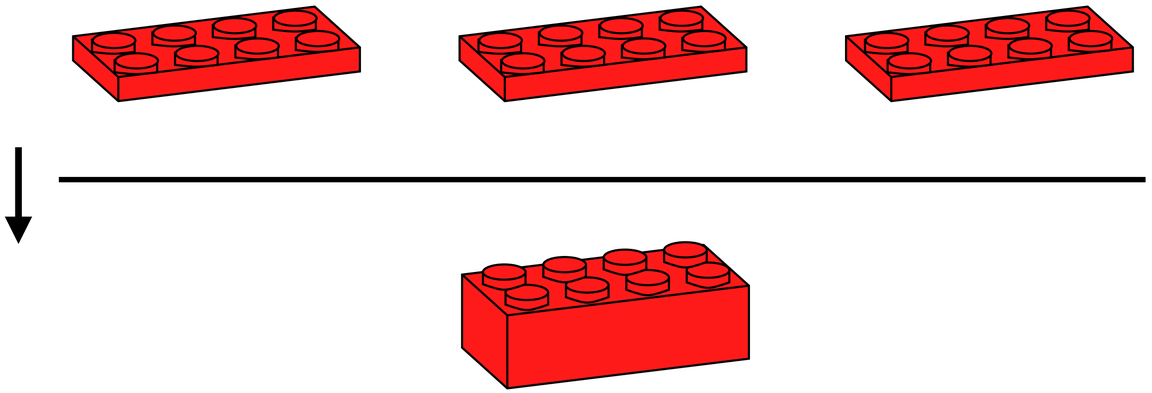


Composition recipes

ingredient ingredient ingredient

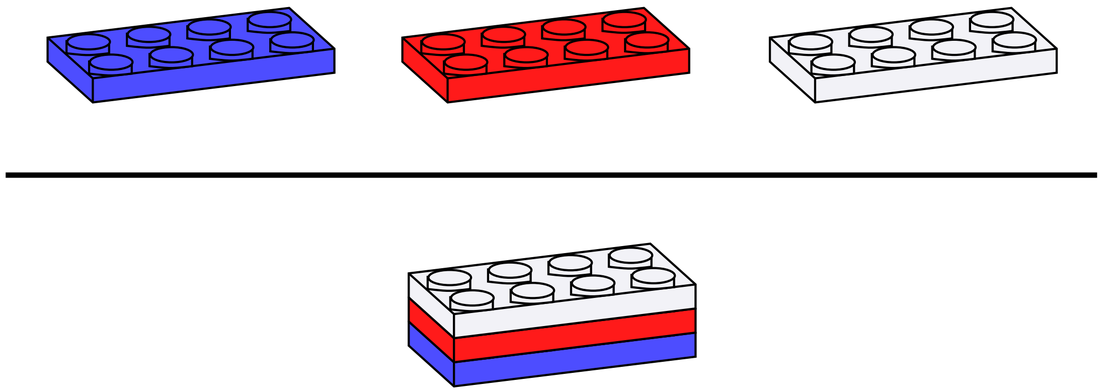
---

result result



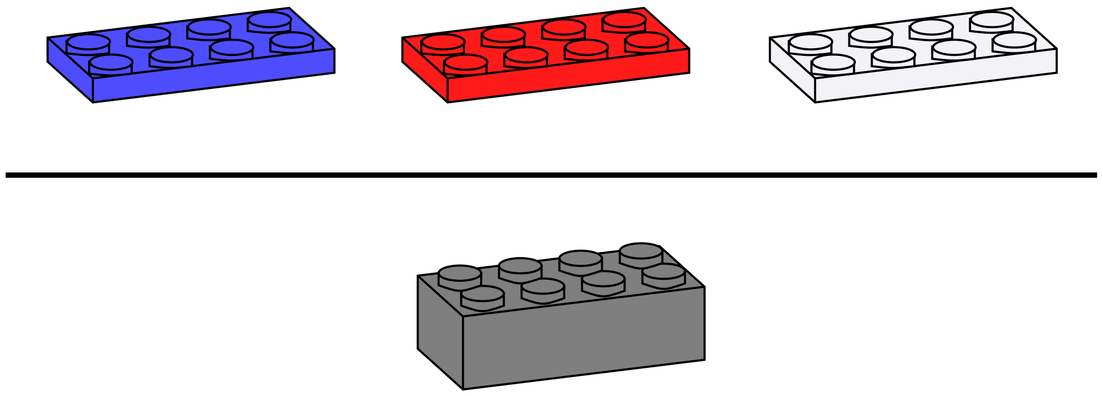
Lego purists

- Lego purists don't like mixing colors!

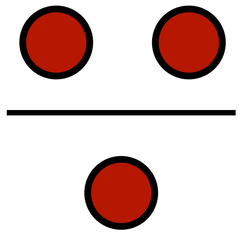


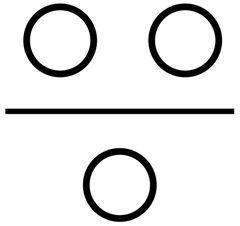
Lego purists

- Lego purists don’t like mixing colors!

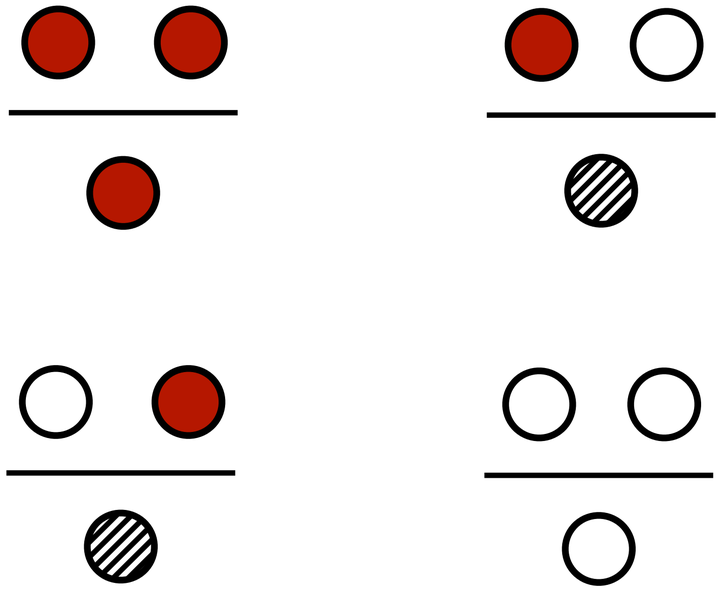


Color mixing for Lego purists

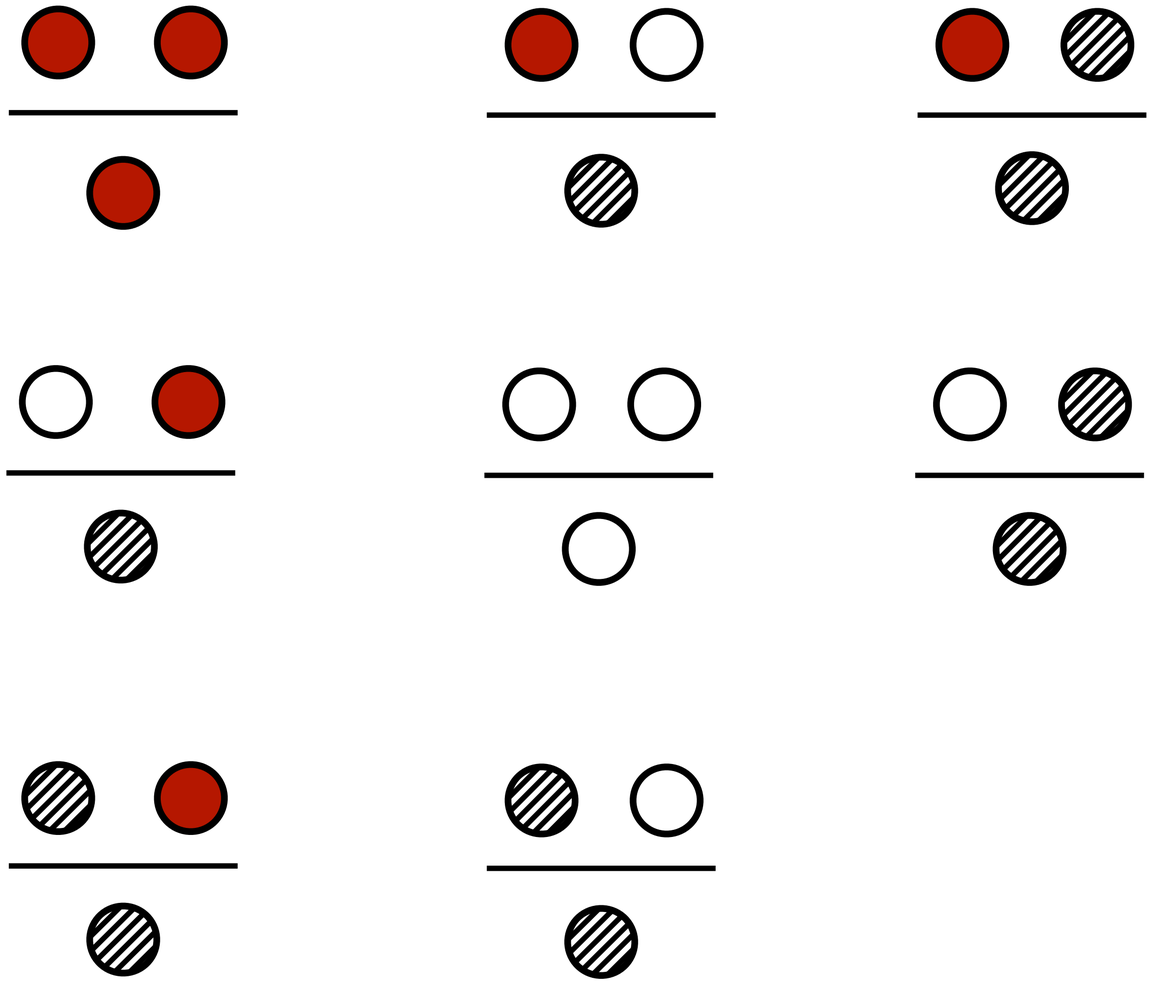




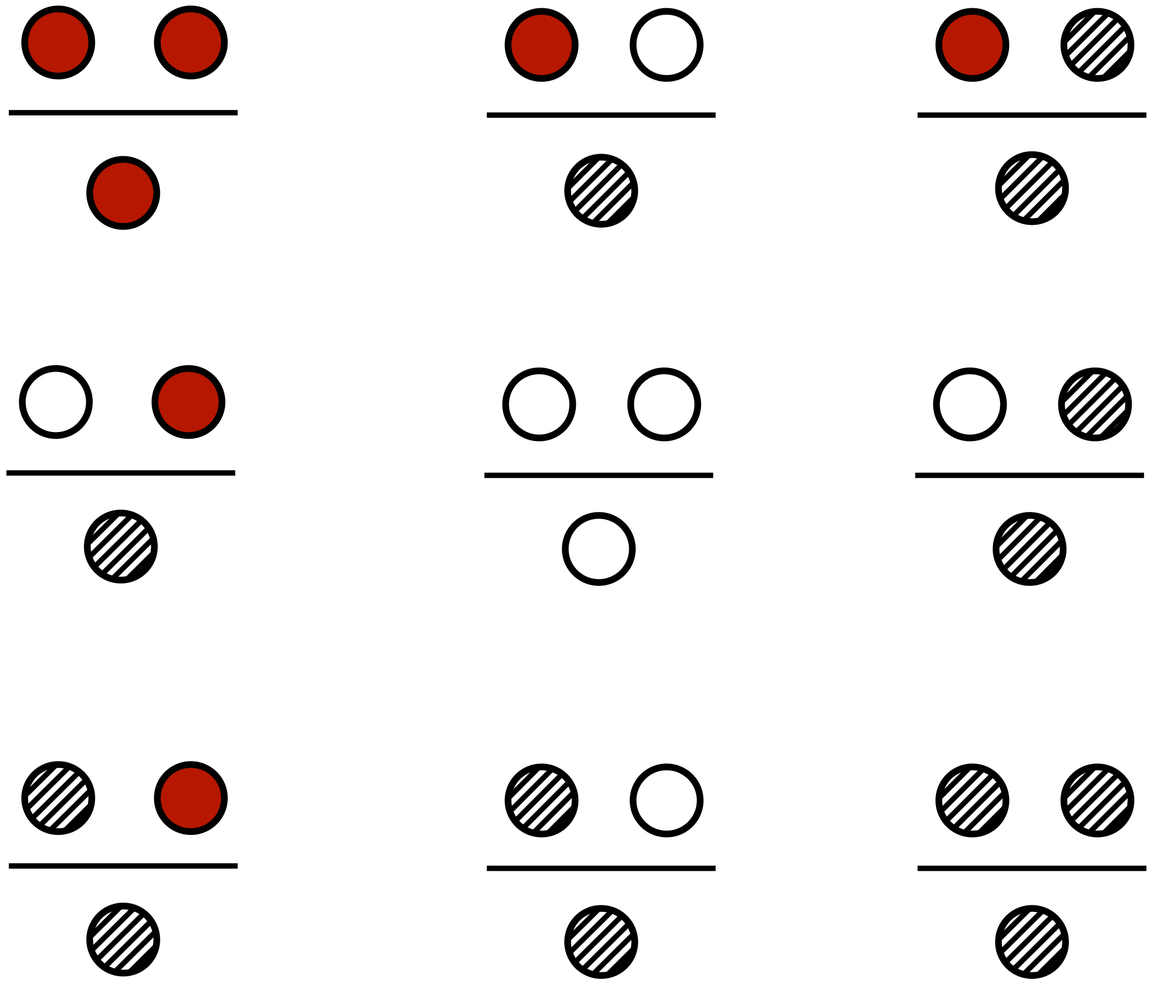
Color mixing for Lego purists



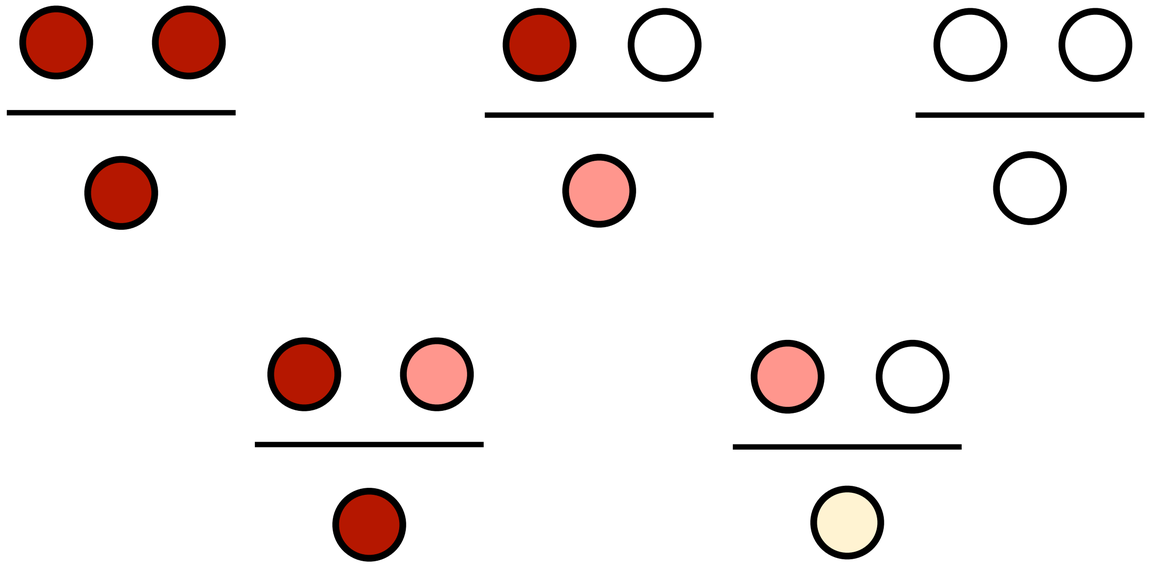
Color mixing for Lego purists



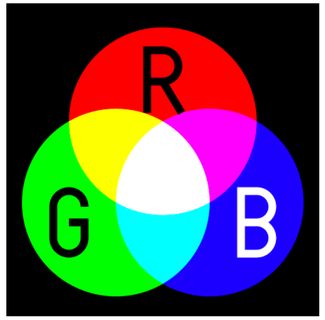
Color mixing for Lego purists



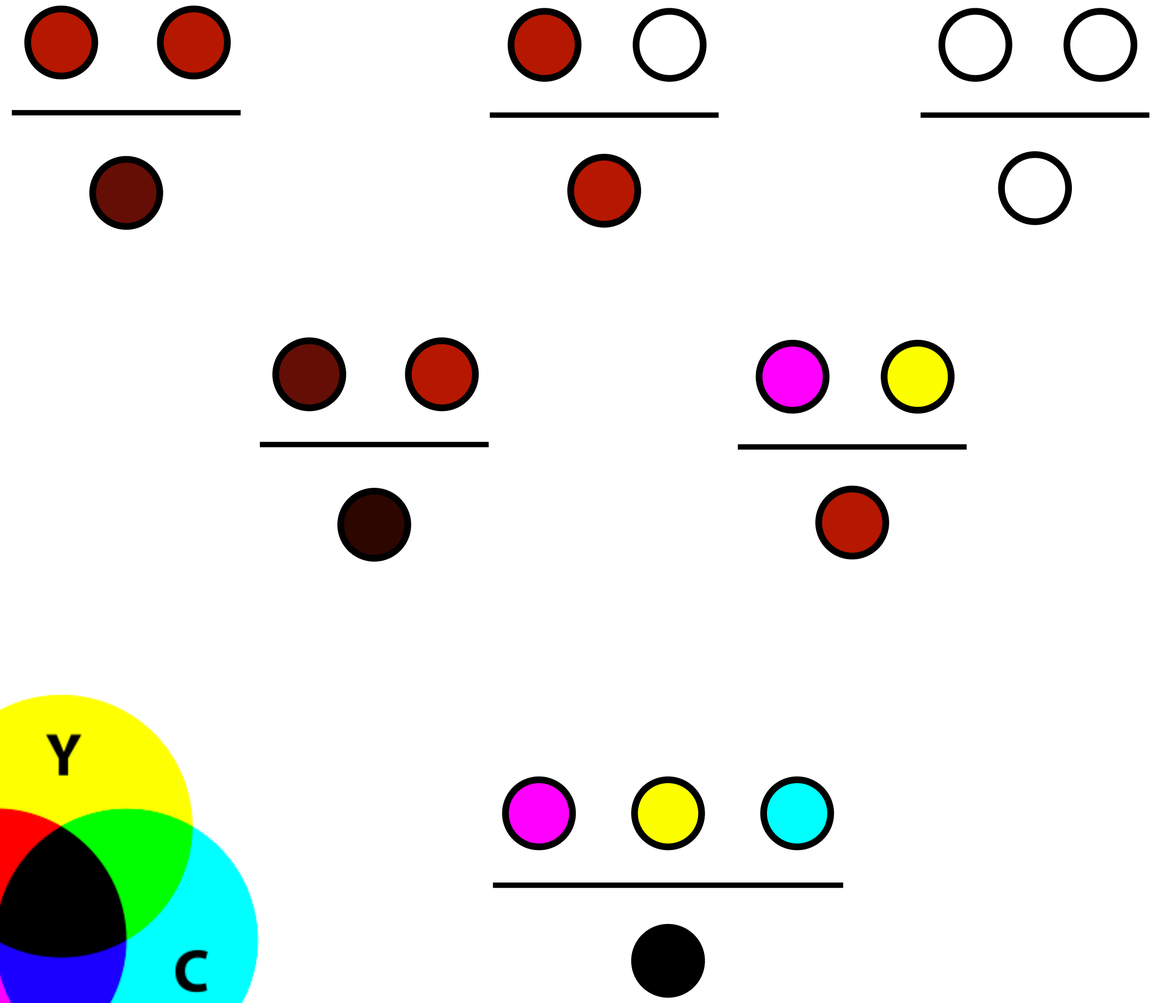
Mixing paint

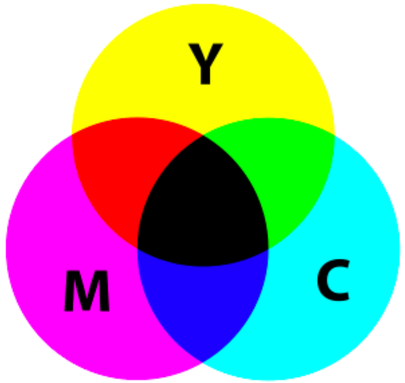


Additive colors (RGB)

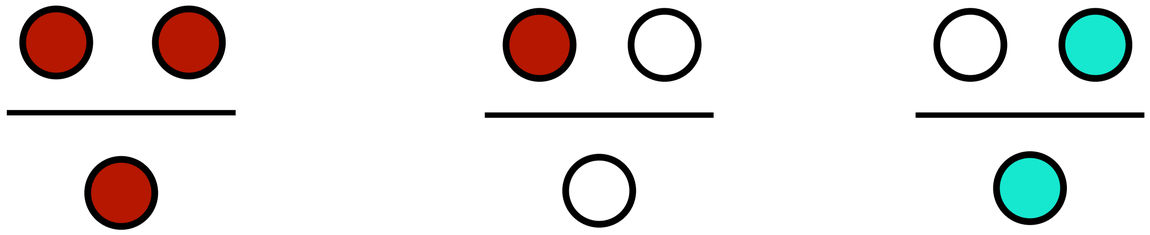


Subtractive colors (CMY)

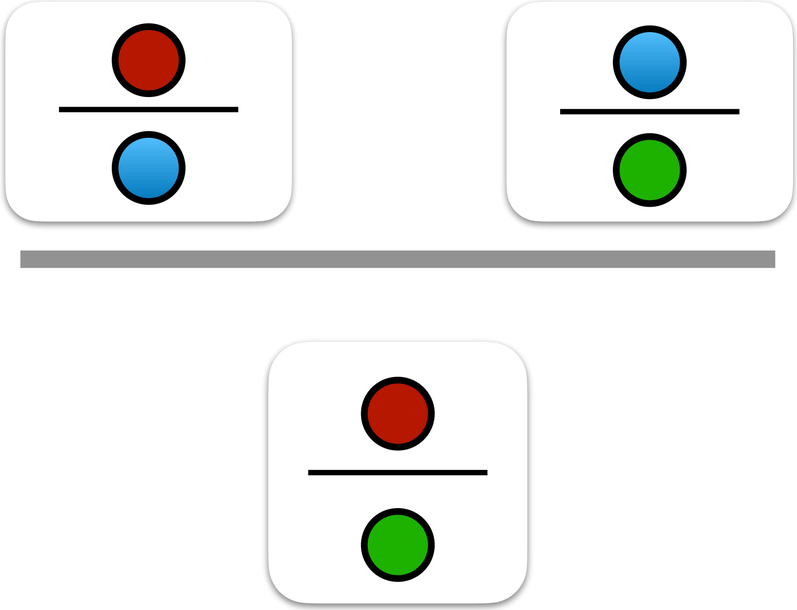




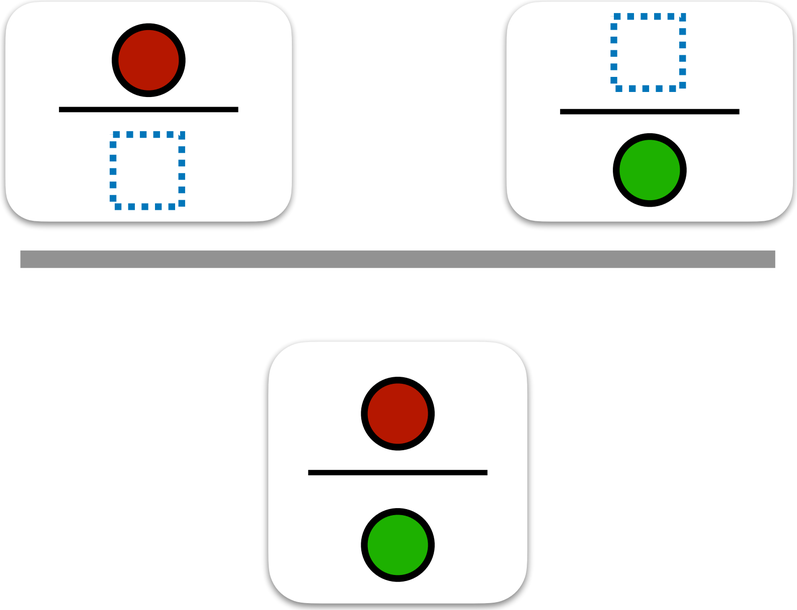
Just paint over it



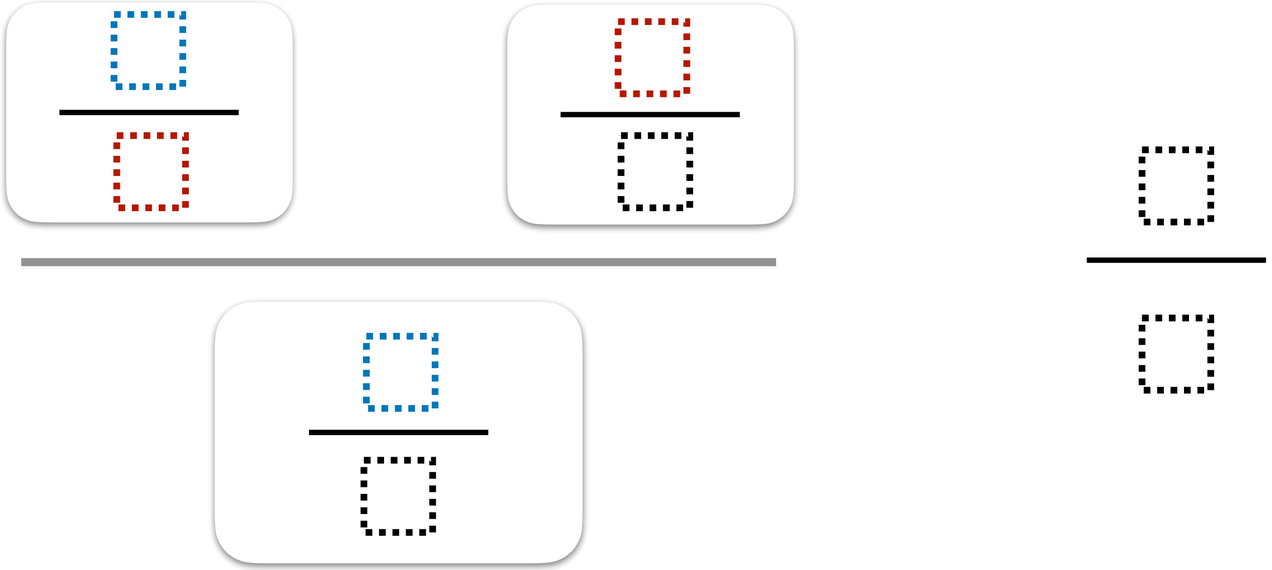
Quoting



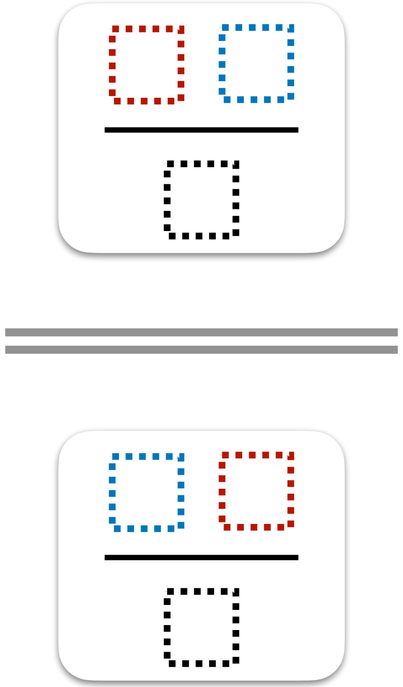
Abstraction



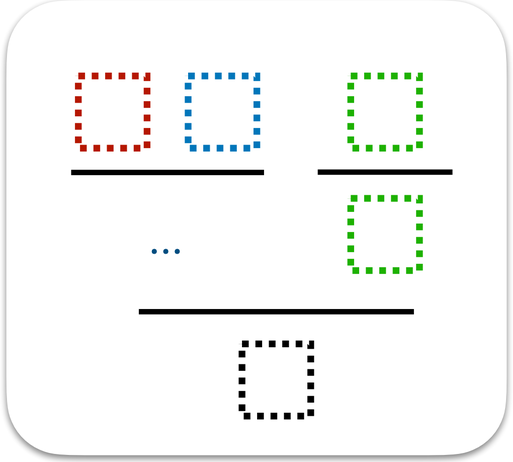
Theorems

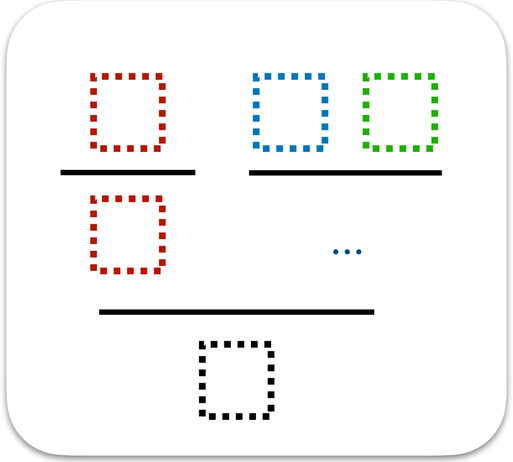


Commutativity

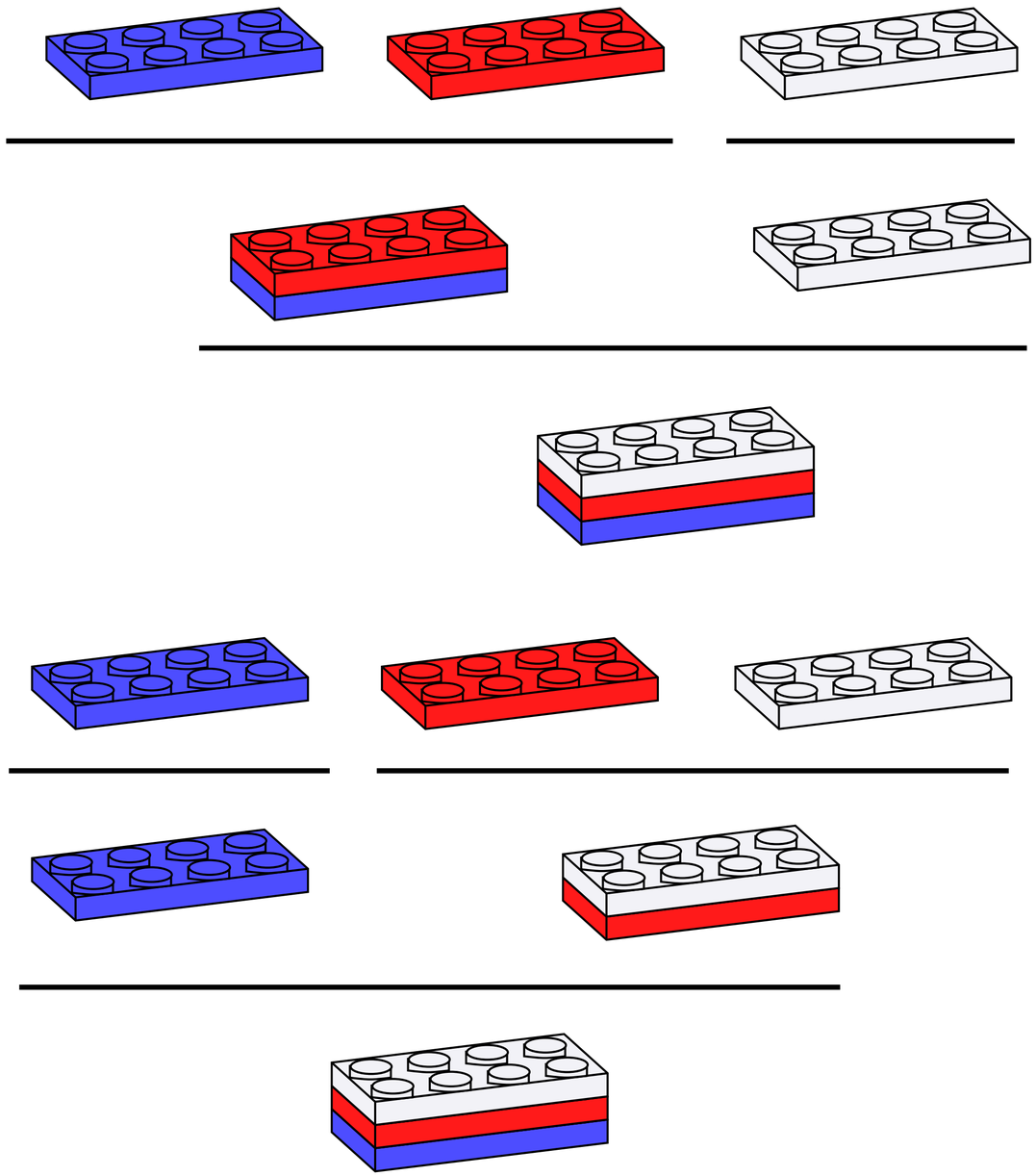


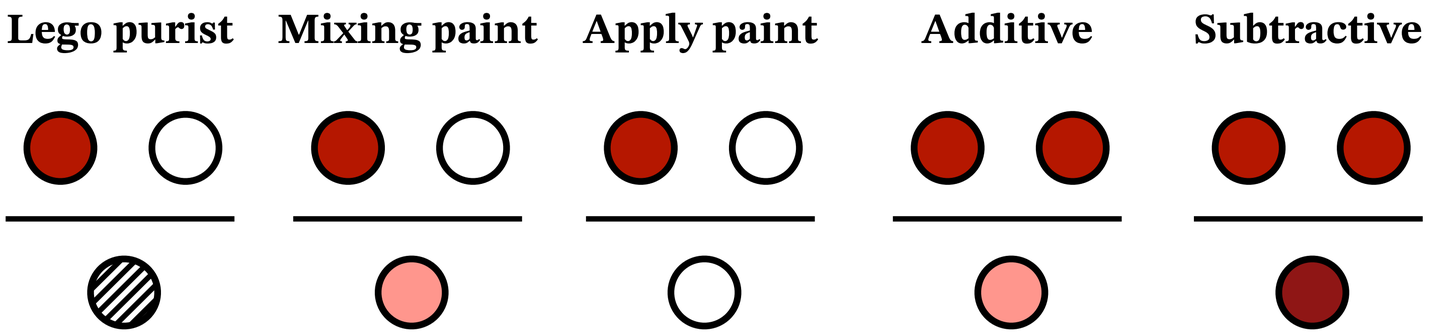
Associativity



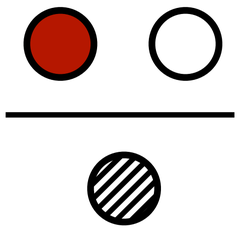
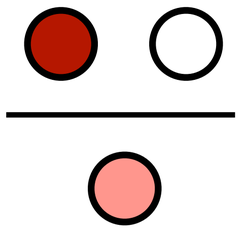
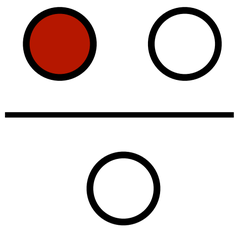
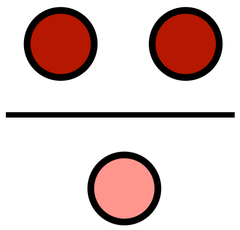
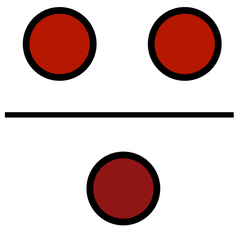


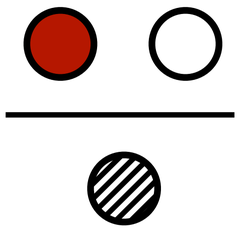
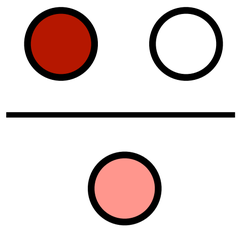
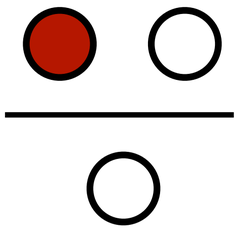
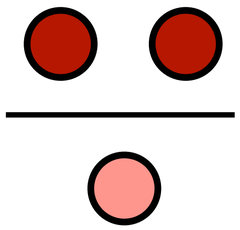
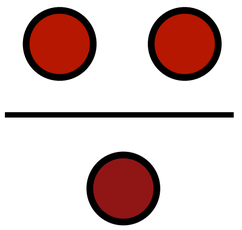
Associativity

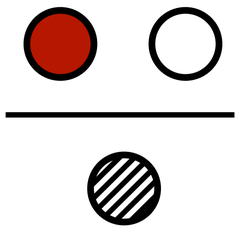
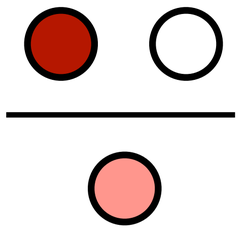
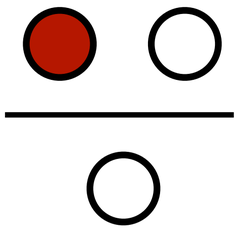
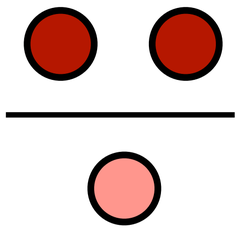
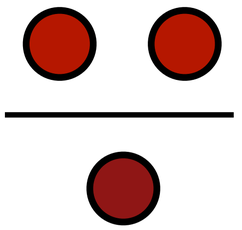


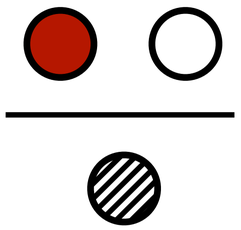
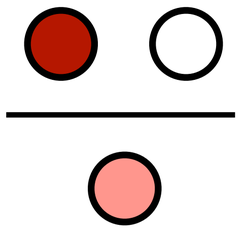
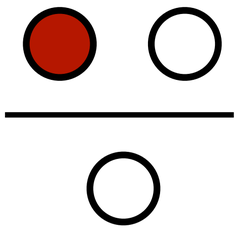
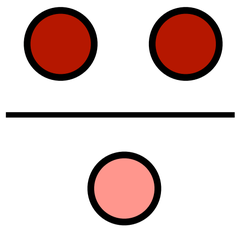
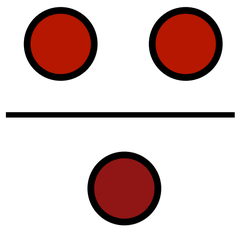
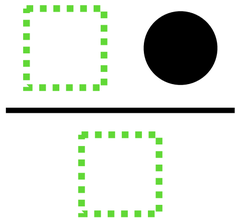
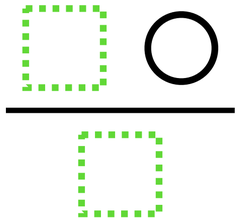


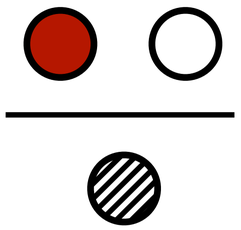
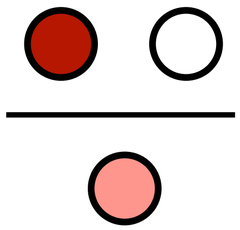
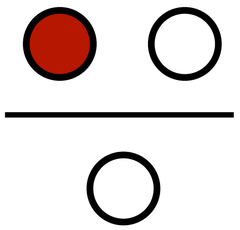
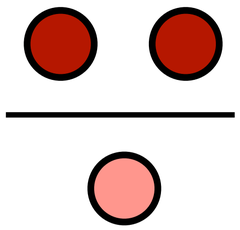
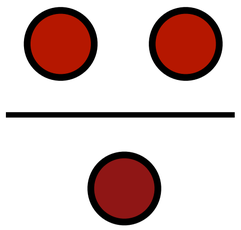
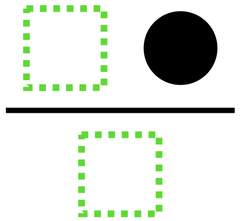
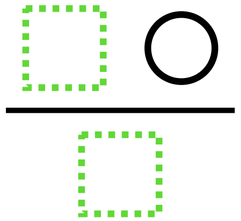
commutative?

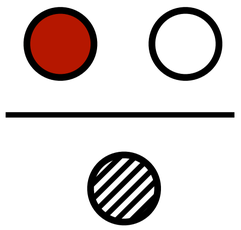
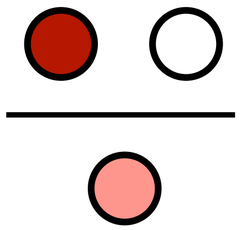
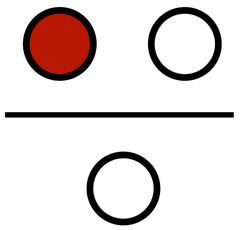
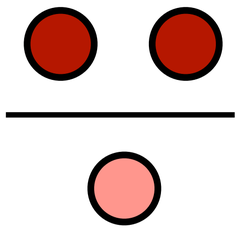
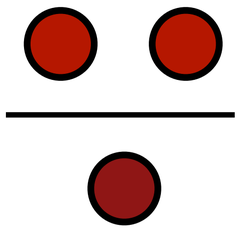
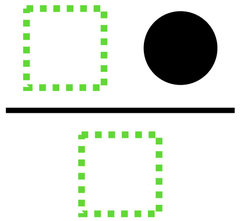
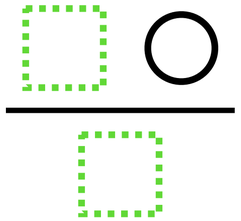
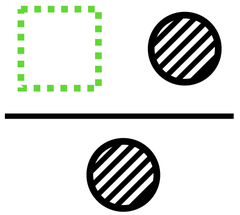
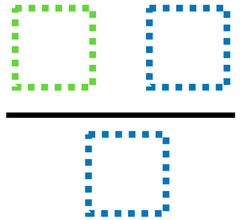
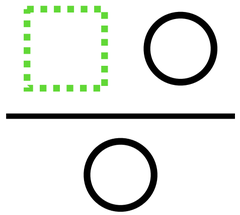
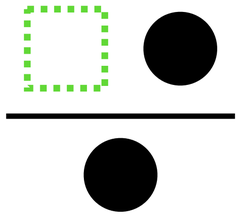
	Lego purist	Mixing paint	Apply paint	Additive	Subtractive
					
commutative?	yes	yes	no	yes	yes
associative?					

	Lego purist	Mixing paint	Apply paint	Additive	Subtractive
					
commutative?	yes	yes	no	yes	yes
associative?	yes	no	yes	yes	yes

	Lego purist	Mixing paint	Apply paint	Additive	Subtractive
					
commutative?	yes	yes	no	yes	yes
associative?	yes	no	yes	yes	yes
neutral element?					

	Lego purist	Mixing paint	Apply paint	Additive	Subtractive
					
commutative?	yes	yes	no	yes	yes
associative?	yes	no	yes	yes	yes
neutral element?	no	no	no		

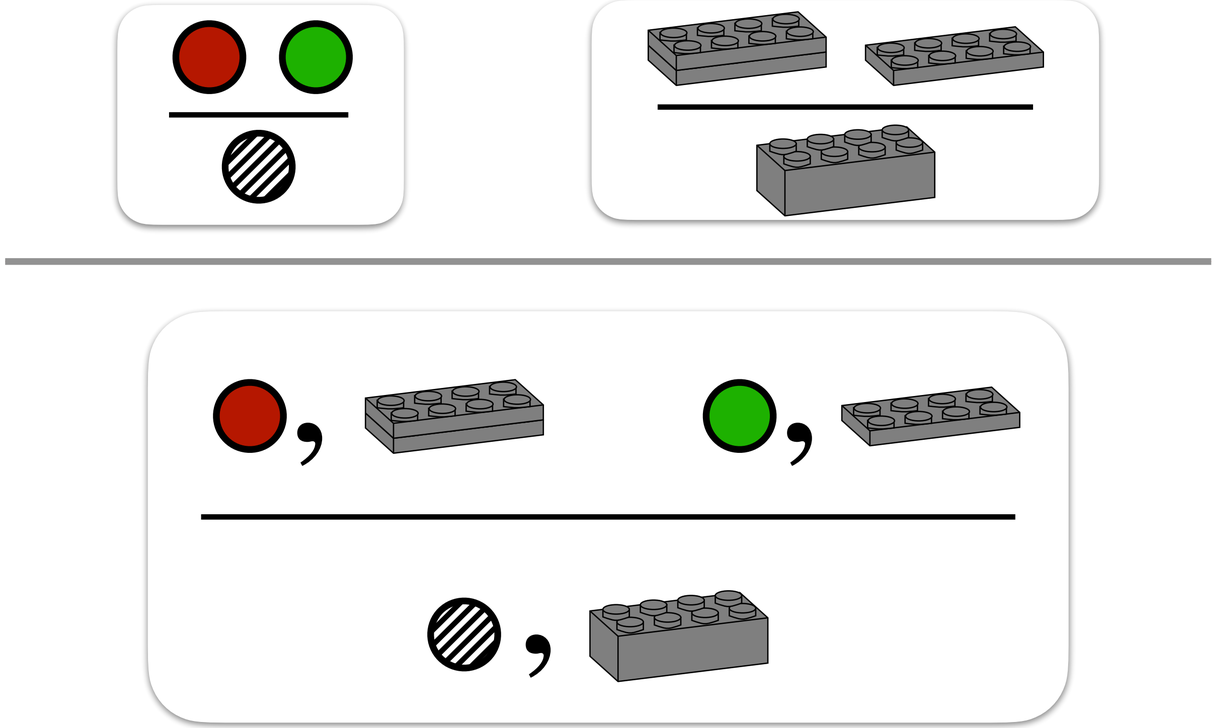
	Lego purist	Mixing paint	Apply paint	Additive	Subtractive
					
commutative?	yes	yes	no	yes	yes
associative?	yes	no	yes	yes	yes
neutral element?	no	no	no		
annihilating element?					

	Lego purist	Mixing paint	Apply paint	Additive	Subtractive
					
commutative?	yes	yes	no	yes	yes
associative?	yes	no	yes	yes	yes
neutral element?	no	no	no		
annihilating element?		no			

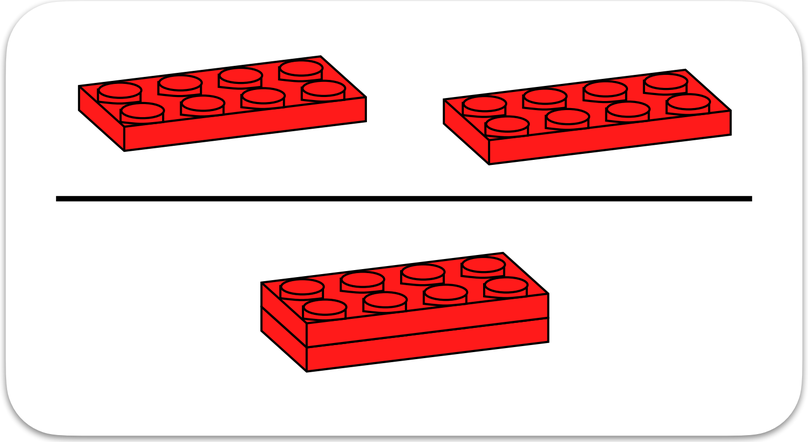
Composition of rules

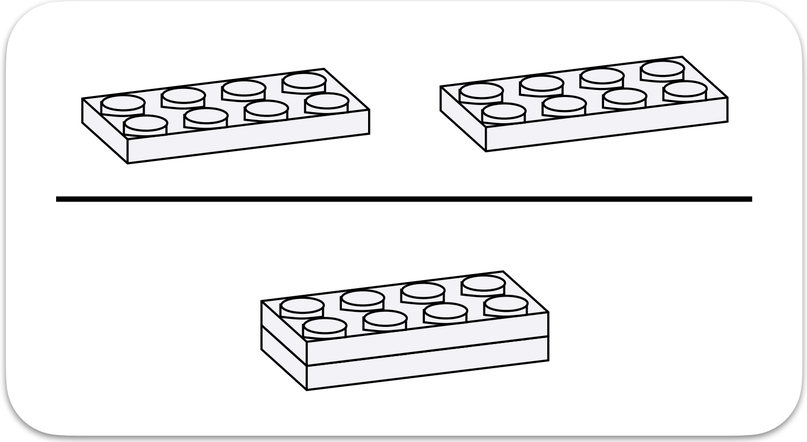
- Each brick has a color and a shape.

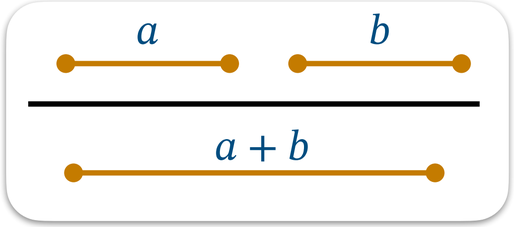
- Composing component-wise:



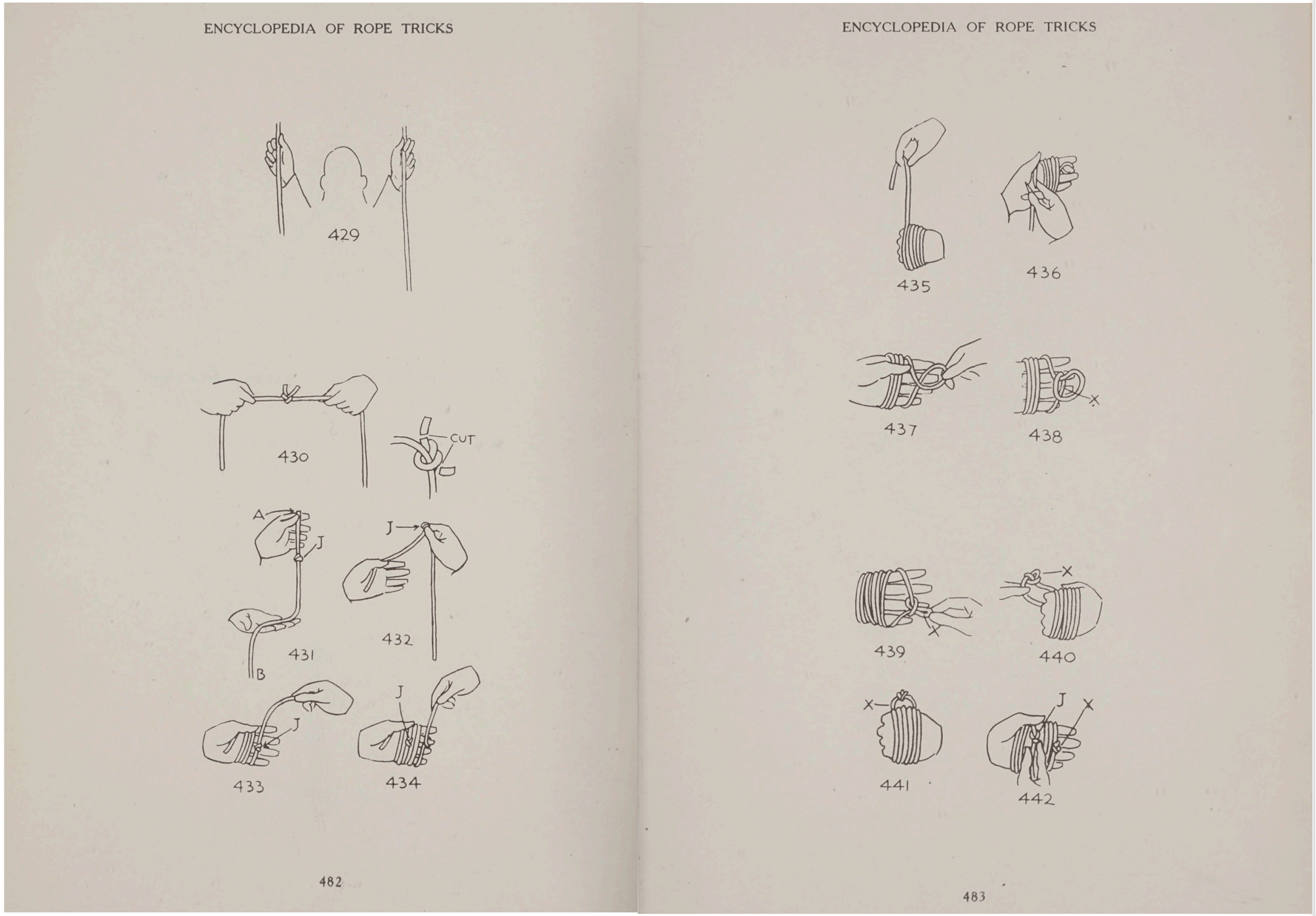
Isomorphisms





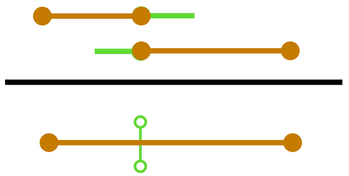


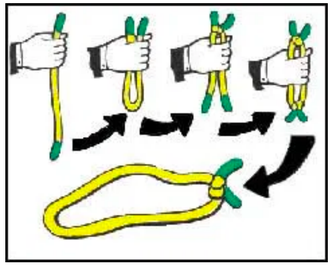
Rope tricks

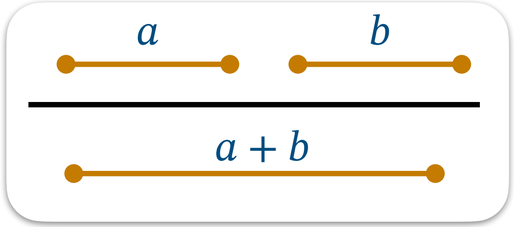

ENCYCLOPEDIA OF ROPE TRICKS
429
430
CUT
A
J
431
B
J
432
J
J
433
J
434
482
ENCYCLOPEDIA OF ROPE TRICKS
435
436
437
X
438
X
X
439
440
X
J
X
441
442
483

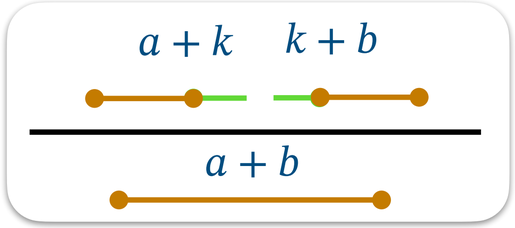
Rope tricks

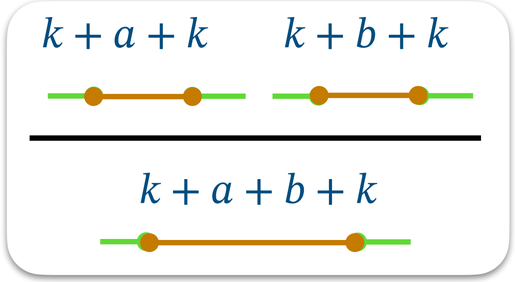
account for knot material

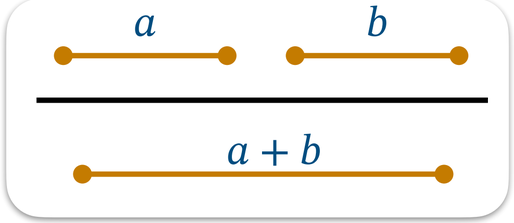




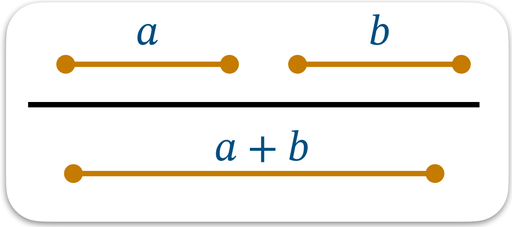


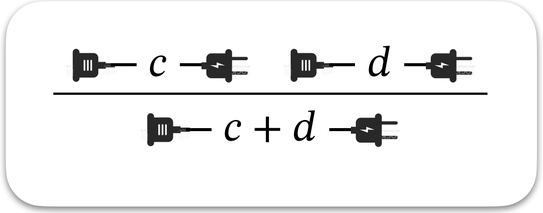




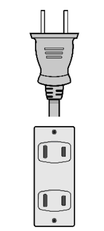


Isomorphisms?

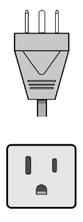




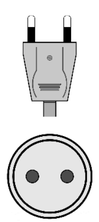
What about the interfaces?



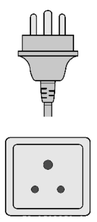
(a) Type A.



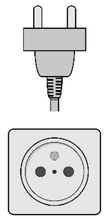
(b) Type B.



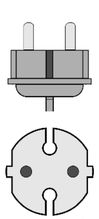
(c) Type C.



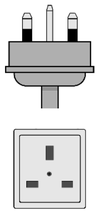
(d) Type D.



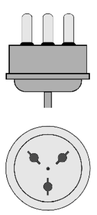
(e) Type E.



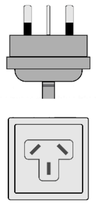
(f) Type F.



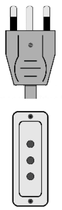
(g) Type G.



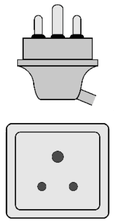
(h) Type H.



(i) Type I.



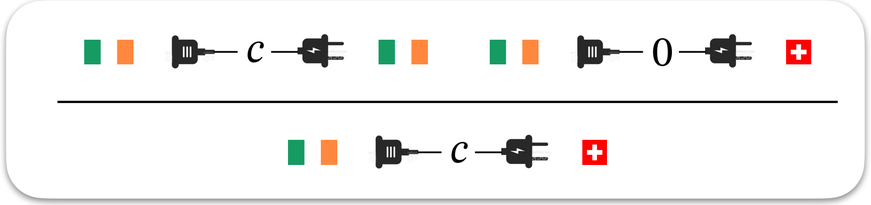
(j) Type L.



(k) Type M.

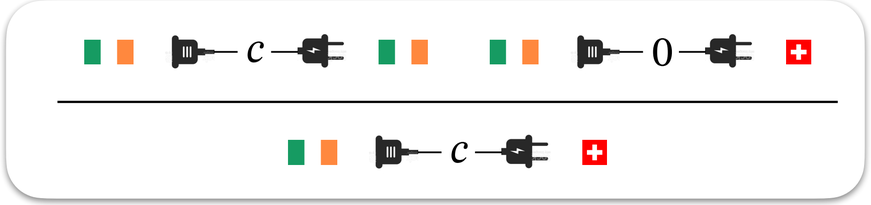
(l) Type N.

Cords with interfaces

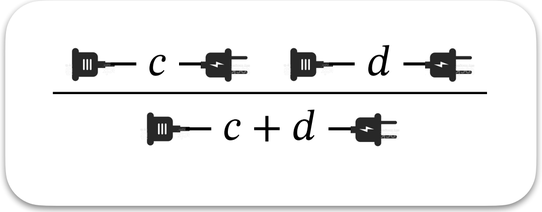


Cords with interfaces

"category"



forget interfaces

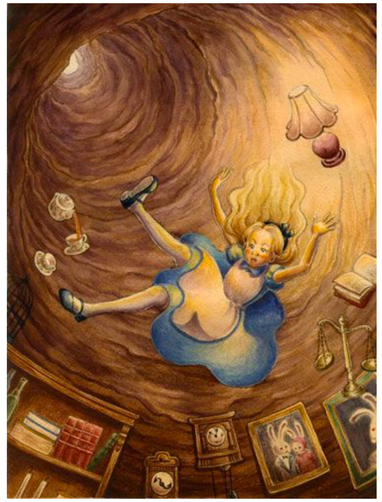


"monoid"

Summary

- **Composition** creates complex structures from small.
- There are **many ways to compose** the same objects.
- **We look for what is common** among different composition structures.

algebra

- Composition is about **interfaces**.
- **Up the ladder of abstraction:** The rules themselves can be composed.
- ...

category theory