Designing Everything (,) Together

Resources and co-design theories for formal engineering design.

- Andrea Censi
  - Senior researcher at ETH Zürich
  - President, Duckietown Foundation
  - Founder, Züpermind
Resource accounting was an early motivation for math (1)

- **You are a sheep herder in 5,000 BCE,** and you have “a lot” of sheep which you take out grazing every day.
- **How can you make sure at the end of the day that no sheep was lost?**

- **The rope method**
  - Take a rope.
  - Tie one knot for every sheep that exits the gate.
  - Untie one knot for each sheep that you bring back at night. (You know the name of each sheep, but it does not matter which sheep)
  - If you have knots left over: go look for the missing sheep!

- **Can you read between the lines?**
  - Definition of natural numbers
  - Definition of cardinality of a set, isomorphisms between sets.
Resource accounting was an early motivation for math (2)

- You are a heir to a sheep empire in 3,000 BCE Babylon. You have a lot of sheep and other riches to count!
- Abstractions of resources like number systems allows to keep track of resources in a compositional way.

3,000 BCE, Babylonian positional number system

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Evolution of Hindu-Arabic modern number systems

with zeros (no resources)
Resource accounting was an early motivation for math (3)

- You are an Egyptian scribe in 2,000 BCE.
- How can you divide the land resources fairly after the annual flood erased last year’s boundaries?

- How can you change one shape into another?
  - rigid transformations
  - joining, diving
- You recognize equivalence through invariants preserved by the operations.
Resource accounting was an early motivation for math (4)

- You are a merchant in India in 600 CE. You need to deal with taxes, loans, and other financial instruments.
- You need the notion of negative resources.

A debt minus zero is a debt.
A fortune minus zero is a fortune.
A zero minus zero is a zero.
A debt subtracted from zero is a fortune.
A fortune subtracted from zero is a debt.

Operations and identities, monoids, conjugations...

Homomorphism from numbers to \{debt, zero, fortune\}.

Brahmagupta, India, 620 CE

Accounting systems
China, 200 BCE
And then?

- Having established proper accounting of sheep, land, and taxes, **math decides to grow beyond thinking about resources**.
  - Friends say she had a fallout with Philosophy and Engineering.
  - Reportedly she went in search of “truth” and “beauty”.
  - Still had occasional flings with the real world with Physics.

**LET’S DO THE TIME WARP**

*To the 21st century!*
22nd Century Problems

- You are a sentient AI in 2125 trying to bootstrap to singularity before the humans notice.
21st Century problems

- You are a few hundreds engineers in 2020 who need to design a fleet of self-driving cars.
The pain of engineering complex systems

So many components (hardware, software, ...), so many choices to make!
Nobody can understand the whole thing!

We forget why we made some choices, and we are afraid to make changes...
These “computer” thingies are not helping us that much for design...

“My dear, it’s simple: you lack a proper theory of co-design!”

anthropomorphization of 21st century engineering malaise

a fleet of self-driving cars =

actuation
sensing
computation
eranetics
software
behavior
coordination
hardware
localization
planning
social acceptance
control
perception
learning
communication
mapping
regulations
liability
interaction
Some references

- **Book**
  - Fong, Spivak: *Seven Sketches in Compositionality*, Chapter 4.

- **Papers**
  - Censi. *A Mathematical Theory of Co-Design* 2015
  - Censi. *Uncertainty in monotone co-design problems*. 2017

- **Online classes**
  - Fong, Spivak - An Invitation to Applied Category Theory
  - Same material: Applied Category Theory by John Baez (Azimuth)
A **design problem** is a **relation** between **provided functionality** and **needed requirements**.
A design problem is a relation between provided functionality and needed requirements.

\[ \langle \mathcal{F}, \leq_{\mathcal{F}} \rangle \quad \text{any partially ordered set} \]

\[ \langle \mathcal{R}, \leq_{\mathcal{R}} \rangle \quad \text{any partially ordered set} \]
A design problem is a relation between provided functionality and needed requirements.
A design problem is a “monotone” relation between a poset of provided functionality and a poset of required resources.

\[ \text{dp}: \mathcal{F}^{\text{op}} \times \mathcal{R} \rightarrow \text{Pos} \rightarrow \text{Bool} \]
A design problem is a relation between provided functionality, required resources, and implementations.

- Choice of battery
  - Capacity [J]
  - Mass [g]
  - Max current [A]
  - Cost [$]

Implementations:
- AA Batteries
- AAA Batteries
- 9V Batteries
- D Batteries
- C Batteries
Is proof relevance relevant?

- **Engineering is “constructive”:** for the purpose of design, we need to know how something is done.

- Morphisms are morally generated by **spans** of this type:
  \[
  \begin{align*}
  \alpha : J &\to \mathcal{F} \\
  \beta : J &\to \mathcal{R}
  \end{align*}
  \]
  \[\mathcal{F} \text{ a poset} \quad \mathcal{R} \text{ a poset} \]
  \[
  f : \mathcal{F}^{\text{op}} \times \mathcal{R} \to \text{Pos} \quad \text{subsets}(J)
  \]
  \[
  \langle f^{\text{op}}, r \rangle \mapsto \{ i \in J : (f \leq_{\mathcal{F}} \alpha(i)) \land (\beta(i) \leq_{\mathcal{R}} r) \}
  \]

- But, **for the purpose of global computation, components only interact through the interfaces**. So we can look at Boolean pro-functors.

  \[
  f : \mathcal{F}^{\text{op}} \times \mathcal{R} \to \text{Pos} \quad \text{subsets}(J) \quad \text{nonempty?} \to \text{Pos} \quad \text{Bool}
  \]

In the **category DP**, also known as $\text{Feas} = \text{Prof}_{\text{Bool}}$

Objects are posets, morphisms are Boolean pro-functors

\[
\begin{align*}
  f &: \mathcal{F}^{\text{op}} \times \mathcal{R} \to \text{Pos} \quad \text{Bool} \\
\end{align*}
\]

**linear logic-like notation:**

\[
  f : \mathcal{R} \to \mathcal{F} \quad (\text{oops!})
\]

- For some algorithms, we will need more assumptions (e.g. objects are DCPOs)
Is proof relevance relevant?

- **Engineering is “constructive”:** for the purpose of design, we need to know how something is done.

- Morphisms are morally generated by **spans** of this type:

\[
\begin{align*}
\forall J & : \text{any set} \\
\alpha : J & \rightarrow \mathcal{F} \\
\beta : J & \rightarrow \mathcal{R} \\
\mathcal{F} & : \text{a poset} \\
\mathcal{R} & : \text{a poset}
\end{align*}
\]

\[
\xrightarrow{f : \mathcal{F}^{\text{op}} \times \mathcal{R} \rightarrow_{\text{Pos}} \text{subsets}(J)}
\]


definition: "which implementations are feasible?"

\[
\begin{align*}
f^{\text{op}}, r & \mapsto \{ i \in J : (f \leq_{\mathcal{F}} \alpha(i)) \land (\beta(i) \leq_{\mathcal{R}} r) \}
\end{align*}
\]

- **If the rest of the talk is too slow** for you:
  - You can re-do all the **constructions with spans**.
  - You can think about the **Curry-Howard isomorphism**.
    (try resources = propositions)
  - You can think about the fact that all is entirely symmetric.
  - Nice Haskell you got there, but is it **reversible**?
resources required by the first system \(\rightarrow\) functionality provided by the second system
Profunctor composition:

\[ f : A \to B \quad g : B \to C \]

\[(f ; g) : A \to C \]

\[ f : A^{\text{op}} \times B \to \text{Pos Bool} \quad g : B^{\text{op}} \times C \to \text{Pos Bool} \]

\[(f ; g) : A^{\text{op}} \times C \to \text{Pos Bool} \]

\[ \langle a^{\text{op}}, c \rangle \mapsto \bigvee_{b_1 \leq b_2} f(a^{\text{op}}, b_1) \land g(b_2^{\text{op}}, c) \]

What is the identity?

\[ \text{Id}_A : A^{\text{op}} \times A \to \text{Pos Bool} \]

\[ \langle a_1^{\text{op}}, a_2 \rangle \mapsto (a_1 \leq_A a_2) \]
choice of chassis

- payload mass [kg]
- max velocity [m/s]

choice of motor

- speed [rad/s]
- torque [Nm]

max current [A]

mass [kg]

required motor velocity [rad/s]

required motor torque [Nm]

extra payload [kg]

max velocity

speed

cost [$]

abstraction

motor mass

torque

cost [$]

max current [A]
Surprising result (at the time):

**The interconnection of any number of monotone design problems is monotone.**
Feedback = irreducible complexity of design

- Battery must power motor
- Motor requires motors to move
- Chassis must carry battery
- Chassis requires motors to move
- Budget must be sufficient for components
- Components must support behaviors

Behaviors implemented should justify the cost
Recap: structure of traced, symmetric, monoidal category

Also: locally posetal; actually enriched in BoundedLat

Objects are posets, hom-sets are lattices: a very “self-aware” category.
Given the functionality to be provided, what are the minimal resources required?

Given the resources that are available, what is the maximal functionality that can be provided?
Assumptions for computation

- For each design problem, we have a representation in terms of antichains.
  
  \[ dp : \mathcal{F}^{\text{op}} \times \mathcal{R} \rightarrow_{\text{Pos}} \text{Bool} \]

  “is this pair of (functionality, resources) feasible?”

- Assume: The posets are **pointed direct-complete partial orders**.

- Assume: The \( h \) maps are **Scott-continuous** ("continuous from the bottom").

- Under these assumptions, we can reduce the optimization problem to a **fixed-point formulation**, and use **Kleene’s algorithm** to find the entire set of optimal solutions (or a certificate of infeasibility).
Semantics as an optimization problem

variables \( r_i \in \langle R_i, \leq R_i \rangle \) \( f_i \in \langle F_i, \preceq F_i \rangle \)

objective \( \text{Min } \bar{r} \)

constraints for each node \( i \): \( f_i \in \mathcal{F}_i \)

\( r_i \in h_i(f_i) \)

for each edge \( (i, j) \):

\( r_i \geq f_j \)

\( f_j \geq r_i \)

\( h_i : \mathcal{F}_i \rightarrow \text{antichains}(\mathcal{R}_i) \)

\( r_i \) chosen by user

\( \bar{r} \) to minimize

\( \bar{f} \)

\( \mathcal{F}_i \) not convex
\( \mathcal{F}_i \) not differentiable
\( \mathcal{F}_i \) not continuous
\( \mathcal{F}_i \) not even defined on continuous spaces
Theorem. The set of minimal feasible resources can be obtained as the least fixed point of a monotone function in the space of antichains.

\[ h_{\text{loop}} : \mathcal{F}_1 \rightarrow \text{antichains}(\mathcal{R}) \]
\[ f_1 \mapsto \text{least-fixed-point}(\Phi_{f_1}) \]

\[ \Phi_{f_1} : \text{antichains}(\mathcal{R}) \rightarrow \text{antichains}(\mathcal{R}) \]
\[ S \mapsto \text{Min} \bigcup_{r \in S} h(f_1, r) \cap \uparrow r \]
Corollary. The set of minimal solutions can be found using Kleene’s algorithm.

\[ S \subset \text{antichains}(R) \]
\[ S_0 = \{ \bot_R \} \]
\[ S_{k+1} = \Phi_{f_1}(S_k) \]

If the iteration diverges, it is a certificate of infeasibility.
The complexity of solving the problem depends on the “thickness” of the “minimal feedback arc set”. 
(Not combinatorial in the size of the implementations!)
A mathematical theory of co-design:

- “co” for **compositional**
- “co” for **computational**
- “co” for **collaborative**
- “co” for **continuous**

What are the **languages and tools**?
Formal language

I developed a **formal language** for co-design

- inspired by Disciplined Convex Programming [Grant & Boyd]

The user can only express monotone constraints

Manual, IDE demo available at http://demo.co-design.science
mc // { 
# We need to fly for this duration
provides endurance [s]
# While carrying this extra payload
provides extra_payload [kg]
# And providing this extra power
provides extra_power [W]

# Sub-design problem: choose the battery
sub battery = mcdp {
  # A battery provides capacity
  provides capacity [J]
  # and requires some mass to be transported
  requires mass [kg]
  # requires cost [$]

  specific_energy_Li_Ion = 500 Wh / kg
  mass >= capacity / specific_energy_Li_Ion
}

# Sub-design problem: actuation
sub actuation = mcdp {
  # actuators need to provide this lift
  provides lift [N]
  # and will require power
  requires power [W]
  # simple model: quadratic
  c = 10.0 W/N^2
  power >= lift * lift * c
}

# Co-design constraint: battery must be large enough
power = actuation.power + extra_power
energy = power * endurance
battery.capacity >= energy

# Co-design constraint: actuators must be powerful enough
gravity = 9.81 m/s^2
weight = (battery.mass + extra_payload) * gravity
actuation.lift >= weight

# suppose we want to optimize for size of the battery
requires mass for battery
}
mcdp {
  # We need to fly for this duration
  provides endurance [s]
  # While carrying this extra payload
  provides extra payload [kg]
  # And providing this extra power
  provides extra_power [W]

  # Sub-design problem: choose the battery
  sub battery = mcdp {
    # A battery provides capacity
    provides capacity [J]
    # and requires some mass to be transported
    requires mass [kg]
    # requires cost [$]

    specific_energy_Li_Ion = 500 Wh / kg

    mass >= capacity / specific_energy_Li_Ion
  }

  # Sub-design problem: actuation
  sub actuation = mcdp {
    # actuators need to provide this lift
    provides lift [N]
    # and will require power
    requires power [W]
    # simple model: quadratic
    c = 10.0 W/N^2
    power >= lift * lift * c

  }  

  # Co-design constraint: battery must be large enough
  power = actuation.power + extra_power
  energy = power * endurance
  battery.capacity >= energy

  # Co-design constraint: actuators must be powerful enough
  gravity = 9.81 m/s^2
  weight = (battery.mass + extra_payload) * gravity
  actuation.lift >= weight

  # suppose we want to optimize for size of the battery
  requires mass for battery
}
Uncertainty in co-design

- A category of **Uncertain DPs:**
  - The objects are posets
  - The morphisms are ordered pairs of morphisms of DP.
- Everything generalizes easily.
Uncertainty in co-design problems

```
battery_uncertain.mcdp

mcdp {
  provides capacity [kWh]
  requires mass [g]
  requires cost [$]
  energy_density = between 140 kWh/kg and 150 kWh/kg
  specific_cost = 200 $/kWh
  required mass \cdot energy_density \geq provided capacity
  required cost \geq provided capacity \cdot specific_cost
}
```

no uncertainty:  "To obtain an endurance of **15 min**, the minimal cost is **$230**"

low uncertainty:  "To obtain an endurance of **15 min**, the minimal cost is **between $220 and $240**"

high uncertainty:  "To obtain an endurance of **15 min**, the minimal cost is **$220 in the best case, and in the worst case the problem is not feasible**"
Uncertainty for relaxation
Next steps for co-design theory

- The simple profunctor formalization does not capture all aspects of interest to the field of engineering design.

- **Seven more sketches** of more refined resource theories:
  1. **Monoidal and tropical** resources.
  2. **Homogeneous** resources
  3. **Higher-order** design theory.
  4. **Linearity** of resources.
  5. **Temporality** of resources.
  6. **Spatiality** of resources.
  7. **Negative co-design**.

- Note: **this is work in progress**, and open for collaboration. Let me know if this picks your fancy.
1. Monoidal and tropical resources

- For many resources posets, there is a “zero resource”, **resources can be added** together, etc.
- Assume the posets to have a **commutative monoid structure** compatible with the order.

\[
(\oplus, 0) \\
\underline{a \leq b} \\
\frac{a \oplus c \leq b \oplus c}
\]

- You can generate for free a lot of interesting DPs.

\[
a \oplus b \oplus c \leq a \oplus b
\]

- Careful, for some resources (e.g. time) you have both + and max:

\[
a \oplus b = \max(a, b) \\
a \otimes b = a + b
\]

“tropical semiring”
2. Homogeneous / heterogeneous resources

- Sometimes you want to further characterize the quality of resources:
  - “To make a car I need 4 identical wheels”
  - “We cannot go to the party if we are dressed the same”

- Say that $A$ is a type with **equality** and **apartedness**.
  
  $x =_A y$ \textit{type of proofs that $x$ and $y$ are equal at $A$}
  
  $x \neq_A y$ \textit{type of proofs that $x$ and $y$ are different at $A$}

\[ A_{\neq 2} \overset{\text{def}}{=} (x : A) \times (y : A) \times (p : x \neq_A y) \quad \text{two certifiably different } A 
\]
\[ A_{= 2} \overset{\text{def}}{=} (x : A) \times (y : A) \times (p : x =_A y) \quad \text{two certifiably equal } A 
\]

- Interestingly, \( (A_{\neq m})_{= n} \sim (A_{= n})_{\neq m} \overset{\text{def}}{=} A_{\neq m} \)

\[
(A_{\neq 2})_{= 3} \quad (A_{= 3})_{\neq 2}
\]
3. Higher-order structure

- Define a “template” as a diagram with holes that we can fill with DPs to obtain another DP.

- One can prove the following “representation result”:
  - For every template, the operation that fills the holes is a monotone map between DPs hom-sets.
  - and viceversa
  - Every monotone map between DPs hom-sets can be expressed as a template-filling operation.

- For a $T : \text{Hom}(A; B) \rightarrow_{\text{Pos}} \text{Hom}(C; D)$ what is $\ldots T(\underbrace{T(T(T(\cdots))))} \ldots$ ?
3. Higher-order structure

- DP as defined is **compact closed**.

\[ A \to (B \to C) \simeq (A \otimes B) \to C \]
\[ (A \to B) \to C \simeq B \to (A \otimes C) \]

- This implies that **all higher-order structure collapses**.
4. Linear resources

- Linear logic is a “logic of resources.”
- Atoms are **uncopiable, unshareable, undiscardable** “resources”.

\[
\begin{align*}
\text{classical logic} & \quad \text{(programming, etc.)} \\
A & \quad A \Rightarrow B \\
\hline
A & \quad A \Rightarrow B & B
\end{align*}
\]

“You are strolling in a garden finding landmarks and maps to find other landmarks.”

\[
\begin{align*}
\text{linear logic} \\
A & \quad A \rightarrow B \\
\hline
\end{align*}
\]

“You are consuming resources”

\[
\begin{align*}
A \rightarrow B & \quad \text{the type of single-use machines that eat } A, \text{ produce } B, \text{ and then disappear}
\end{align*}
\]
4. Linear resources

- Linear logic is a “logic of resources.”
- Atoms are uncopiable, unshareable, undiscardable “resources”.

- Two “multiplicatives”, two “additives” four units, an involution, two modalities!

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A &amp; B$</td>
<td>you can choose to have either an A or a B</td>
<td>$\top$</td>
</tr>
<tr>
<td>$A \oplus B$</td>
<td>somebody chooses if you have an A or a B</td>
<td>$0$</td>
</tr>
<tr>
<td>$A \otimes B$</td>
<td>you have both an A and a B</td>
<td>$1$</td>
</tr>
<tr>
<td>$A &amp;! B$</td>
<td>if you understand this, you are enlightened</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$!A$</td>
<td>you have 0 or more of A</td>
<td></td>
</tr>
<tr>
<td>$?A$</td>
<td>you can discard 0 or more of A</td>
<td></td>
</tr>
</tbody>
</table>
4. Linear resources

- What would a theory of linear co-design be?

- **Linear DPs (?)**
  - resources can be copied and discarded
  - no distinction between external / internal choices
  - all posets are Bool
  - no proof relevance

- **DPs**
- **Linear Logic**
5. Temporal resources

- Suppose I want to keep track of the time it takes to produce something.

- Let’s decorate the morphisms and **refine the hom-types by time** (some commutative monoid).
  
  $\text{id} : A \overset{0}{\rightarrow} A$
  
  $A \overset{s}{\rightarrow} A$ might be uninhabited
  
- **Composition** is as you expect:
  
  $A \overset{\alpha}{\rightarrow} B$  $B \overset{\beta}{\rightarrow} C$
  
  $A \overset{\alpha + \beta}{\rightarrow} C$
5. Temporal resources

- **Monoidal composition** is only allowed if the temporal duration matches or the resources are “non perishable”.

\[
\begin{array}{c}
A \xrightarrow{\alpha} B \quad C \xrightarrow{\alpha} D \\
\hline
A \otimes C \xrightarrow{\alpha} B \otimes D
\end{array}
\]

- The higher order structure does not collapse anymore:

\[
(A \xrightarrow{\alpha} B) \xrightarrow{\beta} C \approx? B \xrightarrow{\gamma} (A \otimes C)
\]

\[
A \xrightarrow{\alpha} (B \xrightarrow{\beta} C) \approx? (A \otimes B) \xrightarrow{\gamma} C
\]
5. Temporal resources

- **Negative time** is also interesting and not quite equivalent to negative resources.

- You have promised that you’ll give me $10 tomorrow.

- **Lending:**
  - If I can lend $10 now, I can have $11 back later.

- **Borrowing:**
  - If I can repay $11 in 1 week, I can have $10 now.

- Net present value axioms with interest rate $\alpha$:
  - $1 \overset{t}{\rightarrow} \$x \quad \Longleftrightarrow \quad 1 \overset{t+\delta}{\rightarrow} \alpha^\delta \cdot \$x$

\[ I \ have: \quad 1 \overset{1 \ \text{day}}{\rightarrow} \$10 \]
\[ You \ have: \quad 1 \overset{1 \ \text{day}}{\rightarrow} -\$10 \]
\[ or: \quad 1 \overset{1 \ \text{day}}{\rightarrow} ($10 \overset{0}{\rightarrow} 1$) \]
6. Spatial resources

- We can describe **constraints on the arrangements of resources.**
- Take a simplified metric structure where things are either “close” or “far apart”.

  \[ A \perp B \quad \text{Two separable resources.} \]

  \[ A \asymp B \quad \text{Two inseparable resources.} \]

  \[ A \mapsto B \quad \text{Two resources that are constrained to be apart.} \]

- It’s a **linear distributive category twice over** (3 monoidal structures):

  \[ A \perp (B \asymp C) \rightarrow (A \perp B) \asymp C \]

  \[ A \perp (B \mapsto C) \rightarrow (A \perp B) \mapsto C \]

- More generally, given a metric space we can create an infinite amount of tensor operations representing resource placement constraints.

- (If object = tensors, morphisms = tensor strength domination, what does this category look like?)

\[ A \boxtimes B \quad \text{You must place } A \text{ and } B \text{ in sets } S_A, S_B \subseteq \mathcal{S} \text{ such that } m \leq d_S(S_A, S_B) \leq M. \]
6. Spatial resources

- Take a simplified metric structure where things are either “close” or “far apart”.
  
  \[
  A \not\sim B \quad \text{Two separable resources.}
  \]
  
  \[
  A \sim B \quad \text{Two inseparable resources.}
  \]
  
  \[
  A \leadsto B \quad \text{Two resources that are constrained to be apart.}
  \]

- Other constraints: e.g. machines need to be close to materials for the production to happen.

- Inspires to be a bit more “precise” in “subtyping”:
7. Negative co-design

- In this universe there is conservation of mass and energy.

The real world axiom.

For any \( f : A \to A \) necessarily \( f \leq \text{Id}_A \)

- This means that if you give me any profunctor \( g : A \to B \)
  - You are telling me something positive
    about what \( B \) I can produce from \( A \).
  - And, you are telling me something negative
    about what \( A \) I can produce from \( B \).

\[
\begin{align*}
\text{For any } h : B & \to A \text{ necessarily } (g ; h) \leq \text{Id}_A \\
(h ; g) & \leq \text{Id}_B
\end{align*}
\]

- Gives an entire new symmetry to explore!
  Very relevant to the computation side (impossibility results).
Conclusions