A Mathematical Theory of Co-Design

Andrea Censi

***Abstract*—One of the challenges of modern engineering, and robotics in particular, is designing complex systems, composed of many subsystems, rigorously and with optimality guarantees. This paper introduces a theory of co-design that describes "design problems", defined as tuples of "functionality space", "implementation space", and "resources space", together with a feasibility relation that relates the three spaces. Design problems can be interconnected together to create "co-design problems", which describe possibly recursive co-design constraints among subsystems. A co-design problem induces a family of optimization problems of the type "find the minimal resources needed to implement a given functionality"; the solution is an antichain (Pareto front) of resources. A special class of co-design problems are Monotone Co-Design Problems (MCDPs), for which functionality and resources are complete partial orders and the feasibility relation is monotone and Scott continuous. The induced optimization problems are multi-objective, nonconvex, nondifferentiable, noncontinuous, and not even defined on continuous spaces; yet, there exists a complete solution. The antichain of minimal resources can be characterized as a least fixed point, and it can be computed using Kleene's algorithm. The computation needed to solve a co-design problem can be bounded by a function of a graph property that quantifies the interdependence of the subproblems. These results make us much more optimistic about the problem of designing complex systems in a rigorous way.**

I. Introduction

ONE of the great engineering challenge of this century is dealing with the design of "complex" systems. A complex system is complex because its components cannot be decoupled; otherwise, it would be just a (simple) product of simple systems. The *design* of a complex system is complicated because of the "co-design constraints", which are the constraints that one subsystem induces on another. This paper is an attempt towards formalizing and systematically solving the problem of "co-design" of complex systems with recursive design constraints.

Robotic systems as the prototype of complex systems: Robotics is the prototypical example of a field that includes heterogeneous multi-domain co-design constraints. The design of a robotic system involves the choice of physical components, such as the actuators, the sensors, the power supply, the computing units, the network links, etc. Not less important is the choice of the software components, including perception, planning, and control modules. All these components induce co-design constraints on each other. Each physical component has SWAP characteristics such as its shape (which must contained somewhere), weight (which adds to the payload), power (which needs to be provided by something else), excess heat (which must be dissipated somehow), etc. Analogously, the software components have similar co-design constraints. For example, a planner needs a state estimate. An estimator provides a state estimate, and requires the data from a sensor, which requires the presence of a sensor, which requires power. Everything costs money to buy or develop or license.

Andrea Censi <censi@mit.edu> is with the Laboratory for Information and Decision Systems (LIDS) at the Massachusetts Institute of Technology.

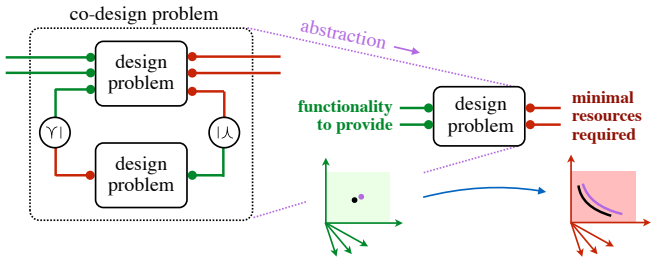


Figure 1. A *design problem* is a relation that relates the implementations available to the functionality provided and the resources required, both represented as partially ordered sets. A *co-design problem* is the interconnection of two or more design problems. An edge in a co-design diagram like in the figure represent a *co-design constraint*: the resources required by the first design problem are a lower bound for the functionality to be provided by the second. The optimization problem to be solved is: find the solutions that are minimal in resources usage, given a lower bound on the functionality to be provided.

What makes system design problems non trivial is that the constraints might be recursive. This is a form of *feedback* in the problem of design (Fig. 1). For example, a battery provides power, which is used by actuators to carry the payload. A larger battery provides more power, but it also increases the payload, so more power is needed. Extremely interesting trade-offs arise when considering constraints between the mechanical system and the embodied intelligence. For control, typically a better state estimate saves energy in the execution, but requires better sensors (which increase the cost and the payload) or better computation (which increases the power consumption).

Contribution: A Principled Theory of Co-Design: This paper describes a theory to deal with arbitrarily complex co-design problems in a principled way. A *design problem* is defined as a tuple of functionality space, implementation space, and a resources space, plus the two maps that relate an implementation to functionality provided and resources required. A design problem defines a family of optimization problems of the type "find the minimal resources needed to implement a given functionality". A *co-design problem* is an interconnection of design problems according to an arbitrary graph structure, including feedback connections. Monotone Co-Design Problems (MCDPs) are the composition of design problems for which both functionality and resources are complete partial orders, and the relation between functionality implemented and resources needed is monotone (order-preserving) and Scott continuous. The first main result in this paper (Theorem 1 on page 9) is that the class of MCDPs is closed with respect to interconnection. The second main result (Theorem 2 on page 10) is that there exists a systematic procedure to solve an MCDP, assuming there is a procedure to solve the primitive design problems. The solution of an MCDP—a Pareto front, or "antichain" of minimal resources—can be found by solving a least fixed point

iteration in the space of antichains. The complexity of this iteration depends on the structure of the co-design diagram.

This paper is a generalization of previous work [1], where the interconnection was limited to one cycle. A conference version of this work appeared as [2].

Outline: Sec. II recalls necessary background about partial orders. Sec. III defines co-design problems. Sec. IV contains a brief statement of results. Sec. V describes composition operators for design problems. Sec. VI shows how any interconnection of design problems can be described using three composition operators (series, parallel, feedback). Sec. VII describes the invariance of a monotonicity property that is preserved by the composition operators. Sec. VIII describes solution algorithms for MCDPs. Sec. IX shows numerical examples. Sec. X discusses related work.

II. Background

We will use basic facts about order theory. Davey and Priestley [3] and Roman [4] are possible reference texts.

Let $\langle \mathcal{P}, \preceq_{\mathcal{P}} \rangle$ be a partially ordered set (poset), which is a set $\mathcal{P}$ together with a partial order $\preceq_{\mathcal{P}}$ (a reflexive, antisymmetric, and transitive relation). The partial order "$\preceq_{\mathcal{P}}$" is written as "$\preceq$" if the context is clear. If a poset has a least element, it is called "bottom" and it is denoted by $\bot_{\mathcal{P}}$. If the poset has a maximum element, it is called "top" and denoted as $\top_{\mathcal{P}}$.

Chains and antichains: A *chain* $x \preceq y \preceq z \preceq \dots$ is a subset of a poset in which all elements are comparable. An *antichain* is a subset of a poset in which *no* elements are comparable. This is the mathematical concept that formalizes the idea of "Pareto front".

Definition 1 (Antichains)**.** A subset $S \subseteq \mathcal{P}$ is an antichain iff no elements are comparable: for $x, y \in S$, $x \preceq y$ implies $x = y$.

Call $\mathsf{A}\mathcal{P}$ the set of all antichains in $\mathcal{P}$. By this definition, the empty set is an antichain: $\emptyset \in \mathsf{A}\mathcal{P}$.

Definition 2 (Width and height of a poset)**.** $\mathsf{width}(\mathcal{P})$ is the maximum cardinality of an antichain in $\mathcal{P}$ and $\mathsf{height}(\mathcal{P})$ is the maximum cardinality of a chain in $\mathcal{P}$.

Minimal elements: Uppercase "Min" will denote the *minimal* elements of a set. The minimal elements are the elements that are not dominated by any other in the set. Lowercase "min" denotes *the least* element, an element that dominates all others, if it exists. (If $\min S$ exists, then $\mathrm{Min}\, S = \{\min S\}$.)

The set of minimal elements of a set are an antichain, so Min is a map from the power set $\mathcal{P}(\mathcal{P})$ to the antichains $\mathsf{A}\mathcal{P}$:

$$\begin{aligned} \mathrm{Min}\colon \mathcal{P}(\mathcal{P}) &\to \mathsf{A}\mathcal{P}, \\ S &\mapsto \{x \in S :\ (y \in S) \wedge (y \preceq x) \Rightarrow (x = y)\ \}. \end{aligned}$$

Max and max are similarly defined.

Upper sets: An "upper set" is a subset of a poset that is closed upward.

Definition 3 (Upper sets)**.** A subset $S \subseteq \mathcal{P}$ is an upper set iff $x \in S$ and $x \preceq y$ implies $y \in S$.

Call $\mathsf{U}\mathcal{P}$ the set of upper sets of $\mathcal{P}$. By this definition, the empty set is an upper set: $\emptyset \in \mathsf{U}\mathcal{P}$.

Lemma 1. $\mathsf{U}\mathcal{P}$ is a poset itself, with the order given by

$$A \preceq_{\mathsf{U}\mathcal{P}} B \quad \equiv \quad A \supseteq B. \tag{1}$$

Note in (1) the use of "$\supseteq$" instead of "$\subseteq$", which might seem more natural. This choice will make things easier later.

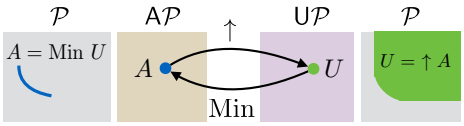
In the poset $\langle \mathsf{U}\mathcal{P}, \preceq_{\mathsf{U}\mathcal{P}} \rangle$, the top is the empty set, and the bottom is the entire poset $\mathcal{P}$.

Order on antichains: The upper closure operator "$\uparrow$" maps a subset of a poset to an upper set.

Definition 4 (Upper closure)**.** The operator $\uparrow$ maps a subset to the smallest upper set that includes it:

$$\begin{aligned} \uparrow : \mathcal{P}(\mathcal{P}) &\to \mathsf{U}\mathcal{P}, \\ S &\mapsto \{y \in \mathcal{P} : \exists x \in S : x \preceq y\}. \end{aligned}$$

Figure 2.



By using the upper closure operator, we can define an order on antichains using the order on the upper sets (Fig. 2).

Lemma 2. $\mathsf{A}\mathcal{P}$ is a poset with the relation $\preceq_{\mathsf{A}\mathcal{P}}$ defined by

$$A \preceq_{\mathsf{A}\mathcal{P}} B \quad \equiv \quad \uparrow A \supseteq \uparrow B.$$

In the poset $\langle \mathsf{A}\mathcal{P}, \preceq_{\mathsf{A}\mathcal{P}} \rangle$, the top is the empty set: $\top_{\mathsf{A}\mathcal{P}} = \emptyset$. If a bottom for $\mathcal{P}$ exists, then the bottom for $\mathsf{A}\mathcal{P}$ is the singleton containing only the bottom for $\mathcal{P}$: $\bot_{\mathsf{A}\mathcal{P}} = \{\bot_{\mathcal{P}}\}$.

Monotonicity and fixed points: We will use Kleene's theorem, a celebrated result that is used in disparate fields. It is used in computer science for defining denotational semantics (see, e.g., [5]). It is used in embedded systems for defining the semantics of models of computation (see, e.g., [6]).

Definition 5 (Directed set)**.** A set $S \subseteq \mathcal{P}$ is *directed* if each pair of elements in S has an upper bound: for all $a, b \in S$, there exists $c \in S$ such that $a \preceq c$ and $b \preceq c$.

Definition 6 (Completeness)**.** A poset is a *directed complete partial order* (DCPO) if each of its directed subsets has a supremum (least of upper bounds). It is a *complete partial order* (CPO) if it also has a bottom.

Example 1 (Completion of $\mathbb{R}_+$ to $\overline{\mathbb{R}}_+$)**.** The set of real numbers $\mathbb{R}$ is not a CPO, because it lacks a bottom. The nonnegative reals $\mathbb{R}_+ = \{x \in \mathbb{R} \mid x \geq 0\}$ have a bottom $\bot = 0$, however, they are not a DCPO because some of their directed subsets do not have an upper bound. For example, take $\mathbb{R}_+$, which is a subset of $\mathbb{R}_+$. Then $\mathbb{R}_+$ is directed, because for each $a, b \in \mathbb{R}_+$, there exists $c = \max\{a, b\} \in \mathbb{R}_+$ for which $a \leq c$ and $b \leq c$. One way to make $\langle \mathbb{R}_+, \leq \rangle$ a CPO is by adding an artificial top element $\top$, by defining $\overline{\mathbb{R}}_+ \triangleq \mathbb{R}_+ \cup \{\top\}$, and extending the partial order $\leq$ so that $a \leq \top$ for all $a \in \mathbb{R}^+$.

Two properties of maps that will be important are monotonicity and the stronger property of Scott continuity.

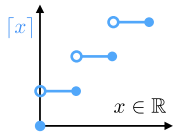
Definition 7 (Monotonicity)**.** A map $f\colon \mathcal{P} \to \mathcal{Q}$ between two posets is *monotone* iff $x \preceq_{\mathcal{P}} y$ implies $f(x) \preceq_{\mathcal{Q}} f(y)$.

Definition 8 (Scott continuity)**.** A map $f : \mathcal{P} \to \mathcal{Q}$ between DCPOs is *Scott continuous* iff for each directed subset $D \subseteq \mathcal{P}$, the image $f(D)$ is directed, and $f(\sup D) = \sup f(D)$.

Remark 1. Scott continuity implies monotonicity.

Remark 2. Scott continuity does not imply topological continuity. A map from the CPO $\langle \overline{\mathbb{R}}_+, \leq \rangle$ to itself is Scott continuous iff it is nondecreasing and left-continuous. For example, the ceiling function $x \mapsto \lceil x \rceil$ is Scott continuous (Fig. 3).

Figure 3.



A *fixed point* of $f : \mathcal{P} \to \mathcal{P}$ is a point x such that $f(x) = x$.

Definition 9. A *least fixed point* of $f : \mathcal{P} \to \mathcal{P}$ is the minimum (if it exists) of the set of fixed points of f:

$$\mathsf{lfp}(f) \doteq \min_{\preceq} \{x \in \mathcal{P} \colon f(x) = x\}. \tag{2}$$

The equality in (2) can be relaxed to "$\preceq$".

The least fixed point need not exist. Monotonicity of the map f plus completeness is sufficient to ensure existence.

Lemma 3 ([3, CPO Fixpoint Theorem II, 8.22])**.** If $\mathcal{P}$ is a CPO and $f : \mathcal{P} \to \mathcal{P}$ is monotone, then $\mathsf{lfp}(f)$ exists.

With the additional assumption of Scott continuity, Kleene's algorithm is a systematic procedure to find the least fixed point.

Lemma 4 (Kleene's fixed-point theorem [3, CPO fixpoint theorem I, 8.15])**.** Assume $\mathcal{P}$ is a CPO, and $f : \mathcal{P} \to \mathcal{P}$ is Scott continuous. Then the least fixed point of f is the supremum of the Kleene ascent chain

$$\bot \preceq f(\bot) \preceq f(f(\bot)) \preceq \cdots \preceq f^{(n)}(\bot) \leq \cdots .$$

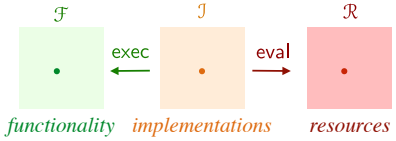
III. Co-Design Problems

The basic objects considered in this paper are "design problems", of which several classes will be investigated. We start by defining a "design problem with implementation", which is a tuple of "functionality space", "implementation space", and "resources space", together with two maps that describe the feasibility relations between these three spaces (Fig. 4).

Definition 10. A *design problem with implementation* (DPI) is a tuple $\langle \mathcal{F}, \mathcal{R}, \mathcal{I}, \mathsf{exec}, \mathsf{eval} \rangle$ where:

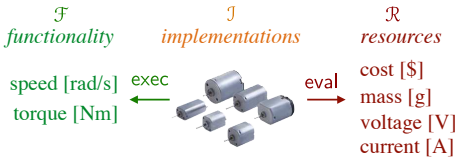
- $\mathcal{F}$ is a poset, called *functionality space*;
- $\mathcal{R}$ is a poset, called *resources space*;
- $\mathcal{I}$ is a set, called *implementation space*;
- the map $\mathsf{exec} \colon \mathcal{I} \to \mathcal{F}$, mnemonics for "execution", maps an implementation to the functionality it provides;
- the map $\mathsf{eval} \colon \mathcal{I} \to \mathcal{R}$, mnemonics for "evaluation", maps an implementation to the resources it requires.

Figure 4.



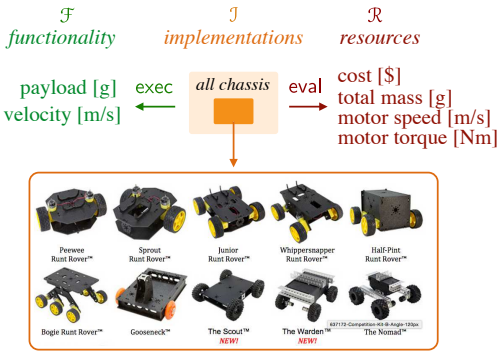
Example 2 (Motor design)**.** Suppose we need to choose a motor for a robot from a given set. The *functionality* of a motor could be parametrized by torque and speed. The *resources* to consider could include the cost [$], the mass [g], the input voltage [V], and the input current [A]. The map $\mathsf{exec} : \mathcal{I} \to \mathcal{F}$ assigns to each motor its functionality, and the map $\mathsf{eval} : \mathcal{I} \to \mathcal{R}$ assigns to each motor the resources it needs (Fig. 14).

Figure 5.



Example 3 (Chassis design)**.** Suppose we need to choose a chassis for a robot (Fig. 6). The implementation space $\mathcal{I}$ could be the set of all chassis that could ever be designed (in case of a theoretical analysis), or just the set of chassis available in the catalogue at hand (in case of a practical design decision). The functionality of a chassis could be formalized as "the ability to transport a certain payload [g]" and "at a given speed [m/s]". More refined functional requirements would include maneuverability, the cargo volume, etc. The resources to consider could be the cost [$] of the chassis; the total mass; and, for each motor to be placed in the chassis, the required speed [rad/s] and torque [Nm].

Figure 6.



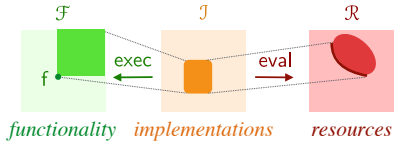
chassis available at ServoCity.com

1) Querying a DPI: A DPI is a model that induces a family of optimization problems, of the type "Given a lower bound on the functionality f, what are the implementations that have minimal resources usage?" (Fig. 7).

Problem 1. Given $\mathsf{f} \in \mathcal{F}$, find the implementations in $\mathcal{I}$ that realize the functionality f (or higher) with minimal resources, or provide a proof that there are none:

$$\begin{cases} \text{using} & \mathsf{i} \in \mathcal{I}, \\ \mathrm{Min}_{\preceq_{\mathcal{R}}} & \mathsf{r}, \\ \text{s.t.} & \mathsf{r} = \mathsf{eval}(\mathsf{i}), \\ & \mathsf{f} \preceq_{\mathcal{F}} \mathsf{exec}(\mathsf{i}). \end{cases} \tag{3}$$

Figure 7.

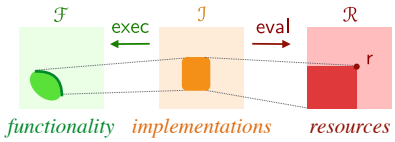


Remark 3 (Minimal *vs* least solutions)**.** Note the use of "$\mathrm{Min}_{\preceq_{\mathcal{R}}}$" in (3), which indicates the set of minimal (non-dominated) elements according to $\preceq_{\mathcal{R}}$, rather than "$\min_{\preceq_{\mathcal{R}}}$", which would presume the existence of a least element. In all problems in this paper, the goal is to find the optimal trade-off of resources ("Pareto front"). So, for each f, we expect to find an antichain $R \in \mathsf{A}\mathcal{R}$. We will see that this formalization allows an elegant way to treat multi-objective optimization. The algorithm to be developed will directly solve for the set R, without resorting to techniques such as *scalarization*, and therefore is able to work with arbitrary posets, possibly discrete.

Remark 4 (Dual formulation)**.** In an entirely symmetric fashion, we could fix an upper bound on the resources usage, and then maximize the functionality provided (Fig. 8). The formulation is entirely dual, in the sense that it is obtained from (3) by swapping Min with Max, $\mathcal{F}$ with $\mathcal{R}$, and exec with eval.

$$\begin{cases} \text{using} & \mathsf{i} \in \mathcal{I}, \\ \mathrm{Max}_{\preceq_{\mathcal{F}}} & \mathsf{f}, \\ \text{s.t.} & \mathsf{f} = \mathsf{exec}(\mathsf{i}), \\ & \mathsf{r} \succeq_{\mathcal{R}} \mathsf{eval}(\mathsf{i}). \end{cases} \tag{4}$$

Figure 8.



2) The functionality-to-minimal resources map h: It is useful to also describe a design problem as a map from functionality to sets of resources that abstracts over implementations.

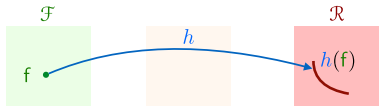
(A useful analogy is the state space representation *vs* the transfer function representation of a linear time-invariant system: the state space representation is richer, but we only need the transfer function to characterize the input-output response.)

Definition 11. Given a DPI $\langle \mathcal{F}, \mathcal{R}, \mathcal{I}, \mathsf{exec}, \mathsf{eval} \rangle$, define the map $h : \mathcal{F} \to \mathsf{A}\mathcal{R}$ that associates to each functionality f the objective function of Problem 1, which is the set of minimal resources necessary to realize f:

$$\begin{aligned} h : \mathcal{F} &\to \mathsf{A}\mathcal{R}, \\ \mathsf{f} &\mapsto \operatorname*{Min}_{\preceq_{\mathcal{R}}} \{\mathsf{eval}(\mathsf{i}) \mid (\mathsf{i} \in \mathcal{I}) \wedge (\mathsf{f} \preceq \mathsf{exec}(\mathsf{i}))\}. \end{aligned}$$

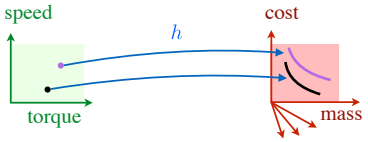
If a certain functionality f is infeasible, then $h(\mathsf{f}) = \emptyset$.

Figure 9.



Example 4. In the case of the motor design problem, the map h assigns to each pair of $\langle \mathsf{speed}, \mathsf{torque} \rangle$ the achievable trade-off of cost, mass, and other resources (Fig. 10). The antichains are depicted as continuous curves, but they could also be composed by a finite set of points.

Figure 10.



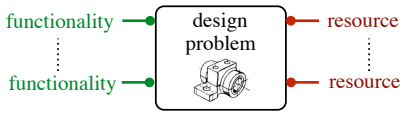
By construction, h is monotone (Def. 7), which means that

$$\mathsf{f}_1 \preceq_{\mathcal{F}} \mathsf{f}_2 \quad \Rightarrow \quad h(\mathsf{f}_1) \preceq_{\mathsf{A}\mathcal{R}} h(\mathsf{f}_2),$$

where $\preceq_{\mathsf{A}\mathcal{R}}$ is the order on antichains defined in Lemma 2. Monotonicity of h means that if the functionality f is increased the antichain of resources will go "up" in the poset of antichains $\mathsf{A}\mathcal{R}$, and at some point it might reach the top of $\mathsf{A}\mathcal{R}$, which is the empty set, meaning that the problem is not feasible.

3) Co-design problems: A graphical notation will help reasoning about composition. A DPI is represented as a box with nf green edges and nr red edges (Fig. 11).

Figure 11.

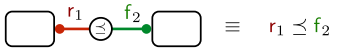


This means that the functionality and resources spaces can be factorized in nf and nr components: $\mathcal{F} = \prod_{i=1}^{\mathsf{nf}} \pi_i \mathcal{F}_i$, $\mathcal{R} = \prod_{j=1}^{\mathsf{nr}} \pi_j \mathcal{R}$, where "$\pi_i$" represents the projection to the i-th component. If there are no green (respectively, red) edges, then nf (respectively, nr) is zero, and $\mathcal{F}$ (respectively, $\mathcal{R}$) is equal to $\mathbb{1} = \{\langle\rangle\}$, the set containing one element, the empty tuple $\langle\rangle$.

These *co-design diagrams* are not to be confused with signal flow diagrams, in which the boxes represent oriented systems and the edges represent signals.

A "co-design problem" will be defined as a multigraph of design problems. Graphically, one is allowed to connect only edges of different color. This interconnection is indicated with the symbol "$\preceq$" in a rounded box (Fig. 12).

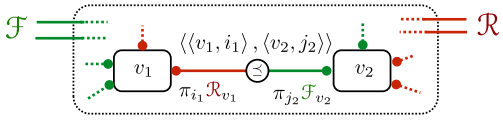
Figure 12.



The semantics of the interconnection is that the resources required by the first DPI are provided by the second DPI. This is a partial order inequality constraint of the type $\mathsf{r}_1 \preceq \mathsf{f}_2$.

Definition 12. A *Co-Design Problem with Implementation* (CDPI) is a tuple $\langle \mathcal{F}, \mathcal{R}, \langle \mathcal{V}, \mathcal{E} \rangle \rangle$, where $\mathcal{F}$ and $\mathcal{R}$ are two posets, and $\langle \mathcal{V}, \mathcal{E} \rangle$ is a multigraph of DPIs. Each node $v \in \mathcal{V}$ is a DPI $v = \langle \mathcal{F}_v, \mathcal{R}_v, \mathcal{I}_v, \mathsf{exec}_v, \mathsf{eval}_v \rangle$. An edge $e \in \mathcal{E}$ is a tuple $e = \langle \langle v_1, i_1 \rangle , \langle v_2, j_2 \rangle \rangle$, where $v_1, v_2 \in \mathcal{V}$ are two nodes and i_1 and j_2 are the indices of the components of the functionality and resources to be connected, and it holds that $\pi_{i_1} \mathcal{R}_{v_1} = \pi_{j_2} \mathcal{F}_{v_2}$ (Fig. 13).

Figure 13.



A CDPI is equivalent to a DPI with an implementation space $\mathcal{I}$ that is a subset of the product $\prod_{v \in \mathcal{V}} \mathcal{I}_v$, and contains only the tuples that satisfy the co-design constraints. An implementation tuple $\mathsf{i} \in \prod_{v \in \mathcal{V}} \mathcal{I}_v$ belongs to $\mathcal{I}$ iff it respects

all functionality–resources constraints on the edges, in the sense that, for all edges $\langle\langle v_1, i_1\rangle, \langle v_2, j_2\rangle\rangle$ in $\mathcal{E}$, it holds that

$$\pi_{i_1}\mathsf{eval}_{v_1}(\pi_{v_1}\mathsf{i}) \preceq \pi_{j_2}\mathsf{exec}_{v_2}(\pi_{v_2}\mathsf{i}).$$

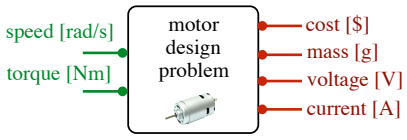
The posets $\mathcal{F}, \mathcal{R}$ for the entire CDPI are the products of the functionality and resources of the nodes that remain unconnected. For a node v, let UF_v and UR_v be the set of unconnected functionalities and resources. Then $\mathcal{F}$ and $\mathcal{R}$ for the CDPI are defined as the product of the unconnected functionality and resources of all DPIs: $\mathcal{F} = \prod_{v\in\mathcal{V}}\prod_{j\in\mathsf{UF}_v}\pi_j\mathcal{F}_v$ and $\mathcal{R} = \prod_{v\in\mathcal{V}}\prod_{i\in\mathsf{UR}_v}\pi_i\mathcal{R}_v$. The maps $\mathsf{exec}, \mathsf{eval}$ return the values of the unconnected functionality and resources:

$$\mathsf{exec} : \mathsf{i} \mapsto \prod_{v\in\mathcal{V}}\prod_{j\in\mathsf{UF}_v}\pi_j\mathsf{exec}_v(\pi_v\mathsf{i}),$$

$$\mathsf{eval} : \mathsf{i} \mapsto \prod_{v\in\mathcal{V}}\prod_{i\in\mathsf{UR}_v}\pi_i\mathsf{eval}_v(\pi_v\mathsf{i}).$$

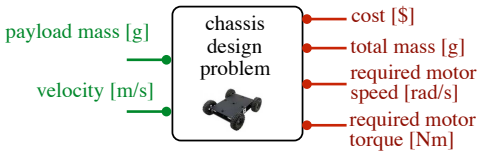
Example 5. Consider the co-design of chassis (Example 3) plus motor (Example 2). The design problem for a motor has speed and torque as the provided functionality (what the motor must provide), and cost, mass, voltage, and current as the required resources (Fig. 14).

Figure 14.



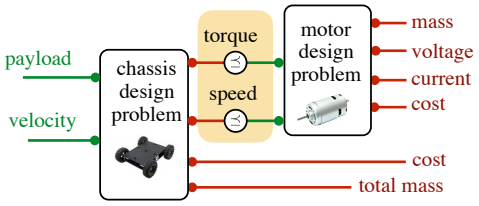
For the chassis (Fig. 15), the provided functionality is parameterized by the mass of the payload and the platform velocity. The required resources include the cost, total mass, and what the chassis needs from its motor(s), such as speed and torque.

Figure 15.



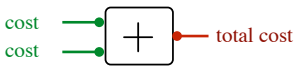
The two design problem can be connected at the edges for torque and speed (Fig. 16). The semantics is that the motor needs to have *at least* the given torque and speed.

Figure 16.



Resources can be summed together using a trivial DP corresponding to the map $h : \langle \mathsf{f}_1, \mathsf{f}_2\rangle \mapsto \{\mathsf{f}_1 + \mathsf{f}_2\}$ (Fig. 17).

Figure 17.



A co-design problem might contain recursive co-design constraints. For example, if we set the payload to be transported to be the sum of the motor mass plus some extra payload, a cycle appears in the graph (Fig. 18).

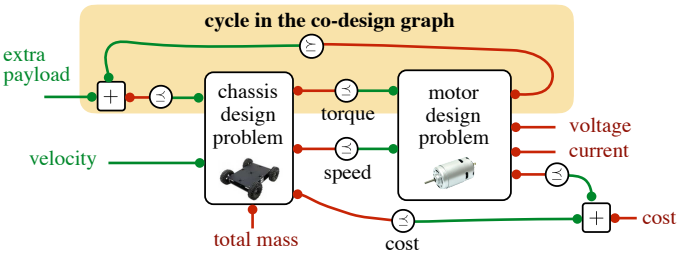
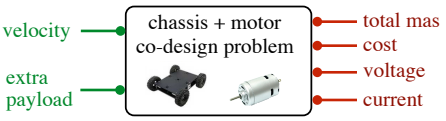


Figure 18.

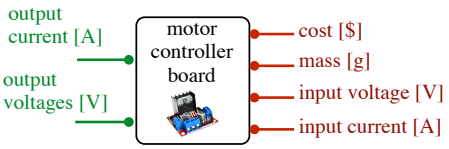
This formalism makes it easy to abstract away the details in which we are not interested. Once a diagram like Fig. 18 is obtained, we can draw a box around it and consider the abstracted problem (Fig. 19).

Figure 19.



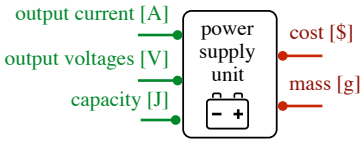
Let us finish assembling our robot. A motor needs a motor control board. The functional requirements are the (peak) output current and the output voltage range (Fig. 20).

Figure 20.



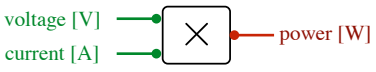
The functionality for a power supply could be parameterized by the output current, the output voltages, and the capacity. The resources could include cost and mass (Fig. 21).

Figure 21.



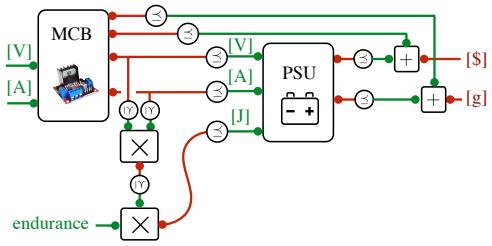
Relations such as current $\times$ voltage $\leq$ power required and power $\times$ endurance $\leq$ energy required can be modeled by a trivial "multiplication" DPI (Fig. 22).

Figure 22.



We can connect these DPs to obtain a co-design problem with functionality voltage, current, endurance and resources mass and cost (Fig. 23).

Figure 23.



Draw a box around the diagram, and call it "MCB+PSU"; then interconnect it with the "chassis+motor" diagram in Fig. 24.

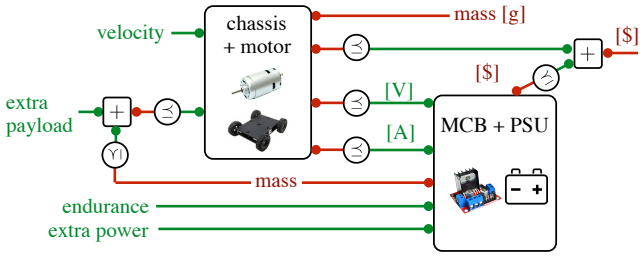


Figure 24.

We can further abstract away the diagram in Fig. 24 as a "mobility+power" CDPI, as in Fig. 25. The formalism allows to consider mass and cost as independent resources, meaning that we wish to obtain the Pareto frontier for the minimal resources. Of course, one can always reduce everything to a scalar objective. For example, a conversion from mass to cost exists and it is called "shipping". Depending on the destination, the conversion factor is between $\$0.5/\text{lbs}$, using USPS, to $\$10\text{k}/\text{lbs}$ for sending your robot to low Earth orbit.

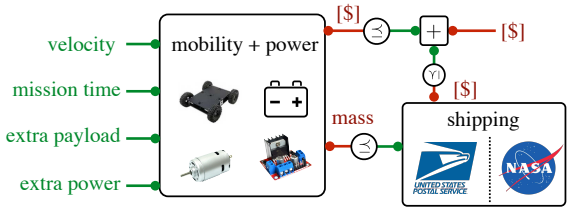
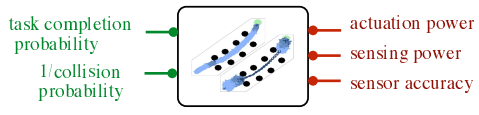


Figure 25.

Examples from the literature: Many recent works in robotics and neighboring fields that deal with minimality and resource constraints can be incorporated in this framework.

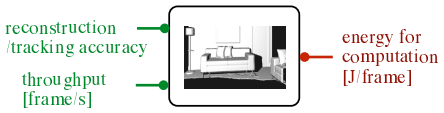
Example 6. Svorenova et al. [7] consider a joint sensor scheduling and control synthesis problem, in which a robot can decide to not perform sensing to save power, given performance objectives on the probability of reaching the target and the probability of collision. The method outputs a Pareto frontier of all possible operating points. This can be cast as a design problem with functionality equal to the probability of reaching the target and (the inverse of) the collision probability, and with resources equal to the actuation power, sensing power, and sensor accuracy.

Figure 26.



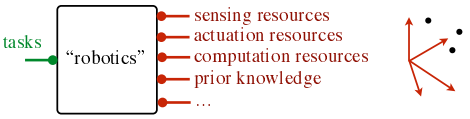
Example 7. Nardi et al. [8] describe a benchmarking system for visual SLAM that provides the empirical characterization of the monotone relation between the accuracy of the visual SLAM solution, the throughput [frames/s] and the energy for computation [J/frame]. The implementation space is the product of algorithmic parameters, compiler flags, and architecture choices, such as the number of GPU cores active. This is an example of a design problem whose functionality-resources map needs to be experimentally evaluated.

Figure 27.



Other examples in minimal robotics: Many works have sought to find "minimal" designs for robots, and can be understood as characterizing the relation between the poset of tasks and the poset of physical resources, which is the product of sensing, actuation, and computation resources, plus other non-physical resources, such as prior knowledge (Fig. 28). Given a task, there is a minimal antichain in the resources poset that describes the possible trade-offs (e.g., compensating lousier sensors with more computation).

Figure 28.



The poset structure arises naturally: for example, in the *sensor lattice* [9], a sensor dominates another if it induces a finer partition of the state space. Similar dominance relations can be defined for actuation and computation. O'Kane and Lavalle [10] define a robot as a union of "robotic primitives", where each primitive is an abstraction for a set of sensors, actuators, and control strategies that can be used together (e.g., a compass plus a contact sensor allow to "drive North until a wall is hit"). The effect of each primitive is modeled as an operator on the robot's information space. It is possible to work out what are the minimal combinations of robotic primitives (minimal antichain) that are sufficient to perform a task (e.g., global localization), and describe a dominance relation (partial order) of primitives. Other works have focused on minimizing the complexity of the controller. Egerstedt [11] studies the relation between the complexity of the environment and a notion of minimum description length of control strategies, which can be taken as a proxy for the computation necessary to perform the task. Soatto [12] studies the relation between the performance of a visual task, and the minimal representation that is needed to perform that task.

The hope is that the theory of co-design presented in this paper will help to integrate all this previous work in the same theoretical and quantitative framework.

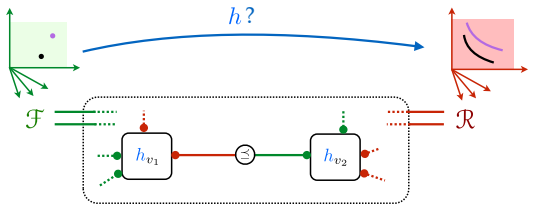
IV. Problem statement and summary of results

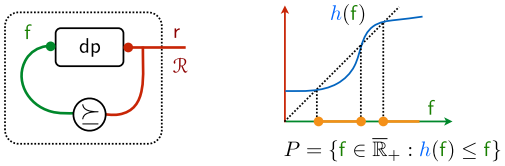
Given an arbitrary graph of design problems, and assuming we know how to solve each problem separately, we ask whether we can solve the *co-design* problem optimally.

Problem 2. Suppose that we are given a CDPI $\langle \mathcal{F}, \mathcal{R}, \langle \mathcal{V}, \mathcal{E} \rangle \rangle$, and that we can evaluate the map h_v for all $v \in \mathcal{V}$. Given a required functionality $\mathsf{f} \in \mathcal{F}$, we wish to find the *minimal* resources in $\mathcal{R}$ for which there exists a feasible implementation vector that makes all sub-problems feasible at the same time and all co-design constraints satisfied; or, if none exist, provide a certificate of infeasibility.

In other words, given the maps $\{h_v, v \in \mathcal{V}\}$ for the subproblems, one needs to evaluate the map $h : \mathcal{F} \to \mathsf{A}\mathcal{R}$ for the entire CDPI (Fig. 29).

Figure 29.





(a) (b)

Figure 30. One feedback connection and a topologically continuous h are sufficient to induce a disconnected feasible set.

The rest of the paper will provide a solution to Problem 2, under the assumption that all the DPIs inside the CDPI are "monotone", in the sense of Def. 13.

Definition 13. A DPI $\langle \mathcal{F}, \mathcal{R}, \mathcal{I}, \mathsf{exec}, \mathsf{eval} \rangle$ is "monotone" if

1) The posets $\mathcal{F}, \mathcal{R}$ are complete partial orders (Def. 6).
2) The map h is Scott continuous (Def. 8).

Call "Monotone Co-Design Problems" (MCDPs) the set of CDPIs for which all subproblems respect the conditions in Def. 13. I will show two main results:

1) A **modeling result** (Theorem 1 on page 9) says that the class of MCDPs is closed with respect to arbitrary interconnections. Therefore, given a co-design diagram, such as the one in Fig. 25, if we know that each design problem is an MCDP, we can conclude that the diagram represents an MCDP as well.

2) An **algorithmic result** (Theorem 2 on page 10) says that the functionality-resources map h for the entire MCDP has an explicit expression in terms of the maps $\{h_v, v \in \mathcal{V}\}$ for the subproblems. If there are cycles in the co-design diagram, the map h involves the solution of a *least fixed point* equation in the space of antichains. This equation can be solved using Kleene's algorithm to find the antichain containing all minimal solutions at the same time.

Approach: The strategy to obtain these results consists in reducing an arbitrary interconnection of design problems to considering only a finite number of composition operators (series, parallel, and feedback). Sec. V defines these composition operators. Sec. VI shows how to turn a graph into a tree, where each junction is one of the three operators. Given the tree representation of an MCDPs, we will be able to give inductive arguments to prove the results.

Expressivity of MCDPs: The results are significant because MCDPs induce a rich family of optimization problems.

We are not assuming, let alone strong properties like convexity, even weaker properties like differentiability or continuity of the constraints. In fact, we are not even assuming that functionality and resources are continuous spaces; they could be arbitrary discrete posets. (In that case, completeness and Scott continuity are trivially satisfied.)

Moreover, even assuming topological continuity of all spaces and maps considered, MCDPs are strongly not convex. What makes them nonconvex is the possibility of introducing feedback interconnections. To show this, I will give an example of a 1-dimensional problem with a continuous h for which the feasible set is disconnected.

Example 8. Consider the CDPI in Fig. 30a. The minimal resources $M \subseteq \mathsf{A}\mathcal{R}$ are the objectives of this optimization problem:

$$M \doteq \begin{cases} \text{using} & \mathsf{f}, \mathsf{r} \in \mathcal{F} = \mathcal{R}, \\ \text{Min}_{\preceq} & \mathsf{r}, \\ & \mathsf{r} \in h(\mathsf{f}), \\ & \mathsf{r} \preceq \mathsf{f}. \end{cases}$$

The feasible set $\Phi \subseteq \mathcal{F} \times \mathcal{R}$ is the set of functionality and resources that satisfy the constraints $\mathsf{r} \in h(\mathsf{f})$ and $\mathsf{r} \preceq \mathsf{f}$:

$$\Phi = \{\langle \mathsf{f}, \mathsf{r} \rangle \in \mathcal{F} \times \mathcal{R} : (\mathsf{r} \in h(\mathsf{f})) \wedge (\mathsf{r} \preceq \mathsf{f})\}. \tag{5}$$

The projection P of Φ to the functionality space is:

$$P = \{\mathsf{f} \mid \langle \mathsf{f}, \mathsf{r} \rangle \in \Phi\}.$$

In the scalar case ($\mathcal{F} = \mathcal{R} = \langle \overline{\mathbb{R}}_+, \leq \rangle$), the map $h \colon \mathcal{F} \to \mathsf{A}\mathcal{R}$ is simply a map $h \colon \overline{\mathbb{R}}_+ \to \overline{\mathbb{R}}_+$. The set P of feasible functionality is described by

$$P = \{\mathsf{f} \in \overline{\mathbb{R}}_+ : h(\mathsf{f}) \leq \mathsf{f}\}. \tag{6}$$

Fig. 30b shows an example of a continuous map h that gives a disconnected feasible set P. Moreover, P is disconnected under any order-preserving nonlinear re-parametrization.

V. Composition operators for design problems

This section defines a handful of composition operators for design problems. Later, Sec. VI will prove that any co-design problem can be described in terms of a subset of these operators.

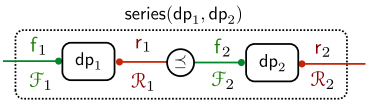
Definition 14 (series)**.** The series composition of two DPIs $\mathsf{dp}_1 = \langle \mathcal{F}_1, \mathcal{R}_1, \mathcal{I}_1, \mathsf{exec}_1, \mathsf{eval}_1 \rangle$ and $\mathsf{dp}_2 = \langle \mathcal{F}_2, \mathcal{R}_2, \mathcal{I}_2, \mathsf{exec}_2, \mathsf{eval}_2 \rangle$, for which $\mathcal{F}_2 = \mathcal{R}_1$, is

$$\mathsf{series}(\mathsf{dp}_1, \mathsf{dp}_2) \doteq \langle \mathcal{F}_1, \mathcal{R}_2, \mathcal{I}, \mathsf{exec}, \mathsf{eval} \rangle,$$

where:

$$\begin{aligned} \mathcal{I} &= \{\langle \mathsf{i}_1, \mathsf{i}_2 \rangle \in \mathcal{I}_1 \times \mathcal{I}_2 \mid \mathsf{eval}_1(\mathsf{i}_1) \preceq_{\mathcal{R}_1} \mathsf{exec}_2(\mathsf{i}_2)\}, \\ \mathsf{exec} &: \langle \mathsf{i}_1, \mathsf{i}_2 \rangle \mapsto \mathsf{exec}_1(\mathsf{i}_1), \\ \mathsf{eval} &: \langle \mathsf{i}_1, \mathsf{i}_2 \rangle \mapsto \mathsf{eval}_2(\mathsf{i}_2). \end{aligned}$$

Figure 31.



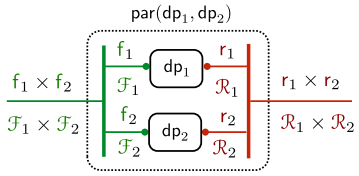
Definition 15 (par)**.** The parallel composition of two DPIs $\mathsf{dp}_1 = \langle \mathcal{F}_1, \mathcal{R}_1, \mathcal{I}_1, \mathsf{exec}_1, \mathsf{eval}_1 \rangle$ and $\mathsf{dp}_2 = \langle \mathcal{F}_2, \mathcal{R}_2, \mathcal{I}_2, \mathsf{exec}_2, \mathsf{eval}_2 \rangle$ is

$$\mathsf{par}(\mathsf{dp}_1, \mathsf{dp}_2) \doteq \langle \mathcal{F}_1 \times \mathcal{F}_2, \mathcal{R}_1 \times \mathcal{R}_2, \mathcal{I}_1 \times \mathcal{I}_2, \mathsf{exec}, \mathsf{eval} \rangle,$$

where:

$$\begin{aligned} \mathsf{exec} &: \langle \mathsf{i}_1, \mathsf{i}_2 \rangle \mapsto \langle \mathsf{exec}_1(\mathsf{i}_1), \mathsf{exec}_2(\mathsf{i}_2) \rangle , \\ \mathsf{eval} &: \langle \mathsf{i}_1, \mathsf{i}_2 \rangle \mapsto \langle \mathsf{eval}_1(\mathsf{i}_1), \mathsf{eval}_2(\mathsf{i}_2) \rangle . \end{aligned} \tag{7}$$

Figure 32.



Definition 16 (loop)**.** Suppose dp is a DPI with factored functionality space $\mathcal{F}_1 \times \mathcal{R}$:

$$\mathsf{dp} = \langle \mathcal{F}_1 \times \mathcal{R}, \mathcal{R}, \mathcal{I}, \langle \mathsf{exec}_1, \mathsf{exec}_2 \rangle , \mathsf{eval} \rangle .$$

Then we can define the DPI loop(dp) as

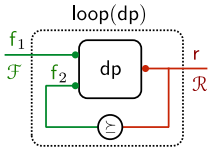
$$\mathsf{loop}(\mathsf{dp}) \doteq \langle \mathcal{F}_1, \mathcal{R}, \mathcal{I}', \mathsf{exec}_1, \mathsf{eval} \rangle ,$$

where $\mathcal{I}' \subseteq \mathcal{I}$ limits the implementations to those that respect the additional constraint $\mathsf{eval}(\mathsf{i}) \preceq \mathsf{exec}_2(\mathsf{i})$:

$$\mathcal{I}' = \{ \mathsf{i} \in \mathcal{I} : \mathsf{eval}(\mathsf{i}) \preceq \mathsf{exec}_2(\mathsf{i}) \}.$$

This is equivalent to "closing a loop" around dp with the constraint $\mathsf{f}_2 \succeq \mathsf{r}$ (Fig. 33).

Figure 33.



The operator loop is asymmetric because it acts on a design problem with 2 functionalities and 1 resources. We can define a symmetric feedback operator loopb as in Fig. 34a, which can be rewritten in terms of loop, using the construction in Fig. 34b.

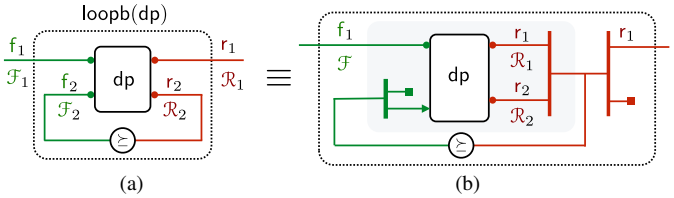


Figure 34. A symmetric operator loopb can be defined in terms of loop.

A "co-product" (see, e.g., [13, Section 2.4]) of two design problems is a design problem with the implementation space $\mathcal{I} = \mathcal{I}_1 \sqcup \mathcal{I}_2$, and it represents the exclusive choice between two possible alternative families of designs.

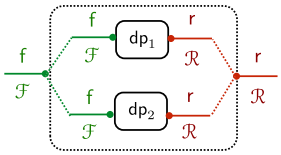
Definition 17 (Co-product)**.** Given two DPIs with same functionality and resources $\mathsf{dp}_1 = \langle \mathcal{F}, \mathcal{R}, \mathcal{I}_1, \mathsf{exec}_1, \mathsf{eval}_1 \rangle$ and $\mathsf{dp}_2 = \langle \mathcal{F}, \mathcal{R}, , \mathcal{I}_2, \mathsf{exec}_2, \mathsf{eval}_2 \rangle$, define their co-product as

$$\mathsf{dp}_1 \sqcup \mathsf{dp}_2 \doteq \langle \mathcal{F}, \mathcal{R}, \mathcal{I}_1 \sqcup \mathcal{I}_2, \mathsf{exec}, \mathsf{eval} \rangle ,$$

where

$$\begin{aligned} \mathsf{exec} &: \mathsf{i} \mapsto \begin{cases} \mathsf{exec}_1(\mathsf{i}), & \text{if } \mathsf{i} \in \mathcal{I}_1, \\ \mathsf{exec}_2(\mathsf{i}), & \text{if } \mathsf{i} \in \mathcal{I}_2, \end{cases} \\ \mathsf{eval} &: \mathsf{i} \mapsto \begin{cases} \mathsf{eval}_1(\mathsf{i}), & \text{if } \mathsf{i} \in \mathcal{I}_1, \\ \mathsf{eval}_2(\mathsf{i}), & \text{if } \mathsf{i} \in \mathcal{I}_2. \end{cases} \end{aligned} \tag{8}$$

Figure 35.



VI. Decomposition of MCDPs

This section shows how to describe an arbitrary interconnection of design problems using only three composition operators. More precisely, for each CDPI with a set of atoms $\mathcal{V}$, there is an equivalent one that is built from series/par/loop applied to the set of atoms $\mathcal{V}$ plus some extra "plumbling" (identities, multiplexers).

Equivalence: The definition of equivalence below ensures that two equivalent DPIs have the same map from functionality to resources, while one of the DPIs can have a slightly larger implementation space.

Definition 18. Two DPIs $\langle \mathcal{F}, \mathcal{R}, \mathcal{I}_1, \mathsf{exec}_1, \mathsf{eval}_1 \rangle$ and $\langle \mathcal{F}, \mathcal{R}, \mathcal{I}_2, \mathsf{exec}_2, \mathsf{eval}_2 \rangle$ are *equivalent* if there exists a map $\varphi : \mathcal{I}_2 \to \mathcal{I}_1$ such that $\mathsf{exec}_2 = \mathsf{exec}_1 \circ \varphi$ and $\mathsf{eval}_2 = \mathsf{eval}_1 \circ \varphi$.

Plumbing: We also need to define "trivial DPIs", which serve as "plumbing". These can be built by taking a map $f : \mathcal{F} \to \mathcal{R}$ and lifting it to the definition of a DPI. The implementation space of a trivial DPI is a copy of the functionality space and there is a 1-to-1 correspondence between functionality and implementation.

Definition 19 (Trivial DPIs)**.** Given a map $f : \mathcal{F} \to \mathcal{R}$, we can lift it to define a trivial DPI $\mathsf{Triv}(f) = \langle \mathcal{F}, \mathcal{R}, \mathcal{F}, \mathsf{Id}_{\mathcal{F}}, f \rangle$, where $\mathsf{Id}_{\mathcal{F}}$ is the identity on $\mathcal{F}$.

Proposition 1. Given a CDPI $\langle \mathcal{F}, \mathcal{R}, \langle \mathcal{V}, \mathcal{E} \rangle \rangle$, we can find an equivalent CDPI obtained by applying the operators par/series/loop to a set of atoms $\mathcal{V}'$ that contains $\mathcal{V}$ plus a set of trivial DPIs. Furthermore, one instance of loop is sufficient.

Proof. We show this constructively. We will temporarily remove all cycles from the graph, to be reattached later. To do this, find an *arc feedback set* (AFS) $F \subseteq \mathcal{E}$. An AFS is a set of edges that, when removed, remove all cycles from the graph (see, e.g., [14]). For example, the CDPI represented in Fig. 36a has a minimal AFS that contains the edge $\mathsf{c} \to \mathsf{a}$ (Fig. 36b).

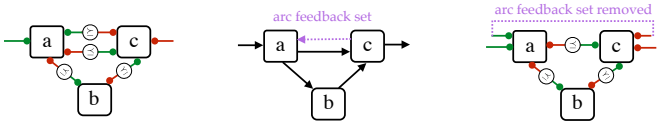
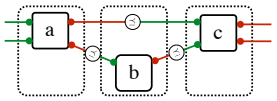


Figure 36. An example co-design diagram with three nodes $\mathcal{V} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, in which a minimal arc feedback set is $\{\mathsf{c} \to \mathsf{a}\}$.

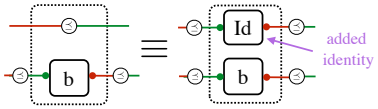
Remove the AFS F from $\mathcal{E}$ to obtain the reduced edge set $\mathcal{E}' = \mathcal{E} \backslash F$. The resulting graph $\langle \mathcal{V}, \mathcal{E}' \rangle$ does not have cycles, and can be written as a series-parallel graph, by applying the operators par and series from a set of nodes $\mathcal{V}'$. The nodes $\mathcal{V}'$ will contain $\mathcal{V}$, plus some extra "connectors" that are trivial DPIs. Find a weak topological ordering of $\mathcal{V}$. Then the graph $\langle \mathcal{V}, \mathcal{E}' \rangle$ can be written as the series of $|\mathcal{V}|$ subgraphs, each containing one node of $\mathcal{V}$. In the example, the weak topological ordering is $\langle \mathsf{a}, \mathsf{b}, \mathsf{c} \rangle$ and there are three subgraphs (Fig. 37).

Figure 37.



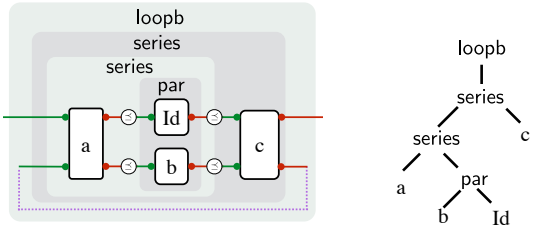
Each subgraph can be described as the parallel interconnection of a node $v \in \mathcal{V}$ and some extra connectors. For example, the second subgraph in the graph can be written as the parallel interconnection of node b and the identity $\mathsf{Triv}(\mathsf{Id})$ (Fig. 38).

Figure 38.



After this is done, we just need to "close the loop" around the edges in the AFS F to obtain a CDPI that is equivalent to the original one. Suppose the AFS F contains only one edge. Then one instance of the loopb operator is sufficient (Fig. 39a). In this example, the tree representation (Fig. 39b) is

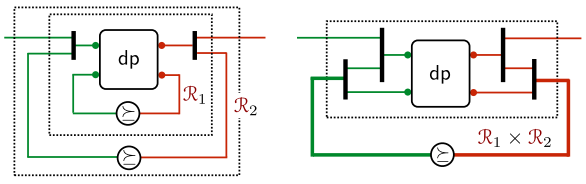
$$\mathsf{loopb}(\mathsf{series}(\mathsf{series}(\mathsf{a}, \mathsf{par}(\mathsf{Id}, \mathsf{b})), \mathsf{c}).$$



(a) (b)

Figure 39. Tree representation for the co-design diagram in Fig. 36a.

If the AFS contains multiple edges, then, instead of closing one loop at a time, one can can always rewrite multiple nested loops as only one loop by taking the product of the edges. For example, a diagram like the one in Fig. 40a can be rewritten as Fig. 40b. This construction is analogous to the construction used for the analysis of process networks [6] (and any other construct involving a traced monoidal category). Therefore, it is possible to describe an arbitrary graph of design problems using only one instance of the loop operator. □



(a) (b)

Figure 40. If there are nested loops in a co-design diagram, they can be rewritten as one loop, by taking the product of the edges.

VII. Monotonicity as compositional property

The first main result of this paper is an invariance result.

Theorem 1. The class of MCDPs is closed with respect to interconnection.

Proof. Prop. 1 has shown that any interconnection of design problems can be described using the three operators par, series, and loop. Therefore, we just need to check that monotonicity in the sense of Def. 13 is preserved by each operator separately. This is done below in Prop. 2–4. □

Proposition 2. If dp_1 and dp_2 are monotone (Def. 13), then also the composition $\mathsf{par}(\mathsf{dp}_1, \mathsf{dp}_2)$ is monotone.

Proof. We need to refer to the definition of par in Def. 15 and check the conditions in Def. 13. If $\mathcal{F}_1, \mathcal{F}_2, \mathcal{R}_1, \mathcal{R}_2$ are CPOs, then $\mathcal{F}_1 \times \mathcal{F}_2$ and $\mathcal{R}_1 \times \mathcal{R}_2$ are CPOs as well.

From Def. 11 and (7) we know h can be written as

$$\begin{aligned} h : \mathcal{F}_1 \times \mathcal{F}_2 &\to \mathsf{A}(\mathcal{R}_1 \times \mathcal{R}_2) \\ \langle \mathsf{f}_1, \mathsf{f}_2 \rangle &\mapsto \operatorname*{Min}_{\preceq_{\mathcal{R}}} \{ \langle \mathsf{eval}_1(\mathsf{i}_1), \mathsf{eval}_2(\mathsf{i}_2) \rangle \mid \\ &\qquad (\langle \mathsf{i}_1, \mathsf{i}_2 \rangle \in \mathcal{I}_1 \times \mathcal{I}_2) \\ &\qquad \wedge \ (\langle \mathsf{f}_1, \mathsf{f}_2 \rangle \preceq \langle \mathsf{exec}_1(\mathsf{i}_1), \mathsf{exec}_2(\mathsf{i}_2) \rangle) \}. \end{aligned}$$

All terms factorize in the two components, giving:

$$\begin{aligned} h : \langle \mathsf{f}_1, \mathsf{f}_2 \rangle \mapsto \ & \operatorname*{Min}_{\mathcal{R}_1} \{ \langle \mathsf{eval}_1(\mathsf{i}_1) \rangle \mid (\mathsf{i} \in \mathcal{I}_1) \wedge (\mathsf{f}_1 \preceq \mathsf{exec}_1(\mathsf{i}_1)) \} \\ & \times \operatorname*{Min}_{\mathcal{R}_2} \{ \langle \mathsf{eval}_2(\mathsf{i}_2) \rangle \mid (\mathsf{i} \in \mathcal{I}_2) \wedge (\mathsf{f}_2 \preceq \mathsf{exec}_2(\mathsf{i}_2)) \}, \end{aligned}$$

which reduces to

$$h : \ \langle \mathsf{f}_1, \mathsf{f}_2 \rangle \ \mapsto \ h_1(\mathsf{f}_1) \times h_2(\mathsf{f}_2). \tag{9}$$

The map h is Scott continuous iff h_1 and h_2 are [15, Lemma II.2.8]. □

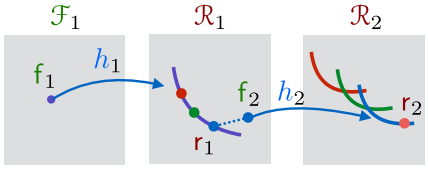
Proposition 3. If dp_1 and dp_2 are monotone (Def. 13), then also the composition $\mathsf{series}(\mathsf{dp}_1, \mathsf{dp}_2)$ is monotone.

Proof. From the definition of series (Def. 14), the semantics of the interconnection is captured by this problem:

$$h : \mathsf{f}_1 \mapsto \begin{cases} \text{using} & \mathsf{r}_1, \mathsf{f}_2 \in \mathcal{R}_1, \quad \mathsf{r}_2 \in \mathcal{R}_2, \\ \operatorname{Min}_{\preceq_{\mathcal{R}_2}} & \mathsf{r}_2, \\ \text{s.t.} & \mathsf{r}_1 \in h_1(\mathsf{f}_1), \\ & \mathsf{r}_1 \preceq_{\mathcal{R}_1} \mathsf{f}_2, \\ & \mathsf{r}_2 \in h_2(\mathsf{f}_2). \end{cases} \tag{10}$$

The situation is described by Fig. 41. The point f_1 is fixed, and thus $h(\mathsf{f}_1)$ is a fixed antichain in $\mathcal{R}_1$. For each point $\mathsf{r}_1 \in h(\mathsf{f}_1)$, we can choose a $\mathsf{f}_2 \succeq \mathsf{r}_1$. For each f_2, the antichain $h_2(\mathsf{f}_2)$ traces the solution in $\mathcal{R}_2$, from which we can choose r_2.

Figure 41.



Because h_2 is monotone, $h_2(\mathsf{f}_2)$ is minimized when f_2 is minimized, hence we know that the constraint $\mathsf{r}_1 \preceq \mathsf{f}_2$ will be tight. We can then conclude that the objective does not change introducing the constraint $\mathsf{r}_1 = \mathsf{f}_2$. The problem is reduced to:

$$h : \mathsf{f}_1 \mapsto \begin{cases} \text{using} & \mathsf{f}_2 \in \mathcal{R}_1, \quad \mathsf{r}_2 \in \mathcal{R}_2, \\ \operatorname{Min}_{\preceq_{\mathcal{R}_2}} & \mathsf{r}_2, \\ \text{s.t.} & \mathsf{f}_2 \in h_1(\mathsf{f}_1), \\ & \mathsf{r}_2 \in h_2(\mathsf{f}_2). \end{cases} \tag{11}$$

Minimizing r_2 with the only constraint being $\mathsf{r}_2 \in h_2(\mathsf{f}_2)$, and with $h_2(\mathsf{f}_2)$ being an antichain, the solutions are all and only $h_2(\mathsf{f}_2)$. Hence the problem is reduced to

$$h : \mathsf{f}_1 \mapsto \begin{cases} \text{using} & \mathsf{f}_2 \in \mathcal{R}_1, \\ \mathrm{Min}_{\preceq_{\mathcal{R}_2}} & h_2(\mathsf{f}_2), \\ \text{s.t.} & \mathsf{f}_2 \in h_1(\mathsf{f}_1). \end{cases} \tag{12}$$

The solution is simply

$$h : \mathsf{f}_1 \mapsto \operatorname*{Min}_{\preceq_{\mathcal{R}_2}} \bigcup_{\mathsf{f}_2 \in h_1(\mathsf{f}_1)} h_2(\mathsf{f}_2). \tag{13}$$

This map is Scott continuous because it is the composition of Scott continuous maps. □

Proposition 4. If dp is monotone (Def. 13), so is $\mathsf{loop}(\mathsf{dp})$.

Proof. The diagram in Fig. 33 implies that the map $h_{\mathsf{loop}(\mathsf{dp})}$ can be described as:

$$h_{\mathsf{loop}(\mathsf{dp})} : \mathcal{F}_1 \to \mathsf{A}\mathcal{R}, \tag{14}$$

$$\mathsf{f}_1 \mapsto \begin{cases} \text{using} & \mathsf{r}, \mathsf{f}_2 \in \mathcal{R}, \\ \mathrm{Min}_{\preceq_{\mathcal{R}}} & \mathsf{r}, \\ \text{s.t.} & \mathsf{r} \in h_{\mathsf{dp}}(\mathsf{f}_1, \mathsf{f}_2), \\ & \mathsf{r} \preceq_{\mathcal{R}} \mathsf{f}_2. \end{cases} \tag{15}$$

Denote by h_{f_1} the map h_{dp} with the first element fixed:

$$h_{\mathsf{f}_1} : \mathsf{f}_2 \mapsto h_{\mathsf{dp}}(\mathsf{f}_1, \mathsf{f}_2).$$

Rewrite $\mathsf{r} \in h_{\mathsf{dp}}(\mathsf{f}_1, \mathsf{f}_2)$ in (14) as

$$\mathsf{r} \in h_{\mathsf{f}_1}(\mathsf{f}_2). \tag{16}$$

Let r be a feasible solution, but not necessarily minimal. Because of Lemma 5, the constraint (16) can be rewritten as

$$\{\mathsf{r}\} = h_{\mathsf{f}_1}(\mathsf{f}_2) \cap {\uparrow}\mathsf{r}. \tag{17}$$

Because $\mathsf{f}_2 \succeq \mathsf{r}$, and h_{f_1} is Scott continuous, it follows that $h_{\mathsf{f}_1}(\mathsf{f}_2) \succeq_{\mathsf{A}\mathcal{R}} h_{\mathsf{f}_1}(\mathsf{r})$. Therefore, by Lemma 6, we have

$$\{\mathsf{r}\} \succeq_{\mathsf{A}\mathcal{R}} h_{\mathsf{f}_1}(\mathsf{r}) \cap {\uparrow}\mathsf{r}. \tag{18}$$

This is a recursive condition that all feasible r must satisfy.

Let $R \in \mathsf{A}\mathcal{R}$ be an antichain of feasible resources, and let r be a generic element of R. Tautologically, rewrite R as the minimal elements of the union of the singletons containing its elements:

$$R = \operatorname*{Min}_{\preceq_{\mathcal{R}}} \bigcup_{\mathsf{r} \in R} \{\mathsf{r}\}. \tag{19}$$

Substituting (18) in (19) we obtain (cf Lemma 7)

$$R \succeq_{\mathsf{A}\mathcal{R}} \operatorname*{Min}_{\preceq_{\mathcal{R}}} \bigcup_{\mathsf{r} \in R} h_{\mathsf{f}_1}(\mathsf{r}) \ \cap \ {\uparrow}\mathsf{r}. \tag{20}$$

[Converse: It is also true that if an antichain R satisfies (20) then all $\mathsf{r} \in R$ are feasible. The constraint (20) means that for any $\mathsf{r}_0 \in R$ on the left side, we can find a r_1 in the right side so that $\mathsf{r}_0 \succeq_{\mathcal{R}} \mathsf{r}_1$. The point r_1 needs to belong to one of the sets of which we take the union; say that it comes from $\mathsf{r}_2 \in R$, so that $\mathsf{r}_1 \in h_{\mathsf{f}_1}(\mathsf{r}_2) \ \cap \ {\uparrow}\mathsf{r}_2$. Summarizing:

$$\forall \mathsf{r}_0 \in R : \ \exists \mathsf{r}_1 : \ (\mathsf{r}_0 \succeq_{\mathcal{R}} \mathsf{r}_1) \ \wedge \ (\exists \mathsf{r}_2 \in R : \ \mathsf{r}_1 \in h_{\mathsf{f}_1}(\mathsf{r}_2) \ \cap \ {\uparrow}\mathsf{r}_2). \tag{21}$$

Because $\mathsf{r}_1 \in h_{\mathsf{f}_1}(\mathsf{r}_2) \cap {\uparrow}\mathsf{r}_2$, we can conclude that $\mathsf{r}_1 \in {\uparrow}\mathsf{r}_2$, and therefore $\mathsf{r}_1 \succeq_{\mathcal{R}} \mathsf{r}_2$, which together with $\mathsf{r}_0 \succeq_{\mathcal{R}} \mathsf{r}_1$, implies $\mathsf{r}_0 \succeq_{\mathcal{R}} \mathsf{r}_2$. We have concluded that there exist two points $\mathsf{r}_0, \mathsf{r}_2$ in the antichain R such that $\mathsf{r}_0 \succeq_{\mathcal{R}} \mathsf{r}_2$; therefore, they are the same point: $\mathsf{r}_0 = \mathsf{r}_2$. Because $\mathsf{r}_0 \succeq_{\mathcal{R}} \mathsf{r}_1 \succeq_{\mathcal{R}} \mathsf{r}_2$, we also conclude that r_1 is the same point as well. We can rewrite (21) by using r_0 in place of r_1 and r_2 to obtain $\forall \mathsf{r}_0 \in R : \mathsf{r}_0 \in h_{\mathsf{f}_1}(\mathsf{r}_0)$, which means that r_0 is a feasible resource.]

We have concluded that all antichains of feasible resources R satisfy (20), and conversely, if an antichain R satisfies (20), then it is an antichain of feasible resources.

Equation (20) is a recursive constraint for R, of the kind

$$\Phi_{\mathsf{f}_1}(R) \preceq_{\mathsf{A}\mathcal{R}} R,$$

with the map Φ_{f_1} defined by

$$\begin{aligned} \Phi_{\mathsf{f}_1} : \mathsf{A}\mathcal{R} &\to \mathsf{A}\mathcal{R}, \\ R &\mapsto \operatorname*{Min}_{\preceq_{\mathcal{R}}} \bigcup_{\mathsf{r} \in R} h_{\mathsf{f}_1}(\mathsf{r}) \ \cap \ {\uparrow}\mathsf{r}. \end{aligned} \tag{22}$$

If we want the *minimal* resources, we are looking for the *least* antichain:

$$\min_{\preceq_{\mathsf{A}\mathcal{R}}} \{ R \in \mathsf{A}\mathcal{R} : \ \Phi_{\mathsf{f}_1}(R) \preceq_{\mathsf{A}\mathcal{R}} R \},$$

which is equal to the *least fixed point* of Φ_{f_1}. Therefore, the map $h_{\mathsf{loop}(\mathsf{dp})}$ can be written as

$$h_{\mathsf{loop}(\mathsf{dp})} : \mathsf{f}_1 \mapsto \mathsf{lfp}(\Phi_{\mathsf{f}_1}). \tag{23}$$

Lemma 8 shows that $\mathsf{lfp}(\Phi_{\mathsf{f}_1})$ is Scott continuous in f_1. □

Lemma 5. Let A be an antichain in $\mathcal{P}$. Then

$$a \in A \quad \equiv \quad \{a\} = A \cap {\uparrow}a.$$

Lemma 6. For $A, B \in \mathsf{A}\mathcal{P}$, and $S \subseteq P$, $A \preceq_{\mathsf{A}\mathcal{R}} B$ implies $A \cap S \preceq_{\mathsf{A}\mathcal{R}} B \cap S$.

Lemma 7. For $A, B, C, D \in \mathsf{A}\mathcal{P}$, $A \preceq_{\mathsf{A}\mathcal{R}} C$ and $B \preceq_{\mathsf{A}\mathcal{R}} D$ implies $A \cup B \preceq_{\mathsf{A}\mathcal{R}} C \cup D$.

Lemma 8. Let $f : \mathcal{P} \times \mathcal{Q} \to \mathcal{Q}$ be Scott continuous. For each $x \in \mathcal{P}$, define $f_x : y \mapsto f(x, y)$. Then $f^\dagger : x \mapsto \mathsf{lfp}(f_x)$ is Scott continuous.

Proof. Davey and Priestly [3] leave this as Exercise 8.26. A proof is found in Gierz et al. [15, Exercise II-2.29]. □

VIII. Solution of MCDPs

The second main result is that the map h of a MCDP has an explicit expression in terms of the maps h of the subproblems.

Theorem 2. The map h for an MCDP has an explicit expression in terms of the maps h of its subproblems, defined recursively using the rules in Table I.

Table I
Recursive expressions for h

series	$\mathsf{dp} = \mathsf{series}(\mathsf{dp}_1, \mathsf{dp}_2)$	$h = h_1 \circledcirc h_2$
parallel	$\mathsf{dp} = \mathsf{par}(\mathsf{dp}_1, \mathsf{dp}_2)$	$h = h_1 \otimes h_2$
feedback	$\mathsf{dp} = \mathsf{loop}(\mathsf{dp}_1)$	$h = h_1^\dagger$
co-product	$\mathsf{dp} = \mathsf{dp}_1 \sqcup \mathsf{dp}_2$	$h = h_1 \circledvee h_2$

Proof. These expressions were derived in the proofs of Prop. 2–4. The operators $\circledcirc, \otimes, \dagger, \circledvee$ are defined in Def. 20–23. □

Definition 20 (Series operator $\circledcirc$)**.** For two maps $h_1 \colon \mathcal{F}_1 \to \mathsf{A}\mathcal{R}_1$ and $h_2 \colon \mathcal{F}_2 \to \mathsf{A}\mathcal{R}_2$, if $\mathcal{R}_1 = \mathcal{F}_2$, define

$$\begin{aligned} h_1 \circledcirc h_2 \colon \mathcal{F}_1 &\to \mathsf{A}\mathcal{R}_2, \\ \mathsf{f}_1 &\mapsto \operatorname*{Min}_{\preceq_{\mathcal{R}_2}} \bigcup_{s \in h_1(\mathsf{f})} h_2(s). \end{aligned}$$

Definition 21 (Parallel operator $\otimes$)**.** For two maps $h_1 \colon \mathcal{F}_1 \to \mathsf{A}\mathcal{R}_1$ and $h_2 \colon \mathcal{F}_2 \to \mathsf{A}\mathcal{R}_2$, define

$$\begin{aligned} h_1 \otimes h_2 : (\mathcal{F}_1 \times \mathcal{F}_2) &\to \mathsf{A}(\mathcal{R}_1 \times \mathcal{R}_2), \\ \langle \mathsf{f}_1, \mathsf{f}_2 \rangle &\mapsto h_1(\mathsf{f}_1) \times h_2(\mathsf{f}_2), \end{aligned} \tag{24}$$

where $\times$ is the product of two antichains.

Definition 22 (Feedback operator $\dagger$)**.** For $h : \mathcal{F}_1 \times \mathcal{R} \to \mathsf{A}\mathcal{R}$, define

$$\begin{aligned} h^\dagger : \mathcal{F}_1 &\to \mathsf{A}\mathcal{R}, \\ \mathsf{f}_1 &\mapsto \mathsf{lfp}\left(\Psi^h_{\mathsf{f}_1}\right), \end{aligned} \tag{25}$$

where $\Psi^h_{\mathsf{f}_1}$ is defined as

$$\begin{aligned} \Psi^h_{\mathsf{f}_1} : \mathsf{A}\mathcal{R} &\to \mathsf{A}\mathcal{R}, \\ R &\mapsto \operatorname*{Min}_{\preceq_{\mathcal{R}}} \bigcup_{\mathsf{r} \in R} h(\mathsf{f}_1, \mathsf{r}) \;\cap \uparrow \mathsf{r}. \end{aligned} \tag{26}$$

Definition 23 (Coproduct operator $\circledvee$)**.** For $h_1, h_2 : \mathcal{F} \to \mathsf{A}\mathcal{R}$, define

$$\begin{aligned} h_1 \circledvee h_2 : \mathcal{F} &\to \mathsf{A}\mathcal{R}, \\ \mathsf{f} &\mapsto \operatorname*{Min}_{\preceq_{\mathcal{R}}} \left(h_1(\mathsf{f}) \cup h_2(\mathsf{f})\right). \end{aligned}$$

A. Example: Optimizing over the natural numbers

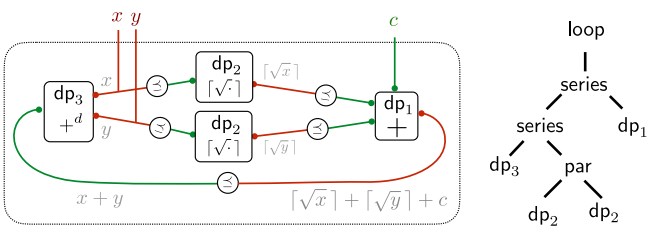
This is the simplest example that can show two interesting properties of MCDPs:

1) the ability to work with discrete posets; and
2) the ability to treat multi-objective optimization problems.

Consider the family of optimization problems indexed by $c \in \mathbb{N}$:

$$\begin{cases} \operatorname{Min}_{\preceq_{\mathbb{N}\times\mathbb{N}}} & \langle x, y \rangle, \\ \text{s.t.} & x + y \geq \lceil \sqrt{x} \rceil + \lceil \sqrt{y} \rceil + c. \end{cases} \tag{27}$$

One can show that this optimization problem is an MCDP by producing a co-design diagram with an equivalent semantics, such as the one in Fig. 42a.



(a)

(b)

Figure 42. Co-design diagram equivalent to (27) and its tree representation.

The diagram contains three primitive DPIs: dp_1, dp_2 (used twice), and dp_3. Their h maps are:

$$\begin{aligned} h_1 : \overline{\mathbb{N}} \times \overline{\mathbb{N}} \times \overline{\mathbb{N}} &\to \mathsf{A}\overline{\mathbb{N}}, \\ \langle \mathsf{f}_1, \mathsf{f}_2, \mathsf{f}_3 \rangle &\mapsto \{\mathsf{f}_1 + \mathsf{f}_2 + \mathsf{f}_3\}, \\ h_2 : \overline{\mathbb{N}} &\to \mathsf{A}\overline{\mathbb{N}}, \\ \mathsf{f} &\mapsto \{\lceil \sqrt{\mathsf{f}} \rceil\}, \\ h_3 : \overline{\mathbb{N}} &\to \mathsf{A}(\overline{\mathbb{N}} \times \overline{\mathbb{N}}), \\ \mathsf{f} &\mapsto \{\langle a, b \rangle \in \overline{\mathbb{N}} \times \overline{\mathbb{N}} : a + b = \mathsf{f}\}. \end{aligned}$$

The tree decomposition (Fig. 42b) corresponds to the expression

$$\mathsf{dp} = \mathsf{loop}(\mathsf{series}(\mathsf{par}(\mathsf{dp}_2, \mathsf{dp}_2), \mathsf{series}(\mathsf{dp}_1, \mathsf{dp}_3))). \tag{28}$$

Consulting Table I, from (28) one obtains an expression for h:

$$h = ((h_2 \otimes h_2) \circledcirc h_1 \circledcirc h_3)^\dagger. \tag{29}$$

This problem is small enough that we can write down an explicit expression for h. By substituting in (29) the definitions given in Def. 20–22, we obtain that evaluating $h(c)$ means finding the least fixed point of a map Ψ_c:

$$h : c \mapsto \mathsf{lfp}(\Psi_c).$$

The map $\Psi_c : \mathsf{A}(\overline{\mathbb{N}} \times \overline{\mathbb{N}}) \to \mathsf{A}(\overline{\mathbb{N}} \times \overline{\mathbb{N}})$ can be obtained from (26) as follows:

$$\Psi_c : R \mapsto \operatorname{Min} \bigcup_{\langle x, y \rangle \in R} \uparrow \langle x, y \rangle \cap$$

$$\cap \left\{ \langle a, b \rangle \in \mathbb{N}^2 : \left(a + b \geq \lceil \sqrt{x} \rceil + \lceil \sqrt{y} \rceil + c\right) \right\}.$$

Kleene's algorithm is the iteration $R_{k+1} = \Psi_c(R_k)$ starting from $R_0 = \bot_{\mathsf{A}(\overline{\mathbb{N}}\times\overline{\mathbb{N}})} = \{\langle 0, 0 \rangle\}$.
For $c = 0$, the sequence converges immediately:

$$R_0 = \{\mathbf{\langle 0, 0 \rangle}\} = h(0).$$

For $c = 1$, the sequence converges at the second step:

$$\begin{aligned} R_0 &= \{\langle 0, 0 \rangle\}, \\ R_1 &= \{\mathbf{\langle 0, 1 \rangle}, \mathbf{\langle 1, 0 \rangle}\} = h(1). \end{aligned}$$

For $c = 2$, the sequence converges at the fourth step; however, some solutions (in bold) converge sooner:

$$\begin{aligned} R_0 &= \{\langle 0, 0 \rangle\}, \\ R_1 &= \{\langle 0, 2 \rangle, \langle 1, 1 \rangle, \langle 2, 0 \rangle\}, \\ R_2 &= \{\mathbf{\langle 0, 4 \rangle}, \langle 2, 2 \rangle, \mathbf{\langle 4, 0 \rangle}\}, \\ R_3 &= \{\mathbf{\langle 0, 4 \rangle}, \mathbf{\langle 3, 3 \rangle}, \mathbf{\langle 4, 0 \rangle}\} = h(2). \end{aligned}$$

The next values in the sequence are:

$$\begin{aligned} h(3) &= \{\mathbf{\langle 0, 6 \rangle}, \mathbf{\langle 3, 4 \rangle}, \mathbf{\langle 4, 3 \rangle}, \mathbf{\langle 6, 0 \rangle}\}, \\ h(4) &= \{\mathbf{\langle 0, 7 \rangle}, \mathbf{\langle 3, 6 \rangle}, \mathbf{\langle 4, 4 \rangle}, \mathbf{\langle 6, 3 \rangle}, \mathbf{\langle 7, 0 \rangle}\}. \end{aligned}$$

Fig. 43 shows the sequence for $c = 20$.

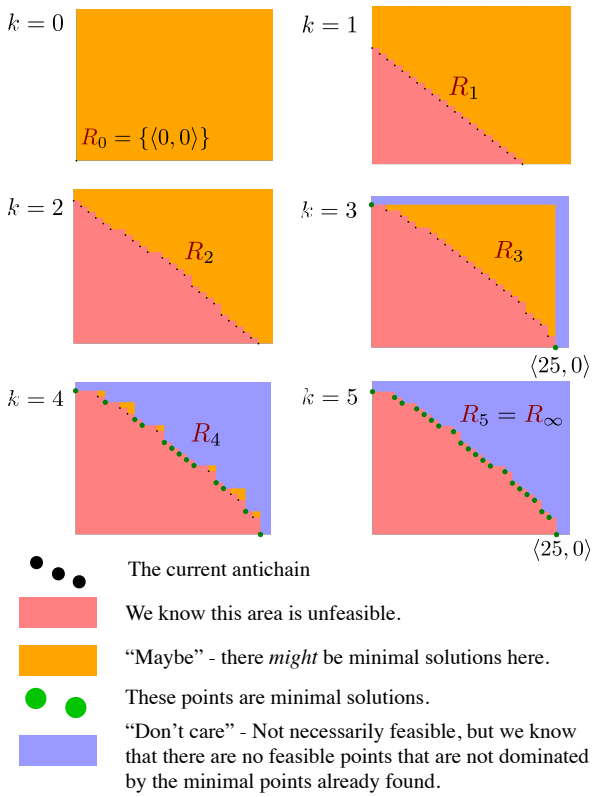


Figure 43. Kleene ascent to solve the problem (27) for $c = 20$. The sequence converges in five steps to $R_5 = R_\infty$.

Guarantees of Kleene ascent: Solving an MCDP with cycles reduces to computing a Kleene ascent sequence R_k. At each instant k we have some additional guarantees.

For any finite k, the resources "below" R_k (the set $\mathcal{R} \setminus\!\uparrow R_k$,) are infeasible. (In Fig. 43, those are colored in red.)

If the iteration converges to a non-empty antichain R_∞, the antichain R_∞ divides $\mathcal{R}$ in two. Below the antichain, all resources are infeasible. However, above the antichain (purple area), it is not necessarily true that all points are feasible, because there might be holes in the feasible set, as in Example 8. Note that this method does not compute the entire feasible set, but rather only the *minimal elements* of the feasible set, which might be much easier to compute.

Finally, if the sequence converges to the empty set, it means that there are no solutions. The sequence R_k can be considered a certificate of infeasibility.

B. Complexity of the solution

1) Complexity of fixed point iteration: Consider first the case of an MCDP that can be described as $\mathsf{dp} = \mathsf{loop}(\mathsf{dp}_0)$, where dp_0 is an MCDP that is described only using the series and par operators. Suppose that dp_0 has resource space $\mathcal{R}$. Then evaluating h for dp is equivalent to computing a least fixed point iteration on the space of antichains $\mathsf{A}\mathcal{R}$. This allows to give worst-case bounds on the number of iterations.

Proposition 5. Suppose that $\mathsf{dp} = \mathsf{loop}(\mathsf{dp}_0)$ and dp_0 has resource space $\mathcal{R}_0$ and evaluating h_0 takes at most c computation. Then we can obtain the following bounds for the algorithm's resources usage:

memory	$O(\mathsf{width}(\mathcal{R}_0))$
number of steps	$O(\mathsf{height}(\mathsf{A}\mathcal{R}_0))$
total computation	$O(\mathsf{width}(\mathcal{R}_0) \times \mathsf{height}(\mathsf{A}\mathcal{R}_0) \times c)$

Proof. The memory utilization is bounded by $\mathsf{width}(\mathcal{R}_0)$, because the state is an antichain, and $\mathsf{width}(\mathcal{R}_0)$ is the size of the largest antichain. The iteration happens in the space $\mathsf{A}\mathcal{R}_0$, and we are constructing an ascending chain, so it can take at most $\mathsf{height}(\mathsf{A}\mathcal{R}_0)$ steps to converge. Finally, in the worst case the map h_0 needs to be evaluated once for each element of the antichain for each step. □

These worst case bounds are strict.

Example 9. Consider solving $\mathsf{dp} = \mathsf{loop}(\mathsf{dp}_0)$ with dp_0 defined by $h_0 \colon \langle \langle\rangle, x\rangle \mapsto x + 1$ with $x \in \overline{\mathbb{N}}$. Then the least fixed point equation is equivalent to solving $\min\{x \colon \Psi(x) \leq x\}$ with $\Psi : x \mapsto x + 1$. The iteration $R_{k+1} = \Psi(R_k)$ converges to $\top$ in $\mathsf{height}(\overline{\mathbb{N}}) = \aleph_0$ steps.

Remark 5. Making more precise claims requires additional more restrictive assumptions on the spaces involved. For example, without adding a metric on $\mathcal{R}$, it is not possible to obtain properties such as linear or quadratic convergence.

Remark 6 (Invariance to re-parameterization)**.** All the results given in this paper are invariant to any order-preserving re-parameterization of all the variables involved.

2) Relating complexity to the graph properties: Prop. 5 above assumes that the MCDP is already in the form $\mathsf{dp} = \mathsf{loop}(\mathsf{dp}_0)$, and relates the complexity to the poset $\mathcal{R}_0$. Here we relate the results to the graph structure of an MCDP.

Take an MCDP $\mathsf{dp} = \langle \mathcal{F}, \mathcal{R}, \langle \mathcal{V}, \mathcal{E}\rangle\rangle$. To put dp in the form $\mathsf{dp} = \mathsf{loop}(\mathsf{dp}_0)$ according to the procedure in Sec. VI, we need to find an arc feedback set (AFS) of the graph $\langle \mathcal{V}, \mathcal{E}\rangle$. Given a AFS $F \subset \mathcal{E}$, then the resource space $\mathcal{R}_0$ for a dp_0 such that $\mathsf{dp} = \mathsf{loop}(\mathsf{dp}_0)$ is the product of the resources spaces along the edges: $\mathcal{R}_0 = \prod_{e \in F} \mathcal{R}_e$.

Now that we have a relation between the AFS and the complexity of the iteration, it is natural to ask what is the optimal choice of AFS—which, so far, was left as an arbitrary choice. The AFS should be chosen as to minimize one of the performance measures in Prop. 5.

Of the three performance measures in Prop. 5, the most fundamental appears to be $\mathsf{width}(\mathcal{R}_0)$, because that is also an upper bound on the number of distinct minimal solutions. Hence we can call it "design complexity" of the MCDP.

Definition 24. Given a graph $\langle \mathcal{V}, \mathcal{E}\rangle$ and a labeling of each edge $e \in \mathcal{E}$ with a poset $\mathcal{R}_e$, the *design complexity* $\mathrm{DC}(\langle \mathcal{V}, \mathcal{E}\rangle)$ is defined as

$$\mathrm{DC}(\langle \mathcal{V}, \mathcal{E}\rangle) = \min_{F \text{ is an AFS}} \mathsf{width}\Big(\prod_{e \in F} \mathcal{R}_e\Big). \tag{30}$$

In general, width and height of posets are not additive with respect to products; therefore, this problem does not reduce to any of the known variants of the minimum arc feedback set problem, in which each edge has a weight and the goal is to minimize the sum of the weights.

3) Considering relations with infinite cardinality: This analysis shows the limitations of the simple solution presented so far: it is easy to produce examples for which $\mathsf{width}(\mathcal{R}_0)$ is infinite, so that one needs to represent a continuum of solutions.

Example 10. Suppose that the platform to be designed must travel a distance d [m], and we need to choose the endurance T [s] and the velocity v [m/s]. The relation among the quantities is $d \leq T\,v$. This is a design problem described by the map

$$\begin{aligned} h : \overline{\mathbb{R}}_+ &\to \mathsf{A}\overline{\mathbb{R}}_+ \times \overline{\mathbb{R}}_+, \\ d &\mapsto \{\langle T, v\rangle \in \overline{\mathbb{R}}_+ \times \overline{\mathbb{R}}_+ : d = T\,v\}. \end{aligned}$$

For each value of d, there is a continuum of solutions.

One approach to solving this problem would be to discretize the functionality $\mathcal{F}$ and the resources $\mathcal{R}$ by sampling and/or coarsening. However, sampling and coarsening makes it hard to maintain completeness and consistency.

One effective approach, outside of the scope of this paper, that allows to use finite computation is to *approximate the design problem itself*, rather than the spaces $\mathcal{F}, \mathcal{R}$, which are left as possibly infinite. The basic idea is that an infinite antichain can be bounded from above and above by two antichains that have a finite number of points. This idea leads to an algorithm that, given a prescribed computation budget, can compute an inner and outer approximation to the solution antichain [16].

IX. Extended Numerical Examples

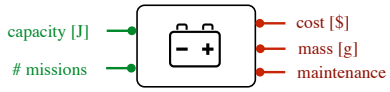
This example considers the choice of different battery technologies for a robot. The goals of this example are: 1) to show how design problems can be composed; 2) to show how to define hard constraints and precedence between resources to be minimized; 3) to show how even relatively simple models can give very complex trade-offs surfaces; and 4) to introduce MCDPL, a formal language for the description of MCDPs.

Language and interpreter/solver: MCDPL is a modeling language to describe MCDPs and their compositions. It is inspired by CVX and "disciplined convex programming" [17]. MCDPL is even more disciplined than CVX; for example, multiplying by a negative number is a *syntax* error. The figures are generated by PyMCDP, an interpreter and solver for MCDPs, which implements the techniques described in this paper. The software and a manual are available at http://mcdp.mit.edu.

Model of a battery: The choice of a battery can be modeled as a DPI (Fig. 44a) with functionalities capacity [J] and number of missions and with resources mass [kg], cost [$] and "maintenance", defined as the number of times that the battery needs to be replaced over the lifetime of the robot.

Each battery technology is described by the three parameters specific energy, specific cost, and lifetime (number of cycles):

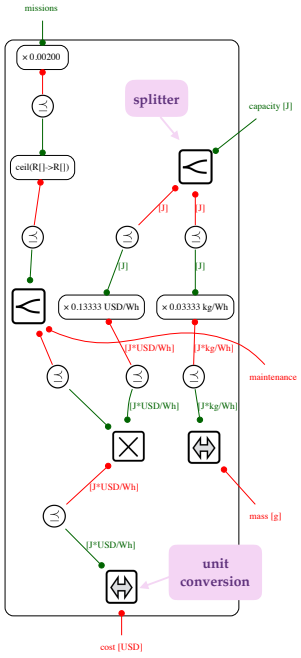
$$\begin{aligned} \rho &\doteq \text{specific energy [Wh/kg]}, \\ \alpha &\doteq \text{specific cost [Wh/\$]}, \\ c &\doteq \text{battery lifetime [\# of cycles]}. \end{aligned}$$



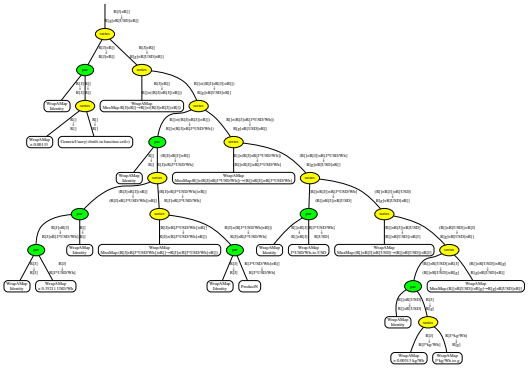
(a) Interface of battery design problem.

```
mcdp {
  provides capacity [J]
  provides missions [R]

  requires mass [g]
  requires cost [$]

  # Number of replacements
  requires maintenance [R]

  # Battery properties
  specific_energy = 30 Wh/kg
  specific_cost = 7.50 Wh/$
  cycles = 500 []

  # Constraint between mass and capacity
  required mass >= provided capacity / specific_energy

  # How many times should it be replaced?
  num_replacements = ceil(provided missions / cycles)
  required maintenance >= num_replacements

  # Cost is proportional to number of replacements
  unit_cost = provided capacity / specific_cost
  required cost >= unit_cost * num_replacements
}
```

(b) MCDPL code equivalent to equations (31)–(33).



(c) Co-design diagram generated by PyMCDP from code in panel (b).



(d) Tree representation using par/series of diagram in panel (c).

Figure 44. Panel (c) shows the co-design diagram generated from the code in (b). Panel (d) shows a tree representation (series, parallel) for the diagram. The edges show the types of functionality and resources. The leaves are labeled with the Python class used internally by the interpreter PyMCDP.

The relation between functionality and resources is described by three nonlinear monotone constraints:

$$\text{mass} \geq \text{capacity}/\rho, \tag{31}$$
$$\text{maintenance} \geq \lceil \text{missions}/c \rceil, \tag{32}$$
$$\text{cost} \geq \lceil \text{missions}/c \rceil\, (\text{capacity}/\alpha). \tag{33}$$

Fig. 44b shows the MCDPL code that describes the model corresponding to (31)–(33). The diagram in Fig. 44c is automatically generated from the code. Fig. 44d shows a tree representation of the diagram using the series/par operators.

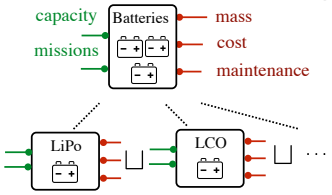
Competing battery technologies: The parameters for the battery technologies used in this example are shown in Table II.

Table II
SPECIFICATIONS OF COMMON BATTERIES TECHNOLOGIES

technology	*energy density* [Wh/kg]	*specific cost* [Wh/$]	*operating life* # cycles
NiMH	100	3.41	500
NiH2	45	10.50	20000
LCO	195	2.84	750
LMO	150	2.84	500
NiCad	30	7.50	500
SLA	30	7.00	500
LiPo	250	2.50	600
LFP	90	1.50	1500

Each row of the table is used to describe a model as in Fig. 44b by plugging in the specific values in lines 12–14.

Given the different models, we can defined their co-product (Fig. 45a) using the MCDPL code in Fig. 45b.



(a) Co-product of battery technologies

```
1  choose(
2     NiMH: `Battery_NiMH,
3     NiH2: `Battery_NiH2,
4      LCO: `Battery_LCO,
5      LMO: `Battery_LMO,
6    NiCad: `Battery_NiCad,
7      SLA: `Battery_SLA,
8    LiPo: `Battery_LiPo,
9      LFP: `Battery_LFP
10 )
```

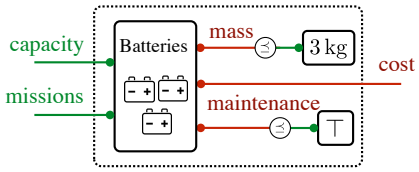
(b) Batteries.mcdp

Figure 45. The co-product of design problems describes the choices among different technologies. The MCDPL keyword for the co-product is "choose".

Introducing other variations or objectives: The design problem for the battery has two functionalities (capacity and number of missions) and three resources (cost, mass, and maintenance). Thus, it describes a family of multi-objective optimization problems, of the type "Given capacity and missions, minimize $\langle \text{cost}, \text{mass}, \text{maintenance} \rangle$". We can further extend the class of optimization problems by introducing other hard constraints and by choosing which resource to prioritize. This can be done by composition of design problems; that is, by creating a larger DP that contains the original DP as a subproblem, and contains some additional degenerate DPs that realize the desired semantics.

For example, suppose that we would like to find the optimal solution(s) such that: 1) The mass does not exceed 3 kg; 2) The mass is minimized as a primary objective, while cost/maintenance are secondary objectives.

This semantics can be described by the co-design diagram in Fig. 46a, which contains two new symbols. The DP labeled "3 kg" implements the semantics of hard constraints. It has one functionality ($\mathcal{F} = \overline{\mathbb{R}}_+^{\text{kg}}$) and zero resources ($\mathcal{R} = \mathbb{1}$). The poset $\mathbb{1} = \{\langle\rangle\}$ has exactly two antichains: $\emptyset$ and $\{\langle\rangle\}$. These



(a) Co-design diagram that expresses hard constraints for mass.

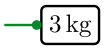
```
1  mcdp {
2    provides capacity [J]
3    provides missions [R]
4
5    requires cost [$]
6
7    battery = instance `Batteries
8
9    provided capacity <= capacity provided by battery
10   provided missions <= missions provided by battery
11
12   mass required by battery <= 3 kg
13
14   ignore maintenance required by battery
15
16   required cost >= cost required by battery
17 }
```

(b) MCDPL code equivalent to diagram in (a).

Figure 46. Composition of MCDPs can express hard constraints and precedence of objectives. In this case, there is a hard constraint on the mass. Because there is only one outgoing edge for mass, and the cost and maintenance are terminated by a dummy constraint ($x \preceq \top$), the semantics of the diagram is that the objective is to minimize the mass as primary objective.

represent "infeasible" and "feasible", respectively. The DP is described by the map

$$h : \overline{\mathbb{R}}_+^{\text{kg}} \to \mathsf{A}\mathbb{1}, \qquad \mathsf{f} \mapsto \begin{cases} \{\langle\rangle\}, & \text{if } \mathsf{f} \leq 3 \text{ kg}, \\ \emptyset, & \text{if } \mathsf{f} > 3 \text{ kg}. \end{cases}$$

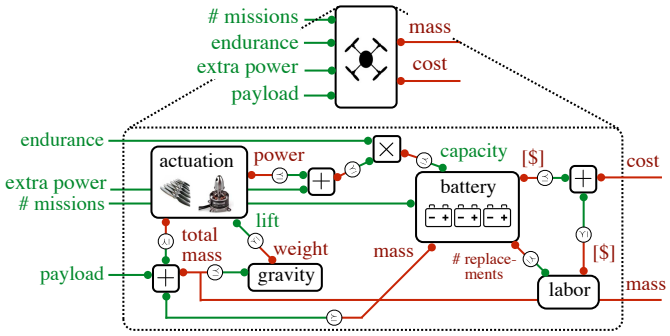


The block labeled "$\top$" is similarly defined and always returns "feasible", so it has the effect of ignoring cost and maintenance as objectives. The only resource edge is the one for mass, which is then the only objective.

The MCDPL code is shown in Fig. 46b. Note the intuitive interface: the user can directly write "mass required by battery $\leq$ 3 kg" and "ignore maintenance required by battery", which is compiled to "maintenance required by battery $\leq \top$".

This relatively simple model for energetics already shows the complexity of MCDPs. Fig. 49 shows the optimal choice of the battery technology as a function of capacity and number of missions, for several slight variations of the problem that differ in constraints and objectives. For each battery technology, the figures show whether at each operating point the technology is the optimal choice, and how many optimal choices there are. Some of the results are intuitive. For example, Fig. 49f shows that if the only objective is minimizing mass, then the optimal choice is simply the technology with largest specific energy (LiPo). The decision boundaries become complex when considering nonlinear objectives. For example, Fig. 49d shows the case where the objective is to minimize the cost, which, defined by (33), is nonlinearly related to both capacity and number of missions. When considering multi-objective problems, such as minimizing jointly $\langle \text{mass}, \text{cost} \rangle$ (Fig. 49h) or $\langle \text{mass}, \text{cost}, \text{maintenance} \rangle$ (Fig. 49h), there are multiple non-dominated solutions.

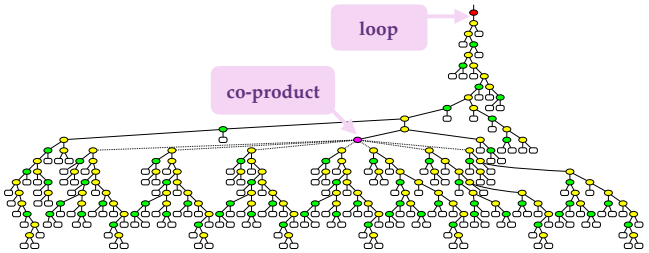
From component to system co-design: The rest of the section reuses the battery DP into a larger co-design problem that considers the co-design of actuation together with energetics for a drone (Fig. 47a). We will see that the decision boundaries



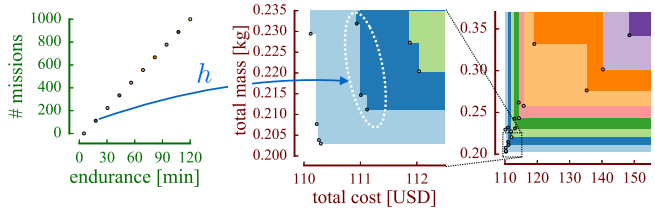
(a) Co-design diagram corresponding to (34)–(35).

```
mcdp {
  provides endurance      [s]
  provides extra_payload  [kg]
  provides extra_power    [W]
  provides num_missions   [R]
  provides velocity       [m/s]

  requires total_cost [$]
  requires total_mass [g]

  battery = instance `Batteries
  actuation = instance `Actuation

  total_power = extra_power +
    power required by actuation

  missions provided by battery >= num_missions

  energy = provided endurance * total_power
  capacity provided by battery >= energy

  total_mass = (
      mass required by battery +
      actuator_mass required by actuation +
      extra_payload)

  required total_mass >= total_mass

  gravity = 9.81 m/s^2
  weight = total_mass * gravity

  lift provided by actuation >= weight
  velocity provided by actuation >= velocity

  replacements = maintenance required by battery
  cost_per_replacement = 10 $
  labor_cost = cost_per_replacement * replacements

  required total_cost >= (
    cost required by actuation +
    cost required by battery +
    labor_cost)
}
```

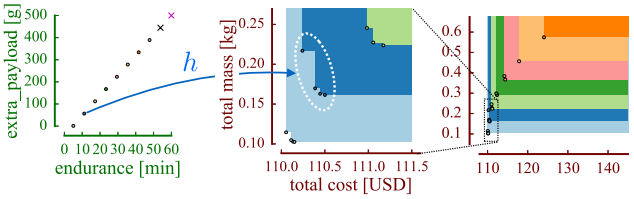
(b) MCDPL code for (34)–(35). The "instance" statements refer to previously defined models for batteries (Fig. 45b) and actuation (not shown).



(c) Tree representation for the MCDP. Yellow/green rounded ovals are series/par junctions. There is one co-product junction, signifying the choice between different battery technologies, and one loop junction, at the root of the tree.



(d) Relation between endurance and number of missions and cost and mass.



(e) Relation between endurance and payload and cost and mass.

Figure 47. In panel (c), the payload is fixed to 100 g and extra power is set to 1 W. In panel (d), the number of missions is fixed to 400 and extra power is set to 1 W. The last two values, marked with "×", are not feasible.

change dramatically, which shows that the optimal choices for a component cannot be made in isolation from the system.

The functionality of the drone's subsystem considered (Fig. 47a) are parametrized by endurance, number of missions, extra power to be supplied, and payload. We model "actuation" as a design problem with functionality lift [N] and resources cost, mass and power, and we assume that power is a quadratic function of lift (Fig. 48). Any other monotone map could be used.

Figure 48.

The co-design constraints that combine energetics and actuation are

$$\begin{aligned} \text{battery capacity} &\geq \text{total power} \times \text{endurance}, && (34)\\ \text{total power} &= \text{actuation power} + \text{extra power},\\ \text{weight} &= \text{total mass} \times \text{gravity},\\ \text{actuation lift} &\geq \text{weight},\\ \text{labor cost} &= \text{cost per replacement} \times \text{battery maintenance},\\ \text{total cost} &= \text{battery cost} + \text{actuation cost} + \text{labor cost},\\ \text{total mass} &= \text{battery mass} + \text{actuation mass} + \text{payload}. && (35) \end{aligned}$$

The co-design graph contains recursive constraints: the power for actuation depends on the total weight, which depends on the mass of the battery, which depends on the capacity to be provided, which depends on the power for actuation. The MCDPL code for this model is shown in Fig. 47b; it refers to the previously defined models for "batteries" and "actuation".

The co-design problem is now complex enough that we can appreciate the compositional properties of MCDPs to perform a qualitative analysis. Looking at Fig. 47a, we know that there is a monotone relation between any pair of functionality and resources, such as payload and cost, or endurance and mass, even without knowing exactly what are the models for battery and actuation.

When fully expanded, the co-design graph (too large to display) contains 110 nodes and 110 edges. It is possible to remove all cycles by removing only one edge (e.g., the energy $\leq$ capacity constraint), so the design complexity (Def. 24) is equal to $\mathsf{width}(\overline{\mathbb{R}}_+) = 1$. The tree representation is shown in Fig. 47c. Because the co-design diagram contains cycles, there is a loop operator at the root of the tree, which implies we need to solve a least fixed point problem. Because of the scale of the problem, it is not possible to show the map h explicitly, like we did in (28) for the previous example. The least fixed point sequence converges to 64 bits machine precision in 50-100 iterations.

To visualize the multidimensional relation

$$h\colon \overline{\mathbb{R}}_+ \times \overline{\mathbb{R}}_+^{\mathrm{s}} \times \overline{\mathbb{R}}_+^{\mathrm{W}} \times \overline{\mathbb{R}}_+^{\mathrm{g}} \to \mathsf{A}(\overline{\mathbb{R}}_+^{\mathrm{kg}} \times \overline{\mathbb{R}}_+^{\mathrm{USD}}),$$

we need to project across 2D slices. Fig. 47d shows the relation when the functionality varies in a chain in the space endurance/missions, and Fig. 47e shows the results for a chain in the space endurance/payload.

Finally, Fig. 50 shows the optimal choices of battery technologies in the endurance/missions space, when one wants to minimize mass, cost, or $\langle \text{mass}, \text{cost} \rangle$. The decision boundaries are completely different from those in Fig. 49. This shows that

it is not possible to optimize a component separately from the rest of the system, if there are cycles in the co-design diagram.

X. Discussion of related work

Theory of design: Modern engineering has long since recognized the two ideas of modularity and hierarchical decomposition, yet there exists no general quantitative theory of design that is applicable to different domains. Most of the works in the theory of design literature study abstractions that are meant to be useful for a human designer, rather than an automated system. For example, a *function structure* diagram [18, p. 32] decomposes the function of a system in subsystems that exchange energy, materials, and signals, but it is not a formal representation. From the point of view of the theory of design, the contribution of this work is that the *design problem* abstraction developed, where one takes functionality and resources as the interfaces for the subsystems, is at the same time (1) mathematically precise; (2) intuitive to understand; and (3) leads to tractable optimization problems.

This work also provides a clear answer to one long-standing issue in the theory of design: the inter-dependence between subsystems, (i.e., cycles in the co-design graph). Consider, as an example, Suh's theory of *axiomatic design* [19], in which the first "axiom" is to keep the design requirements orthogonal (i.e., do not introduce cycles). This work shows that it is possible to deal elegantly with recursive constraints.

Partial Order Programming: In "Partial Order Programming" [20] Parker studies a hierarchy of optimization problems that can be represented as a set of partial order constraints. The main interest is to study partial order constraints as the means to define the semantics of programming languages and for declarative approaches to knowledge representation.

In Parker's hierarchy, MCDPs are most related to the class of problems called *continuous monotone partial order program* (CMPOP). CMPOPs are the least specific class of problems studied by Parker for which it is possible to obtain existence results and a systematic solution procedure. MCDPs subsume CMPOPs. A CMPOP is an MCDP where: 1) All functionality and resources belong to the same poset $\mathcal{P}$ ($\mathcal{F}_v = \mathcal{R}_v = \mathcal{P}$); 2) Each functionality/resource relation is a simple map, rather than a multi-valued relation; 3) There are no dangling functionality edges in the co-design diagram ($\mathcal{F} = \mathbb{1}$).

In a MCDP, each DP is described by a Scott continuous map $h : \mathcal{F} \to \mathsf{A}\mathcal{R}$ which maps one functionality to a minimal set of resources. By contrast, in a CMPOP an operator corresponds to a Scott continuous map $h : \mathcal{F} \to \mathcal{R}$. The consequence is that a CMPOP has a unique solution [20, Theorem 8], while an MCDP can have multiple minimal solutions (or none at all).

Abstract interpretation: The methods used from order theory are the same used in the field of *abstract interpretation* [21]. In that field, the least fixed point semantics arises from problems such as computing the sets of reachable states. Given a starting state, one is interested to find a subset of states that is closed under the dynamics (i.e. a fixed point), and that is the smallest that contains the given initial state (i.e. a *least* fixed point). Reachability and other properties lead to considering systems of equation of the form

$$x_i = \varphi_i(x_1, \ldots, x_i, \ldots, x_n), \quad i = 1, \ldots, n, \tag{36}$$

where each value of the index i is for a control point of the program, and φ_i are Scott continuous functions on the abstract lattice that represents the properties of the program. In the simplest case, each x_i could represent intervals that a variable could assume at step i. By applying the iterations, one finds which properties can be inferred to be valid at each step.

We can repeat the same considerations we did for Parker's CMPOPs vs MCDPs. In particular, in MCDP we deal with multi-valued maps, and there is more than one solution.

In the field of abstract interpretation much work has been done towards optimizing the rate of convergence. The order of evaluation in (36) does not matter. Asynchronous and "chaotic" iterations were proposed early [22] and are still object of investigation [23]. To speed up convergence, the so called "widening" and "narrowing" operators are used [24]. The ideas of chaotic iteration, widening, narrowing, are not immediately applicable to MCDPs, but it is a promising research direction.

XI. Conclusions

This paper described a mathematical theory of co-design, in which the primitive objects are design problems, defined axiomatically as relations between functionality, resources, and implementation. Monotone Co-Design Problems (MCDPs) are the interconnection of design problems whose functionality and resources are complete partial orders and the relation is Scott continuous. These were shown to be non-convex, non-differentiable, and not even defined on continuous spaces. Yet, MCDPs have a systematic solution procedure in the form of a least fixed point iteration, whose complexity depends on a measure of interdependence between design problems—it is easier to design a system composed of subsystems that are only loosely coupled. Based on this theory, it is possible to create modeling languages and optimization tools that allow the user to quickly define and solve multi-objective design problems in heterogeneous domains.

References

[1] A. Censi. "A Class of Co-Design Problems with Cyclic Constraints and Their Solution". In: *Robotics and Automation Letters* (2016).

[2] A. Censi. "Monotone Co-Design Problems; or, everything is the same". In: *Proceedings of the American Control Conference (ACC)*. 2016.

[3] B. Davey and H. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002. DOI: 10.1017/cbo9780511809088.

[4] S. Roman. *Lattices and Ordered Sets*. Springer, 2008. DOI: 10.1007/978-0-387-78901-9.

[5] E. Manes and M. Arbib. *Algebraic approaches to program semantics*. Springer-Verlag, 1986. DOI: 10.1007/978-1-4612-4962-7.

[6] E. A. Lee and S. A. Seshia. *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*. 1st ed. Available for download on the authors' website http://leeseshia.org/. 2010.

[7] M. Svorenova, M. Lahijanian, A. A. Morye, D. Rao, I. Posner, P. Newman, H. Kress-Gazit, and M. Kwiatkowska. *Resource-Performance Trade-off Analysis for Mobile Robots*. 2016.

[8] M. Z. Zia, L. Nardi, A. Jack, E. Vespa, B. Bodin, P. H. J. Kelly, and A. J. Davison. "Comparative Design Space Exploration of Dense and Semi-Dense SLAM". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2016.

[9] S. M. LaValle. *Sensing and Filtering: A Fresh Perspective Based on Preimages and Information Spaces*. Foundations and Trends in Robotics Series. Now Publishers, 2012. DOI: 10.1561/2300000004.

[10] J. M. O'Kane and S. M. LaValle. "On comparing the power of robots". In: *International Journal of Robotics Research* 27.1 (2008).

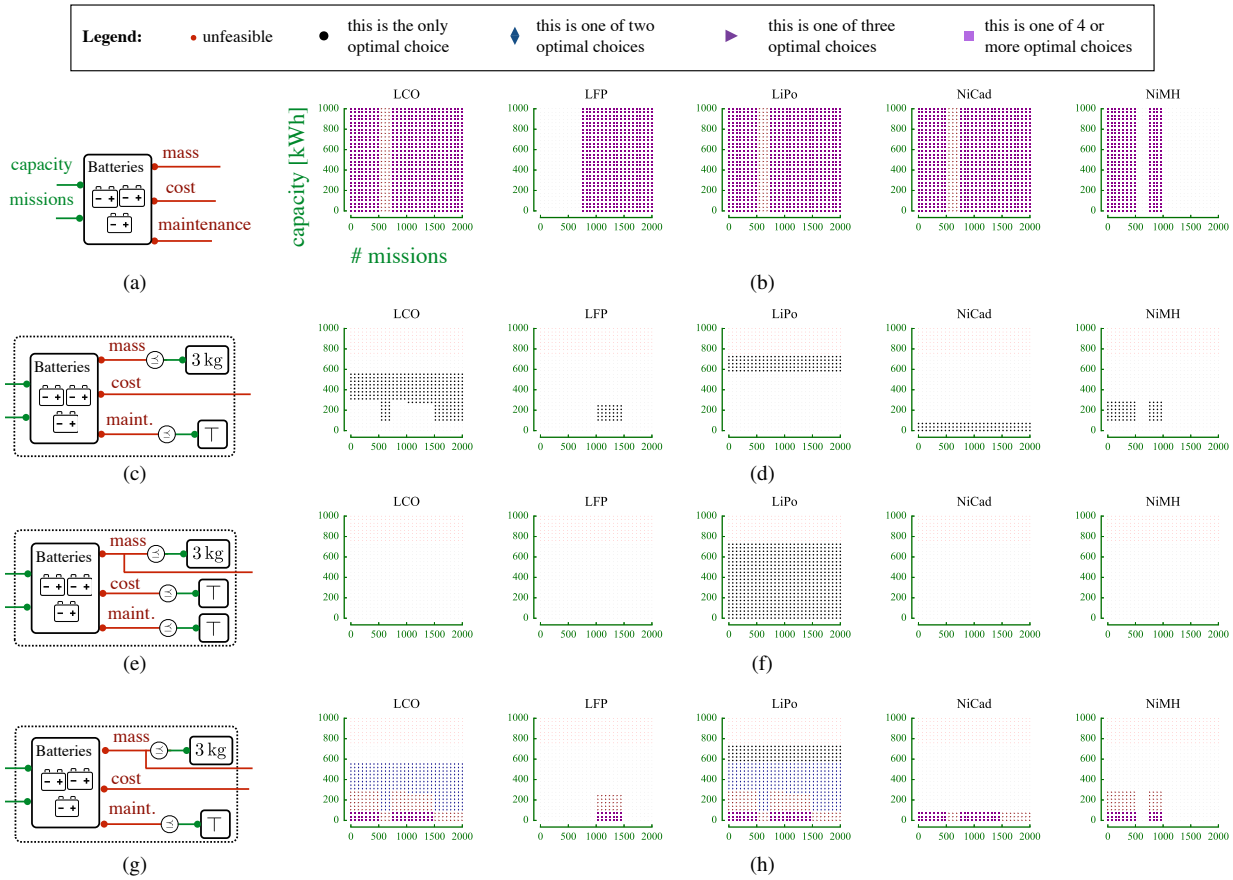


Figure 49. This figure shows the optimal decision boundaries for the different battery technologies for the design problem "batteries", defined as the co-product of all battery technologies (Fig. 45). Each row shows a different variation of the problem. The first row (panels *a–b*) shows the case where the objective function is the product of ⟨mass, cost, maintenance⟩. The shape of the symbols shows how many minimal solutions exists for a particular value of the functionality ⟨capacity, missions⟩. In this case, there are always three or more minimal solutions. The second row (panels *c–d*) shows the decision boundaries when minimizing only the scalar objective cost, with a hard constraint on mass. The hard constraints makes some combinations of the functionality infeasible. Note how the decision boundaries are nonconvex, and how the formalisms allows to define slight variations of the problem.

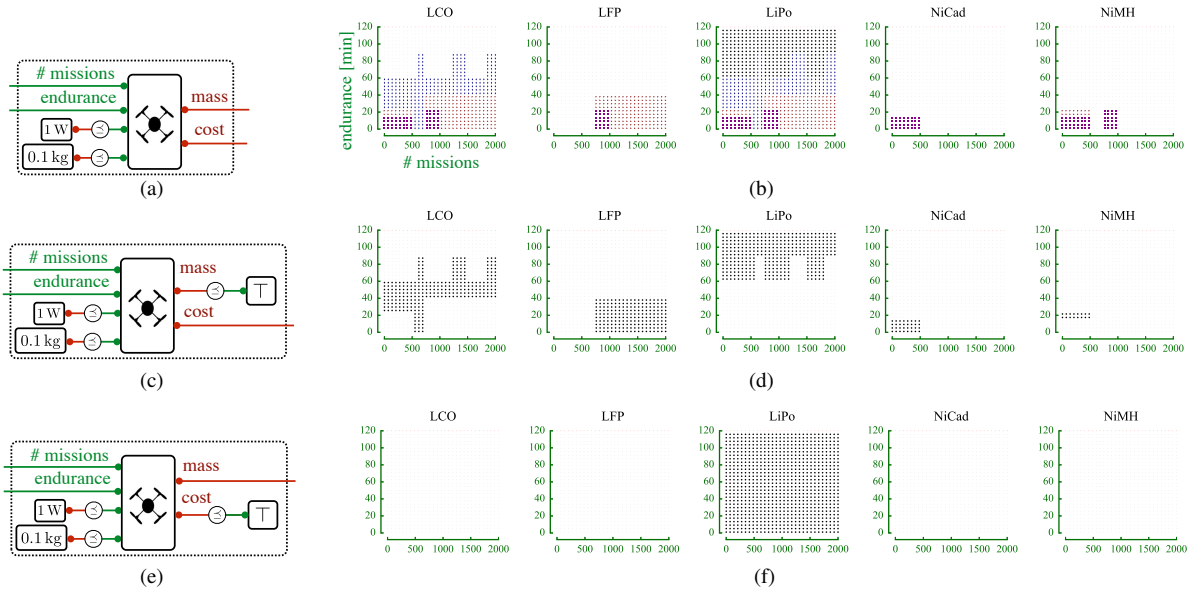


Figure 50. This figure shows the decision boundaries for the different values of battery technologies for the integrated actuation-energetics model described in Fig. 47. Please see the caption of Fig. 49 for an explanation of the symbols. Notice how in most cases the decision boundaries are different than those in Fig. 49: this is an example in which one component cannot be optimized by itself without taking into account the rest of the system.

[11] M. Egerstedt. “Motion Description Languages for Multi-Modal Control in Robotics”. In: *Control Problems in Robotics*. Springer Science + Business Media, 2003. DOI: 10.1007/3-540-36224-x_5.

[12] S. Soatto. “Steps Towards a Theory of Visual Information: Active Perception, Signal-to-Symbol Conversion and the Interplay Between Sensing and Control”. In: *CoRR* abs/1110.2053 (2011). URL: http://arxiv.org/abs/1110.2053.

[13] D. I. Spivak. *Category Theory for the Sciences*. MIT, 2014.

[14] P. A. Golovach, P. Heggernes, D. Kratsch, and Y. Villanger. “An Incremental Polynomial Time Algorithm to Enumerate All Minimal Edge Dominating Sets”. In: *Algorithmica* 72.3 (June 2015).

[15] G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott. *Continuous Lattices and Domains*. Cambridge University Press, 2003. DOI: 10.1017/cbo9780511542725.

[16] A. Censi. “Handling Uncertainty in Monotone Co-Design Problems”. In: *cs.RO* abs/1609.03103 (2016). URL: https://arxiv.org/abs/1609.03103.

[17] M. Grant and S. Boyd. “Graph implementations for nonsmooth convex programs”. In: *Recent Advances in Learning and Control*. Ed. by V. Blondel, S. Boyd, and H. Kimura. Lecture Notes in Control and Information Sciences. http://stanford.edu/~boyd/graph_dcp.html. Springer-Verlag Limited, 2008.

[18] G. Pahl, W. Beitz, J. Feldhusen, and K.-H. Grote. *Engineering Design: A Systematic Approach*. 3rd. Springer, Jan. 2007.

[19] N. Suh. *Axiomatic Design: Advances and Applications*. Oxford University Press, 2001.

[20] D. S. Parker Jr. “Partial Order Programming”. In: *Principles of Programming Languages* (1989). Also see the extended technical report (CSD-870067) available on the author’s webpage at the url http://web.cs.ucla.edu/~stott/pop/.

[21] P. Cousot and R. Cousot. “Abstract Interpretation: Past, Present and Future”. In: *Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2014. DOI: 10.1145/2603088.2603165.

[22] P. Cousot. *Asynchronous iterative methods for solving a fixed point system of monotone equations in a complete lattice*. Res. rep. R.R. 88. Université scientifique et médicale de Grenoble, 1977.

[23] F. Bourdoncle. “Efficient chaotic iteration strategies with widenings”. In: *Formal Methods in Programming and Their Applications*. Springer Science + Business Media, 2005. DOI: 10.1007/bfb0039704.

[24] A. Cortesi and M. Zanioli. “Widening and narrowing operators for abstract interpretation”. In: *Computer Languages, Systems & Structures* 37.1 (2011). DOI: 10.1016/j.cl.2010.09.001.