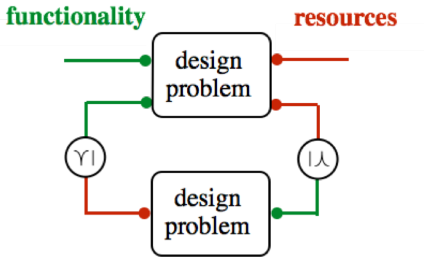


A Categorical Theory of Co-Design

Joshua Tan (MIT, Oxford, Categorical Informatics)

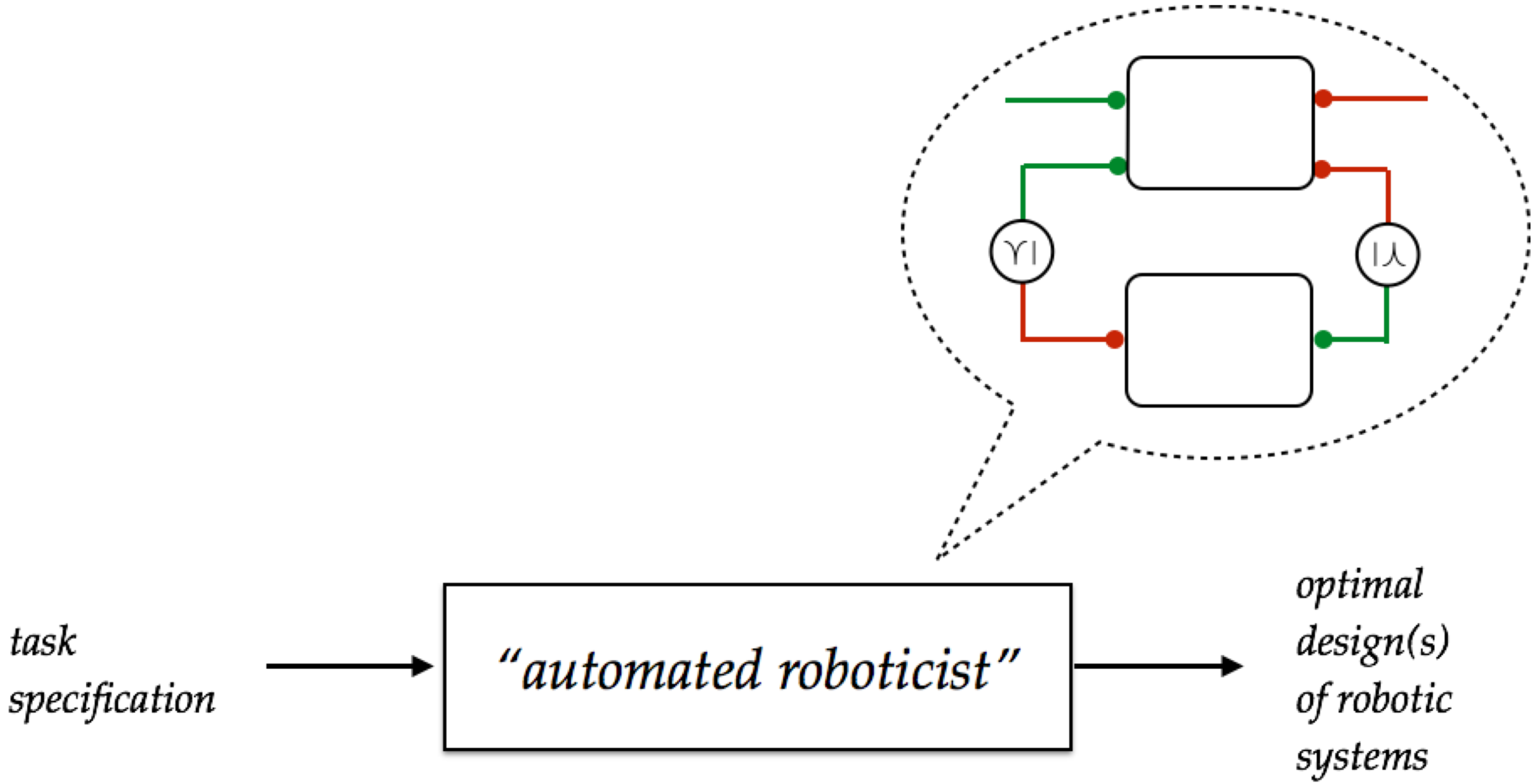
Applied Topology Seminar, University of Pennsylvania

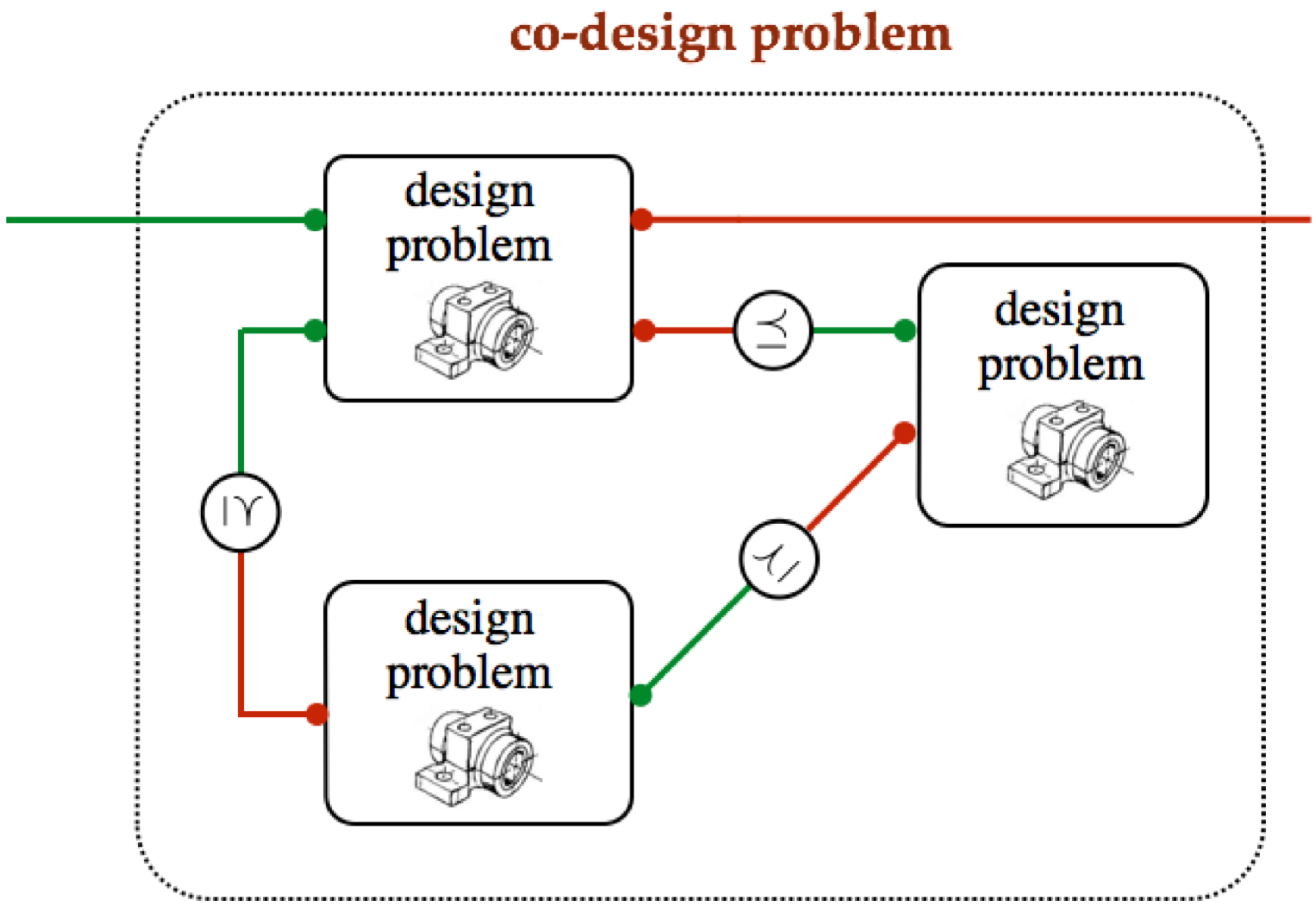
September 6, 2016

“A Mathematical Theory of Co-Design” (2015)

- Andrea Censi* (MIT)
- “A new class of optimization problems”
- “Rich enough to capture most of the irreducible complexity in robotics”
- “Possibly useful in other fields having equally complex systems to design”

*credit for graphics


task specification
"automated roboticist"
optimal design(s) of robotic systems



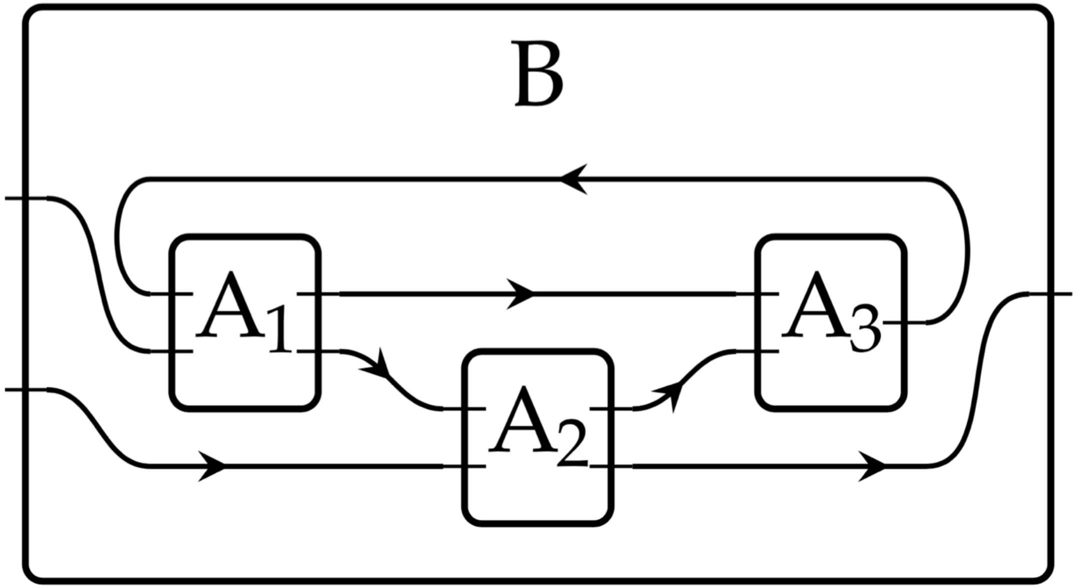
co-design problem
design problem
design problem
design problem

Wiring Diagrams in Category Theory

- David Spivak (MIT)
- Wiring diagrams can be formalized as **traced symmetric monoidal categories***
- Doing so is useful because the syntax is
  visual and intuitive,
  expressive,
  but still regular and consistent,
  well-tailored to complex simulations (think Simulink)
  extensible

*also as operads

$$\phi : (A_1, A_2, A_3) \rightarrow B$$



Can wiring diagrams capture the semantics of co-design diagrams?

Tan, Censi, Spivak: Yes!

Structure of this talk

- Definition of design problems as a class of optimization problems

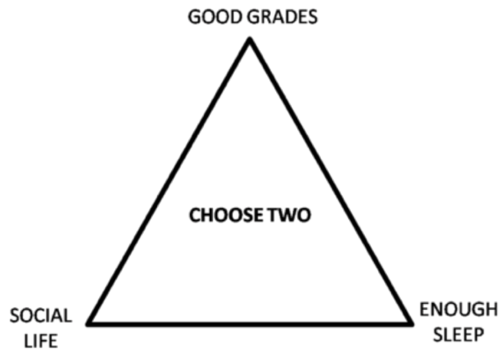
High-level: design problems
MTC: {F, R, I, exec, eval}
CTC: an object in the **DP** category

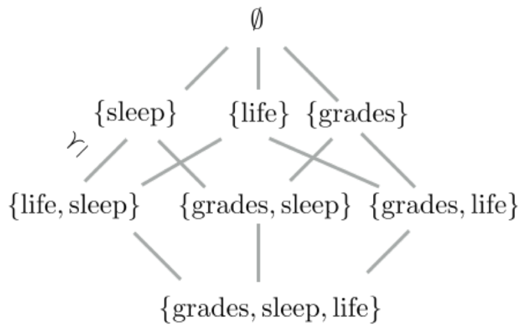
- Theorem 1: how to compose design problems
- Theorem 2: how to solve design problems
- Theorem 3 (in progress): computational complexity
- Future work

Partially-ordered sets

High-level: a poset is a set with a reflexive, antisymmetric, and transitive relation, $\preccurlyeq$

$$a \preceq b \doteq \text{“I prefer } a \text{ to } b\text{.”}$$





MTC: posets are just sets

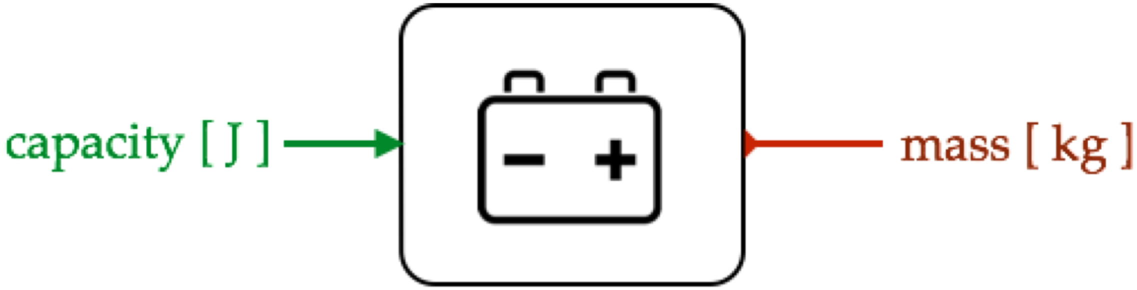
CTC: posets are also categories! In particular, they form a category, **Poset**

Monotone Co-Design Problems

High-level: a design problem is relation between the required inputs ("functionalities") and outputs ("resources")

MTC: dp = {*F, R, I, exec, eval*}, where:

- *F* is a poset of functionalities,
- *R* is a poset of resources,
- *I* is a set of implementations
- *exec: I* → F and *eval: I* → *R*



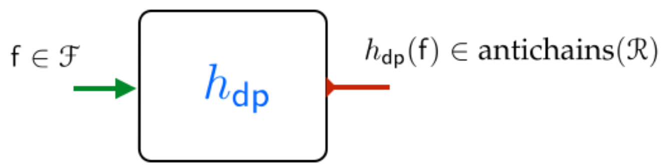

```
mcdp {
    provides capacity [J]
    requires mass [kg]

    specific_energy_Li_Ion = 500 Wh / kg

    mass >= capacity / specific_energy_Li_Ion
}
```

Monotone Co-Design Problems

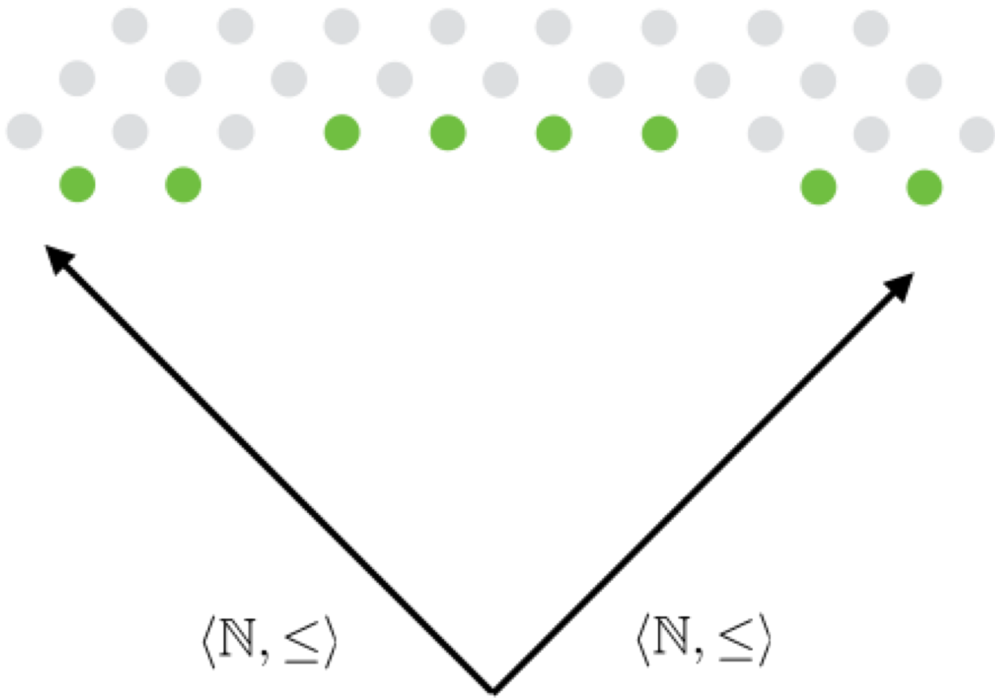
High-level: a design problem is relation between the required inputs ("functionalities") and outputs ("resources")

MTC: Further, every dp induces a map $h_{dp} : F \to \mathcal{A}R$ that represents the question: given the **minimal** functionality required, what are the **minimal** sets of resources which provide it?

- **Definition**: $S \subseteq P$ is an *antichain* if no two elements are comparable: for $x, y \in S$, $x \preccurlyeq y$ implies $x = y$. We write $\mathcal{A}P$ for the poset of all antichains of P.



The **minimal elements** of a poset are an antichain.



Monotone Co-Design Problems

High-level: a design problem is relation between the required inputs ("functionalities") and outputs ("resources")

CTC: every design problem is a morphism in the **DP** category with objects posets and morphisms design problems, where a design problem is represented as a *profunctor*

$$\mathrm{dp} : A \to B \qquad = \qquad [\mathrm{dp}]: A^{\mathrm{op}} \times B \to \mathbf{Set}$$

Monotone Co-Design Problems

High-level: a design problem is relation between the required inputs ("functionalities") and outputs ("resources")

CTC: every design problem is a morphism in the $\mathbf{DP_B}$ category with objects posets and morphisms design problems, where a design problem is represented as a *profunctor*

$$\mathrm{dp} : A \to B \qquad = \qquad [\mathrm{dp}]: A^{\mathrm{op}} \times B \to \mathbf{Bool}$$

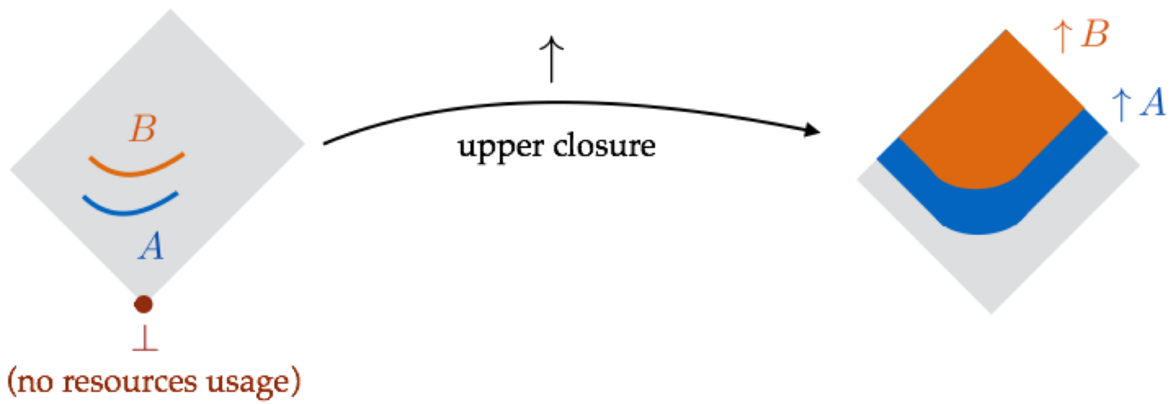
Monotone Co-Design Problems

High-level: a design problem is "monotone" if decreasing the functionality required or increasing the resources available will never decrease the number of feasible solutions.

MTC: assume $h_{dp} : F \to \mathcal{A}R$ is *monotone*

- **Definition**: a map $f : P \to Q$ between two posets is *monotone* (order-preserving) iff $p \preccurlyeq p'$ implies $f(p) \preccurlyeq f(p')$

CTC: assume [dp]: $A^{op} \times B \to$ **Bool** is monotone, i.e. that it is an actual *(pro)functor*

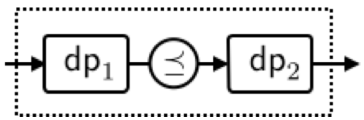


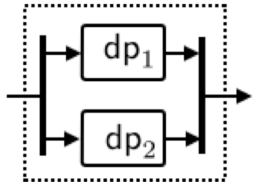
B
A
$\bot$
(no resources usage)
$\uparrow$
upper closure
$\uparrow B$
$\uparrow A$

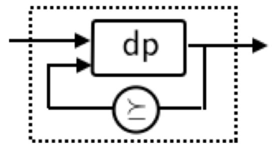
Monotone Co-Design Problems

High-level: a monotone co-design problem is the composition of many design problems under three operations

MTC: three operations: “series”, “parallel”, and “loop” that preserve monotonicity



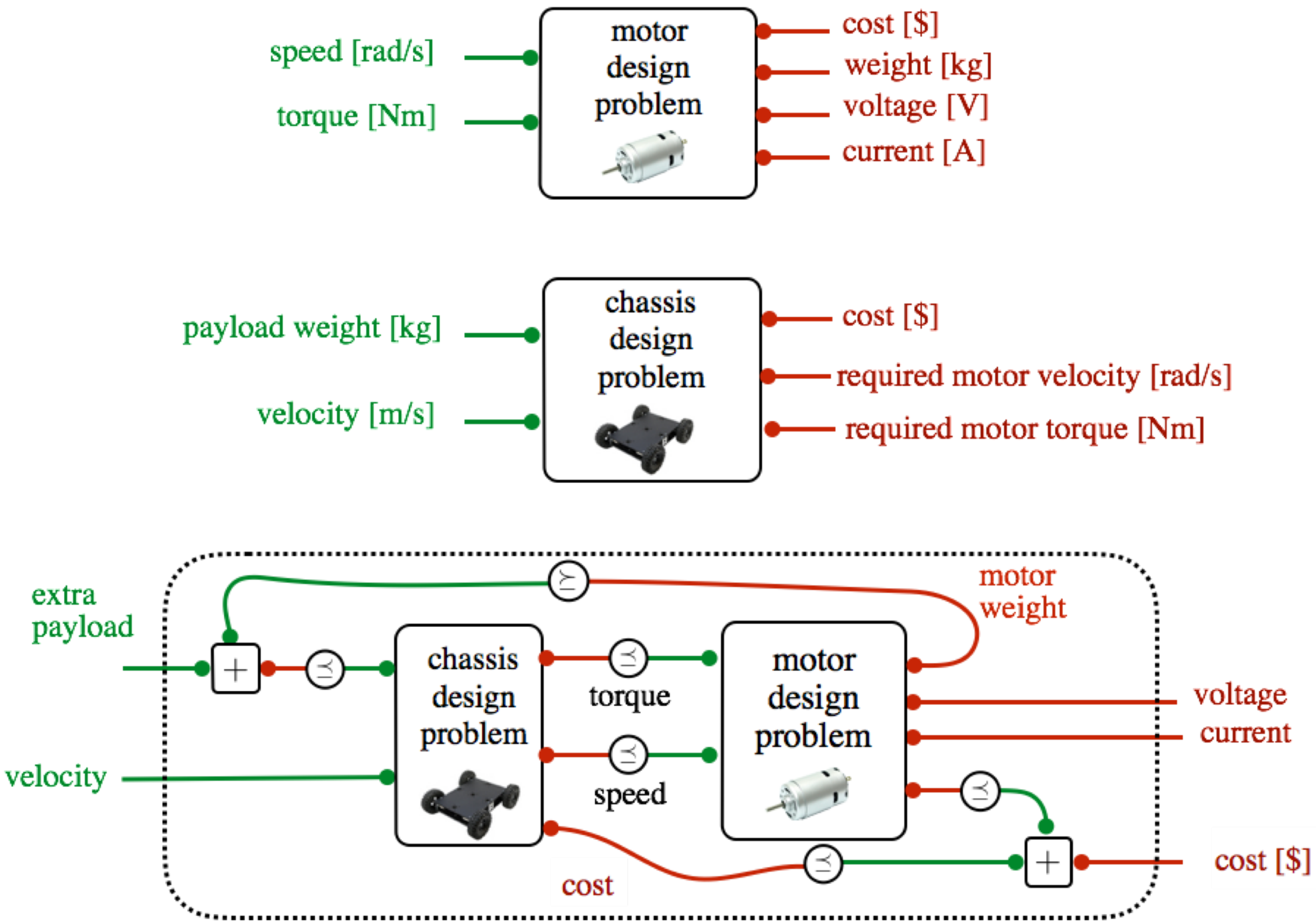




CTC: three properties of **DP**: composition, monoidal product, trace

$dp_2 \circ dp_1$ “$dp_1 \times dp_2$” “Tr(dp1)”

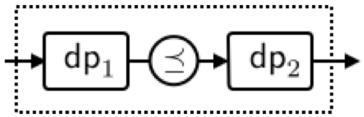
How do we compose design problems to form co-design problems?



motor
design
problem
speed [rad/s]
torque [Nm]
cost [$]
weight [kg]
voltage [V]
current [A]
chassis
design
problem
payload weight [kg]
velocity [m/s]
cost [$]
required motor velocity [rad/s]
required motor torque [Nm]
extra
payload
velocity
chassis
design
problem
torque
speed
motor
design
problem
motor
weight
voltage
current
cost
cost [$]

“Series”

High-level: the “series” of design problems is still a design problem



MTC:

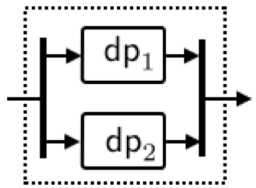
$$\begin{aligned} h_{\mathsf{series}(\mathsf{dp}_1,\mathsf{dp}_2)} : \mathcal{F}_1 &\to \mathsf{A}\mathcal{R}_2, \\ \mathsf{f}_1 &\mapsto \mathrm{Min}_{\preceq_{\mathcal{R}_2}} \uparrow h_{\mathsf{dp}_2}(h_{\mathsf{dp}_1}(\mathsf{f}_1)). \end{aligned}$$

CTC:

$$[\mathrm{dp}_2 \circ \mathrm{dp}_1](a, c) = \bigvee_{b, b' \in B, b \leq b'} [\mathrm{dp}_1](a, b) \wedge [\mathrm{dp}_2](b', c).$$

“Parallel”

High-level: the “parallel” of two design problems is still a design problem

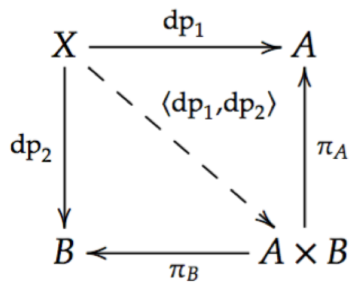


MTC:

$$\begin{aligned} h_{\mathsf{par}(\mathsf{dp}_1,\mathsf{dp}_2)} : \mathcal{F}_1 \times \mathcal{F}_2 &\to \mathsf{A}(\mathcal{R}_1 \times \mathcal{R}_2) \\ \mathsf{f}_1 \times \mathsf{f}_2 &\mapsto h_{\mathsf{dp}_1}(\mathsf{f}_1) \times h_{\mathsf{dp}_2}(\mathsf{f}_2) \end{aligned}$$

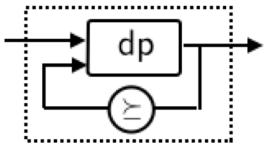
CTC:

$$[\langle \mathrm{dp}_A, \mathrm{dp}_B \rangle](x, a, b) = [\mathrm{dp}_1](x, a) \wedge [\mathrm{dp}_2](x, b).$$



"Loop"

High-level: the "loop" of a design problem is still a design problem



MTC:

$$\begin{aligned} h_{\mathsf{loop(dp)}} : \mathcal{F}_1 &\rightarrow \mathsf{A}\mathcal{R} \\ \mathsf{f}_1 &\mapsto \operatorname*{Min}_{\preceq_{\mathcal{R}}} \mathsf{lfp}[S \mapsto h_{\mathsf{dp}}(\mathsf{f}_1, S)], \\ & \qquad\qquad S \in \mathsf{A}\mathcal{R} \end{aligned}$$

CTC:

$$[\mathrm{Tr}^{C}_{A,B}(\mathrm{dp})](a, b) = \bigvee_{c \in C} [\mathrm{dp}](a, c, b, c)$$

How do we compute design problems?

Given the **minimal** functionality required, what are the **minimal** sets of resources which provide it?

Computing from "atomic" design problems

High-level: co-design problems are computable, and have exact solutions

MTC: proof by recursion, since every primitive design problem (excepting loop) has an exact solution

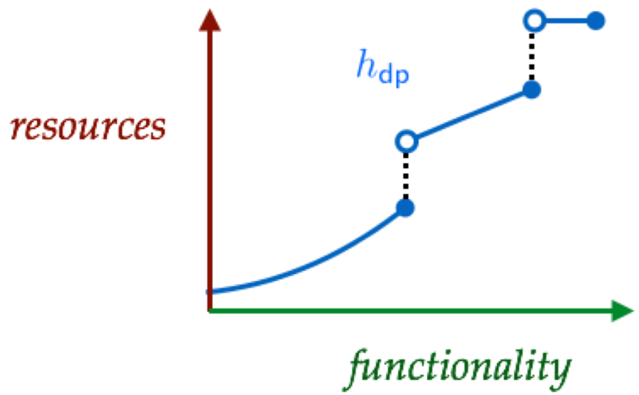
CTC: this is a direct property of **DP**, where

$$[\mathrm{dp}_2 \circ \mathrm{dp}_1](a, c) = \bigcup_{\substack{(b,b') \in B \times B^{\mathrm{op}} \\ b \leq_B b'}} \Big[[\mathrm{dp}_1](a, b) \times [\mathrm{dp}_2](b', c) \Big].$$

Dealing with loops, pt. 1

High-level: to compute a (unique) least fixed point for h, we need to use an algorithm called Kleene ascent, and to run Kleene ascent, we need to assume that h is *Scott-continuous*

MTC:



✓ **Scott-continuous**

(monotone and left-continuous)

CTC: there is a category $\mathbf{DP_S}$ with objects DCPOs and morphisms Scott-continuous functions; alternately, it is the subcategory of **DP** whose morphisms preserve all sequential colimits

Kleene ascent

High-level: start from bot, iterate until you get to the least fixed point

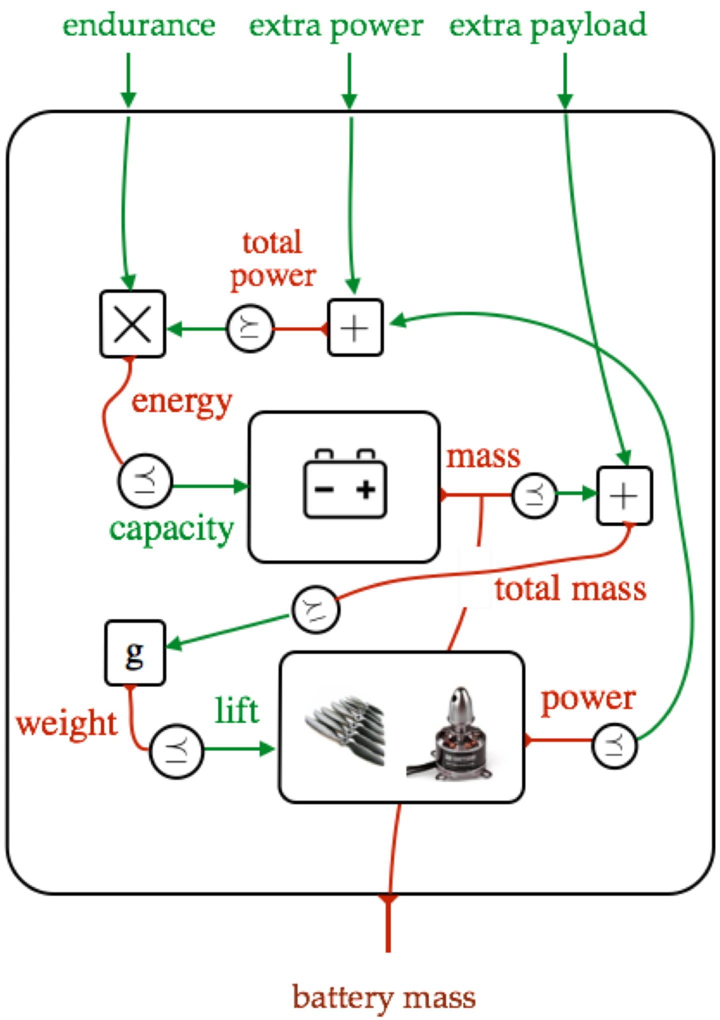
MTC:

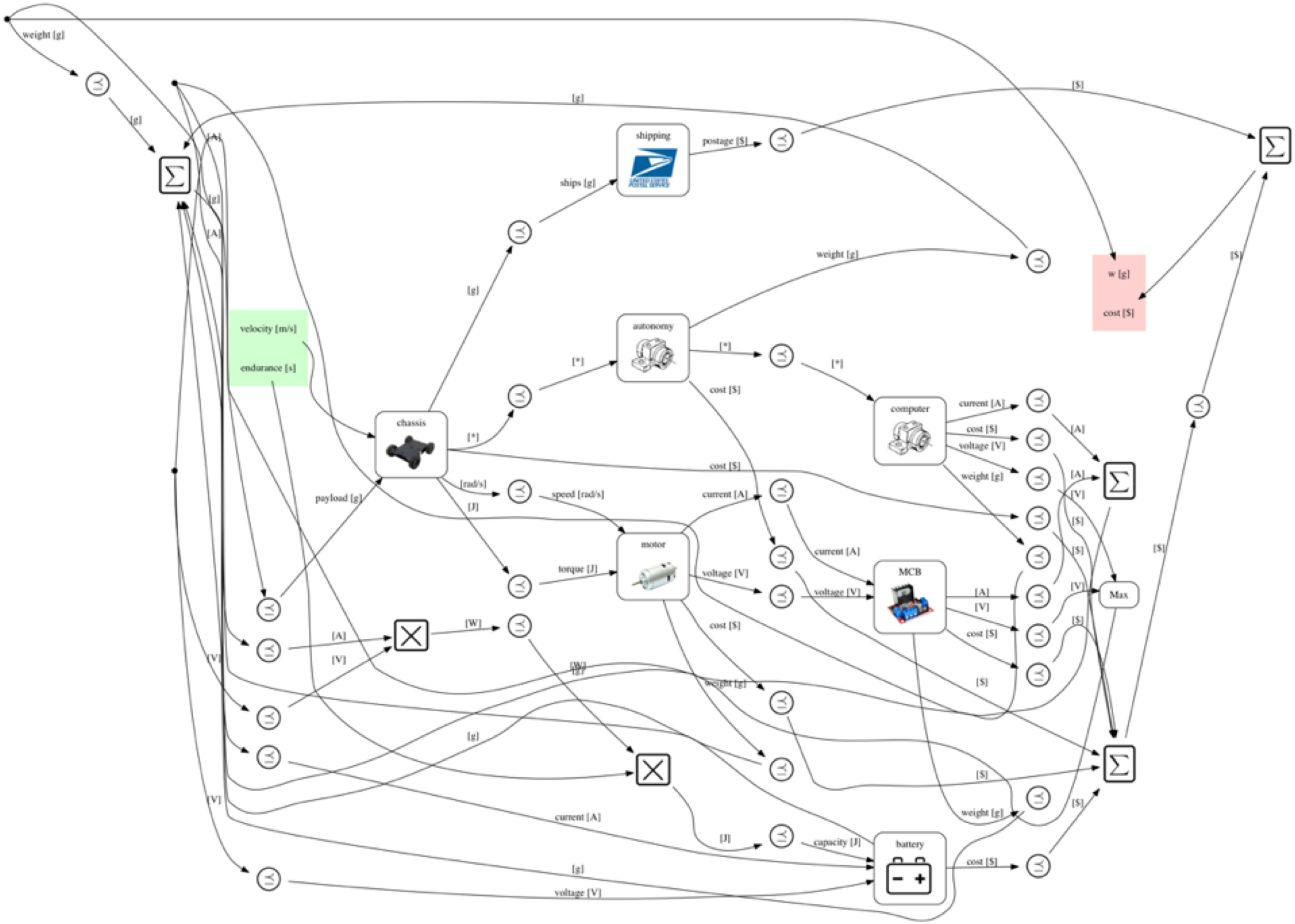
$$\begin{aligned} \Phi_{\mathsf{f}_1} : \mathsf{A}\mathcal{R} &\rightarrow \mathsf{A}\mathcal{R} \\ S &\mapsto \operatorname*{Min}_{\preceq_{\mathcal{R}}} \bigcup_{\mathsf{r} \in S} h_{\mathsf{dp}}(\mathsf{f}_1, \mathsf{r}) \cap \uparrow \mathsf{r} \end{aligned}$$

CTC: Adamek's theorem for algebras over an endofunctor (whiteboard)

How can we make computation more tractable? (in progress)

```
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
}
```





weight [g]
[g]
[A]
[g]
[A]
velocity [m/s]
endurance [s]
chassis
payload [g]
[*]
[rad/s]
[J]
shipping
ships [g]
postage [$]
[$]
autonomy
[*]
cost [$]
weight [g]
computer
current [A]
cost [$]
voltage [V]
weight [g]
w [g]
cost [$]
motor
speed [rad/s]
torque [J]
current [A]
voltage [V]
cost [$]
weight [g]
MCB
current [A]
voltage [V]
[A]
[V]
cost [$]
Max
[W]
[V]
[J]
battery
capacity [J]
current [A]
voltage [V]
weight [g]
cost [$]

Minimize the upper bound

High-level: the complexity of computing any design problem is proportional to the width and the height of the resource poset

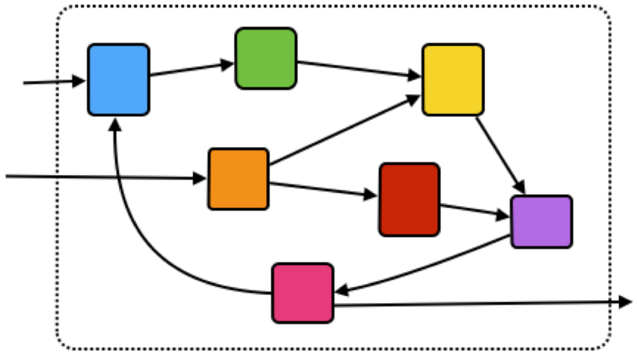
MTC: recall,

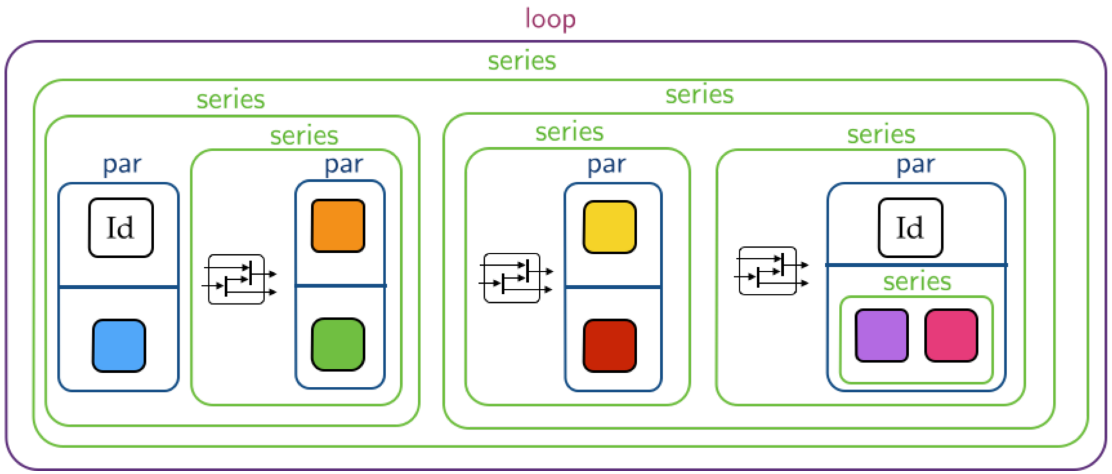
$$\begin{aligned} \Phi_{\mathsf{f}_1} : \mathsf{A}\mathcal{R} &\rightarrow \mathsf{A}\mathcal{R} \\ S &\mapsto \underset{\preceq_{\mathcal{R}}}{\mathrm{Min}} \bigcup_{\mathsf{r} \in S} h_{\mathsf{dp}}(\mathsf{f}_1, \mathsf{r}) \cap \uparrow \mathsf{r} \end{aligned}$$

The upper bound of this algorithm is **width(R) × height($\mathcal{A}$R) × c**, where c = # of times h_{dp} must be evaluated, to a max of width(R) times

Graph rewrite problem

MTC: any co-design diagram can be rewritten as a tree with leaves the primitive design problems and junctions given by "series", "parallel", and "loop". The problem: what tree is best?





CTC: how do we minimize the cost of clustering a hypergraph?

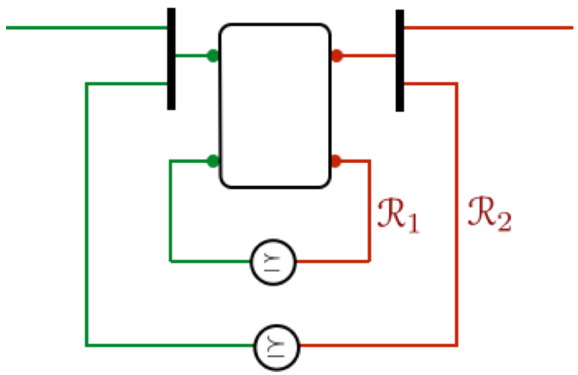
http://mathoverflow.net/questions/247852/minimizing-the-cost-of-clustering-a-hypergraph

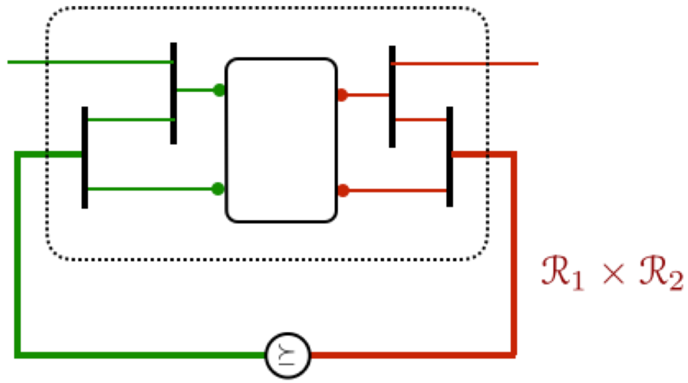
Dealing with loops pt. 2

High-level: any dp with two (nested) loops can be reduced to a dp with one loop

MTC: direct proof

CTC: “vanishing II” axiom for trace





What else can we do with this theory?

+ other future work

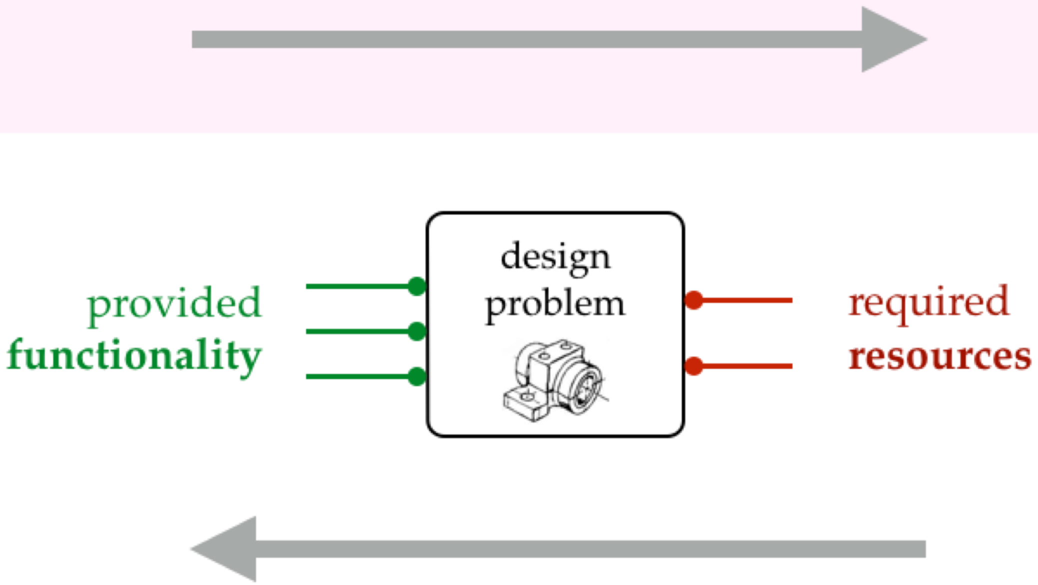
Compare with convex optimization

MCDP		**Convex Optimization**
DCPOs	***objects***	convex sets
monotone maps	***morphisms***	convex functions
composition of monotone maps is monotone	***compositional structure***	composition of convex functions is convex
Kleene ascent	***algorithms***	gradient descent

Reverse arrows

CTC: DP^{op}??

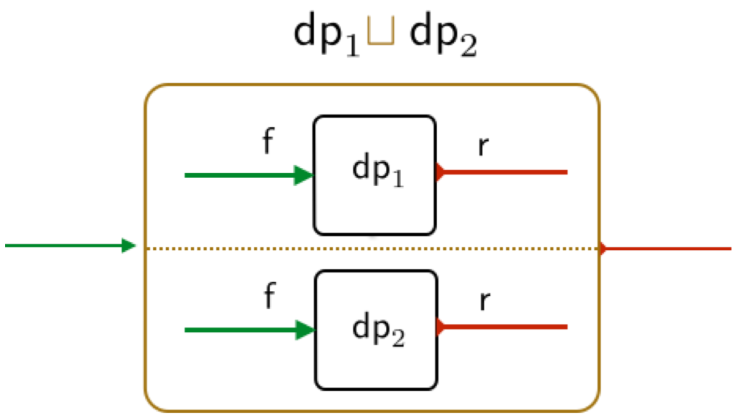
- **Given the minimal functionality** to be provided, what are the **minimal resources** required?



- **Given the maximal resources** that are available, what is the **maximal functionality** that can be provided?

Coproducts

High-level: choose between two different technologies!

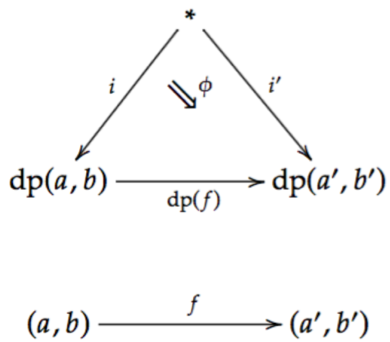


CTC: The disjoint union exists and is the coproduct in **DP**

The Grothendieck construction

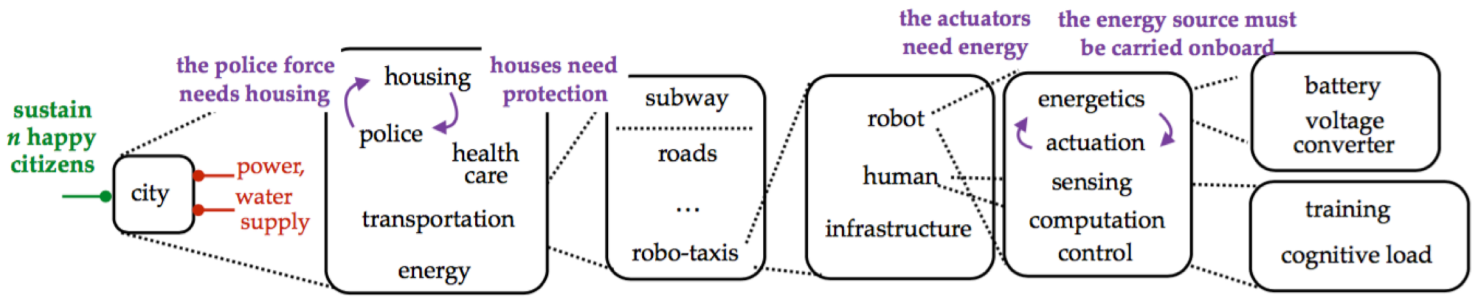
High-level: a category where morphisms are the individual implementations

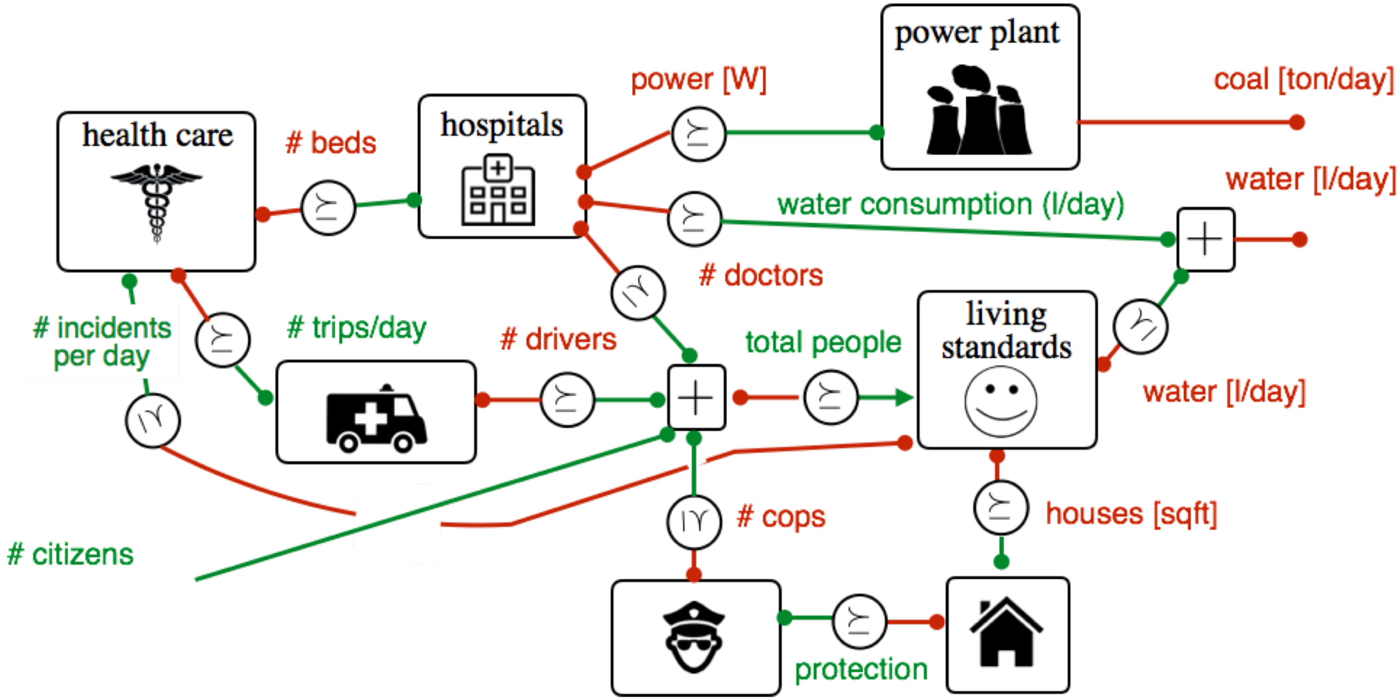
CTC: The *Grothendieck construction* on a design problem [dp] : $A^{op} \times B \to$ Set is a category $\int$ dp with objects (a,b,i) where $i \in$ dp(a, b) and morphisms (f , ϕ) : (a, b, i) → (a', b', i') satisfying the following:



Other future work

- A "differential" on design problems so that we can model questions like "in which design problem should I invest time or R&D grants"?
- Examples beyond robotics?





power plant
coal [ton/day]
power [W]
health care
beds
hospitals
water [l/day]
water consumption (l/day)
doctors
living
standards
incidents
per day
trips/day
drivers
total people
water [l/day]
cops
houses [sqft]
citizens
protection

Thank you!

For more, see mcdp.mit.edu.