Handling Uncertainty in Monotone Co-Design Problems

Andrea Censi

***Abstract*— The work presented here contributes to a compositional theory of "co-design" that allows to optimally design a robotic platform. In this framework, a user models each subsystem as a monotone relation between functionality provided and resources required. These models can be easily composed to express the co-design constraints between different subsystems. The user then queries the model, to obtain the design with minimal resources usage, subject to a lower bound on the provided functionality. This paper concerns the introduction of uncertainty in the framework. Uncertainty has two roles: first, it allows to deal with limited knowledge in the models; second, it also can be used to generate consistent relaxations of a problem, as the computation requirements can be lowered should the user accept some uncertainty in the answer.**

I. Introduction

The design of a robotic platform involves the choice and configuration of many hardware and software subsystems (actuation, energetics, perception, control, ...) in an harmonious entity in which all *co-design constraints* are respected. Because robotics is a relatively young discipline, there is still little work towards obtaining systematic procedures to derive optimal designs. Therefore, robot design is a lengthly design process mainly based on empirical evaluation and trial and error. The work presented here contributes to a theory of co-design that allows to optimally design a robotic platform based on formal models of the performance of its subsystems. The goal is to allow a designer to create better designs, faster. This paper describes the introduction of uncertainty in the theory.

Previous work: In previous work [1–3], I have proposed a compositional theory for co-design. The user defines "design problems" (DPs) that describe the constraints for each subsystem. These DPs can then be hierarchically composed and interconnected to obtain the class of Monotone Co-Design Problems (MCDPs).

An example of MCDP is sketched in Fig. 1. The design problem consists in finding an optimal configuration of a UAV, optimizing over actuators, sensors, processors, and batteries. Each design problem (DP) is formalized as a relation between functionality and resources. For example, the functionality of the UAV is parameterized by three numbers: the distance to travel for each mission; the payload to transport; the number of missions to fly. The optimal design is defined as the one that satisfies the functionality constraints while using the minimal amount of resources (cost and mass).

The convenience of the MCDP framework is that the user can define design problems for each subsystem and then compose them. The definition of the DPs is specified

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

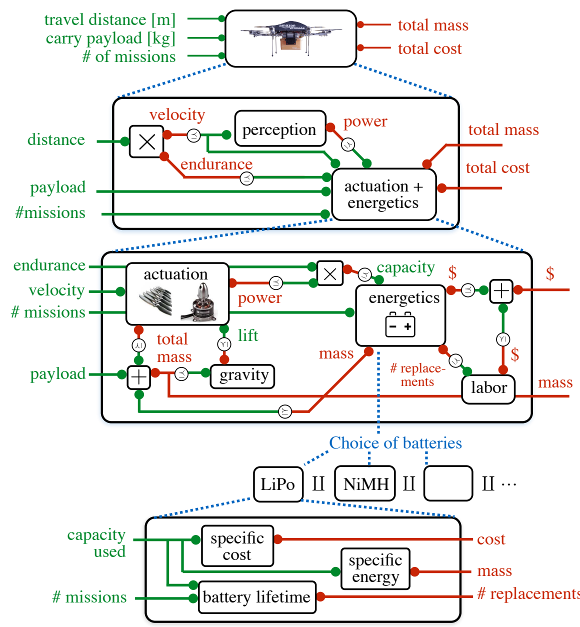


Fig. 1. Monotone Co-Design Problems (MCDPs) can capture much of the complexity of the optimal robot design process. The user defines a co-design diagram by hierarchical composition and arbitrary interconnection of primitive "design problems", modeled as monotone relations between functionality and resources. The semantics of the MCDP of the figure is the minimization of the total mass and cost of the platform, subject to functionality constraints (distance, payload, number of missions). This paper describes how to introduce uncertainty in this framework, which allows, for example, to introduce parametric uncertainty in the definition of components properties (e.g. specific cost of batteries).

using a domain-specific language that promotes composition and code reuse; the formal specification is contained in the supplementary materials. In the figure, the model is exploded to show how actuation and energetics are modeled. Perception is modeled as a relation between the velocity of the platform and the power required. Actuation is modeled as a relation between lift and power/cost. Batteries are described by a relation between capacity and mass/cost. The interconnection between these describe the "co-design constraints": e.g., actuators must lift the batteries, the batteries must power the actuators. In this example, there are different battery technologies (LiPo, etc.), each specified by specific energy, specific cost, and lifetime, thus characterized by a different relation between capacity, number of missions and mass and cost.

Once the model is defined, it can be queried to obtain the *minimal* solution in terms of resources — here, total cost and total mass. The output to the user is the Pareto front containing all non-dominated solutions. The corresponding optimization problem is, in general, nonconvex. Yet, with few assumptions, it is possible to obtain a systematic solution procedure, and show that there exists a dynamical system whose fixed point corresponding to the set of minimal solutions.

Contribution: This paper describes how to add a notion of *uncertainty* in the MCDP framework. The model of uncertainty considered is interval uncertainty on arbitrary partial orders. For a poset $\langle \mathcal{P}, \preceq \rangle$, these are sets of the type $\{x \in \mathcal{P} \colon a \preceq x \preceq b\}$. I will show how one can introduce this type of uncertainty in the MCDP framework by considering ordered pairs of design problems. Each pair describes lower and upper bounds for resources usage. These *uncertain design problems* (UDPs) can be composed using series, parallel, and feedback interconnection, just like their non-uncertain counterparts.

The user is then presented with *two* Pareto fronts, corresponding to a lower bound and an upper bound for resource consumption, in the best case and in the worst case, respectively.

This is different from the usual formalization of "robust optimization" (see e.g., [4, 5]), usually formulated as a "worst case" analysis, in which one the uncertainty in the problem is described by a set of possible parameters, and the optimization problem is posed as finding the one design that is valid for all cases.

Uncertainty plays two roles: it can be used as a *modeling tool*, where the relations are uncertain because of our limited knowledge, and it can be used as a *computational tool*, in which we deliberately choose to consider uncertain relations as a relaxation of the problem, to reduce the computational load, while maintaining precise consistency guarantees. With these additions, the MCDP framework allows to describe even richer design problems and to efficiently solve them.

Paper organization: Section II and III summarize previous work. They give a formal definition of design problems (DPs) and their composition, called Monotone Co-Design Problems (MCDPs). Section IV through VI describe the notion of Uncertain Design Problem (UDP), the semantics of their interconnection, and the general theoretical results. Section VII describes three specific applications of the theory with numerical results. The supplementary materials include detailed models written in MCDPL and pointers to obtain the source code and a virtual machine for reproducing the experiments.

II. Design Problems

A design problem (DP) is a monotone relation between *provided functionality* and *required resources*. Functionality and resources are complete partial orders (CPO) [6], indicated by $\langle \mathcal{F}, \preceq_{\mathcal{F}} \rangle$ and $\langle \mathcal{R}, \preceq_{\mathcal{R}} \rangle$. The graphical representations uses nodes for DPs and green (red) edges for functionality and resources (Fig. 2).

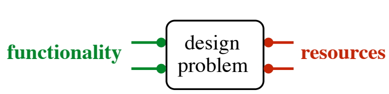


Fig. 2.

Example 1. The first-order characterization of a battery is as a store of energy, in which the capacity [kWh] is the functionality (what the battery provides) and mass [kg] and cost [$] are resources (what the battery requires) (Fig. 3).

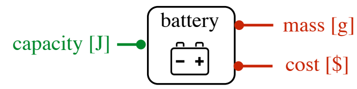


Fig. 3.

In general, fixed a functionality $f \in \mathcal{F}$, there will be multiple resources in $\mathcal{R}$ sufficient to perform the functionality that are incomparable with respect to $\preceq_{\mathcal{R}}$. For example, in the case of a battery one might consider different battery technologies that are incomparable in the mass/cost resource space (Fig. 4).

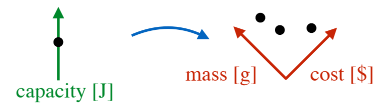


Fig. 4.

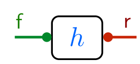
A subset with "minimal", "incomparable" elements is called "antichain". This is the mathematical formalization of what is informally called a "Pareto front".

Definition 2. An *antichain* S in a poset $\langle \mathcal{P}, \preceq \rangle$ is a subset of $\mathcal{P}$ such that no element of S dominates another element: if $x, y \in S$ and $x \preceq y$, then $x = y$.

Lemma 3. *Let $\mathsf{A}\mathcal{P}$ be the set of antichains of $\mathcal{P}$. $\mathsf{A}\mathcal{P}$ is a poset itself, with the partial order $\preceq_{\mathsf{A}\mathcal{P}}$ defined as*

$$S_1 \preceq_{\mathsf{A}\mathcal{P}} S_2 \equiv \uparrow S_1 \supseteq \uparrow S_2. \tag{1}$$

Definition 4. A *monotone design problem* (DP) is a tuple $\langle \mathcal{F}, \mathcal{R}, h \rangle$ such that $\mathcal{F}$ and $\mathcal{R}$ are CPOs, and $h : f \to \mathsf{A}\mathcal{R}$ is a monotone and Scott-continuous function ([7] or [3, Definition 11]).



$\equiv \quad r \in h(f)$

Fig. 5.

Each functionality f corresponds to an antichain of resources $h(f) \in \mathsf{A}\mathcal{R}$ (Fig. 6).

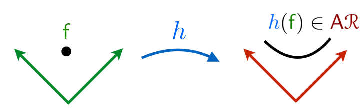


Fig. 6.

Monotonicity implies that, if the functionality is increased, then the required resources increase as well (Fig. 7).

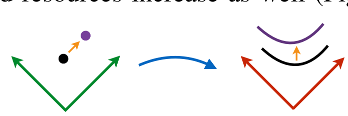
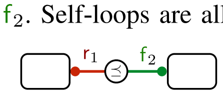


Fig. 7.

III. Monotone Co-Design Problems

A Monotone Co-Design Problem is a multigraph of DPs. Two DPs can be connected by adding an edge (Fig. 8). The semantics of the interconnection is that the resources required by the first DP must be provided by the second DP. Mathematically, this is a partial order inequality constraint of the type $r_1 \preceq f_2$. Self-loops are allowed as well.



$\equiv \quad r_1 \preceq f_2$

Fig. 8.

Example 5. The MCDP in Fig. 9 is the interconnection of 3 DPs h_a, h_b, h_c. The semantics of the MCDP as an optimization problem is shown in Fig. 10.

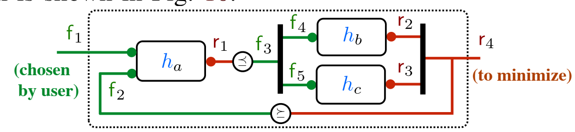
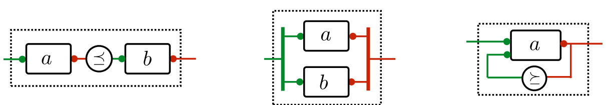
Fig. 9. 

Fig. 10.
$$\mathsf{f}_1 \mapsto \left\{ \begin{array}{llll} \operatorname{Min} \mathsf{r}_4 & \mathsf{r}_1 \in h_a(\mathsf{f}_1, \mathsf{f}_2) & \mathsf{r}_1 \preceq \mathsf{f}_3 & \mathsf{f}_3 = \langle \mathsf{f}_4, \mathsf{f}_5 \rangle \\ & \mathsf{r}_2 \in h_b(\mathsf{f}_4) & \mathsf{r}_4 \preceq \mathsf{f}_2 & \mathsf{r}_4 = \langle \mathsf{r}_2, \mathsf{r}_3 \rangle \\ & \mathsf{r}_3 \in h_c(\mathsf{f}_5) & & \end{array} \right.$$

To describe the interconnection, the obvious choice is to describe it as a graph, as a set of nodes and of edges. For our goals, it is more convenient to use an algebraic definition. In the algebraic definition, the graph is a represented by a tree, where the leaves are the nodes, and the junctions are one of three operators (series, par, loop), as in Fig. 11.

Similar constructions are widespread in computer science. One can see this in the spirit of series-parallel graphs (see, e.g., [8]), with an additional feedback operator to be able to represent all graphs. Equivalently, we are defining a symmetric traced monoidal category (see, e.g., [9] or [10] for an introduction); note that the loop operator is related to the "trace" operator but not exactly equivalent, though they can be defined in terms of each other. An equivalent construction for network processes is given in Stefanescu [11].



(a) $\mathsf{series}(a, b)$ (b) $\mathsf{par}(a, b)$ (c) $\mathsf{loop}(a)$

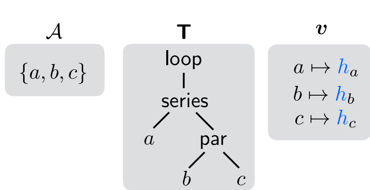
Fig. 11. The three operators used in the inductive definition of MCDPs.

Let us use a standard definition of "operators", "terms", and "atoms" (see, e.g., [12, p.41]). Given a set of operators ops and a set of atoms $\mathcal{A}$, let $\mathsf{Terms}(\mathsf{ops}, \mathcal{A})$ be the set of all inductively defined expressions. For example, if the operator set contains only an operator f of *arity* 1, and there is only one atom a, then the terms are $\mathsf{Terms}(\{f\}, \{a\}) = \{a, f(a), f(f(a)), \dots\}$.

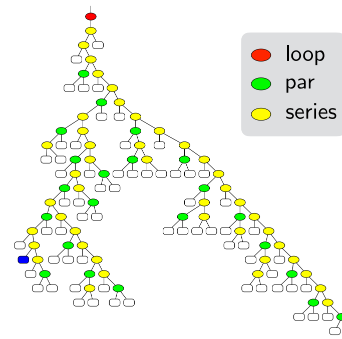
Definition 6 (Algebraic definition Monotone Co-Design Problems)**.** An MCDP is a tuple $\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v} \rangle$, where:

1) $\mathcal{A}$ is any set of atoms, to be used as labels.
2) The term $\mathbf{T}$ in the $\{\mathsf{series}, \mathsf{par}, \mathsf{loop}\}$ algebra describes the structure of the graph:
$$\mathbf{T} \in \mathsf{Terms}(\{\mathsf{series}, \mathsf{par}, \mathsf{loop}\}, \mathcal{A}).$$
3) The *valuation* $\boldsymbol{v}$ is a map $\boldsymbol{v} : \mathcal{A} \to \mathsf{DP}$ that assigns a DP to each atom.

Example 7. The MCDP in Fig. 9 can be described by the atoms $\mathcal{A} = \{a, b, c\}$, the term $\mathbf{T} = \mathsf{loop}(\mathsf{series}(a, \mathsf{par}(b, c)))$, plus the valuation $\boldsymbol{v} : \{a \mapsto h_a, b \mapsto h_b, c \mapsto h_c\}$. The tuple $\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v} \rangle$ for this example is shown in Fig. 12.

Fig. 12. 

Example 8. A sketch of the algebraic representation for part of the example in Fig. 1 is shown in Fig. 13. The supplementary materials contain more detailed visualizations of the trees for the numerical examples, which take too much space for including in this paper.

Fig. 13. 

A. Semantics of MCDPs

We can now define the *semantics* of an MCDP. The *semantics* is a function φ that, given an algebraic definition of an MCDP, returns a DP. Thanks to the algebraic definition, to define φ, we need to only define what happens in the base case (equation 2), and what happens for each operator series, par, loop (equations 3–5).

Definition 9 (Semantics of MCDP)**.** Given an MCDP in algebraic form $\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v} \rangle$, the semantics
$$\varphi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v} \rangle]\!] \in \mathsf{DP}$$
is defined as follows:
$$\varphi[\![\langle \mathcal{A}, a, \boldsymbol{v} \rangle]\!] \doteq \boldsymbol{v}(a), \qquad \text{for all } a \in \mathcal{A}, \tag{2}$$
$$\varphi[\![\langle \mathcal{A}, \mathsf{series}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v} \rangle]\!] \doteq \varphi[\![\langle \mathcal{A}, \mathbf{T}_1, \boldsymbol{v} \rangle]\!] \circledcirc \varphi[\![\langle \mathcal{A}, \mathbf{T}_2, \boldsymbol{v} \rangle]\!], \tag{3}$$
$$\varphi[\![\langle \mathcal{A}, \mathsf{par}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v} \rangle]\!] \doteq \varphi[\![\langle \mathcal{A}, \mathbf{T}_1, \boldsymbol{v} \rangle]\!] \otimes \varphi[\![\langle \mathcal{A}, \mathbf{T}_2, \boldsymbol{v} \rangle]\!], \tag{4}$$
$$\varphi[\![\langle \mathcal{A}, \mathsf{loop}(\mathbf{T}), \boldsymbol{v} \rangle]\!] \doteq \varphi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v} \rangle]\!]^{\dagger}. \tag{5}$$

The operators $\circledcirc, \otimes, \dagger$ are defined in Def. 11–Def. 12. Please see [3, Section VI] for details about the interpretation of these operators and how they are derived.

The $\otimes$ operator is a regular product in category theory: we are considering all possible combinations of resources required by h_1 and h_2.

Definition 10 (Product operator $\otimes$)**.** For two maps $h_1 : \mathcal{F}_1 \to \mathsf{A}\mathcal{R}_1$ and $h_2 : \mathcal{F}_2 \to \mathsf{A}\mathcal{R}_2$, define
$$\begin{aligned} h_1 \otimes h_2 : (\mathcal{F}_1 \times \mathcal{F}_2) &\to \mathsf{A}(\mathcal{R}_1 \times \mathcal{R}_2), \\ \langle \mathsf{f}_1, \mathsf{f}_2 \rangle &\mapsto h_1(\mathsf{f}_1) \times h_2(\mathsf{f}_2), \end{aligned}$$
where $\times$ is the product of two antichains.

The $\circledcirc$ operator is similar to a convolution: fixed f_1, one evaluates the resources $\mathsf{r}_1 \in h_1(\mathsf{f})$, and for each r_1, $h_2(\mathsf{r}_1)$ is evaluated. The Min operator then chooses the minimal elements.

Definition 11 (Series operator $\circledcirc$)**.** For two maps $h_1 : \mathcal{F}_1 \to \mathsf{A}\mathcal{R}_1$ and $h_2 : \mathcal{F}_2 \to \mathsf{A}\mathcal{R}_2$, if $\mathcal{R}_1 = \mathcal{F}_2$, define

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

The dagger operator $\dagger$ is actually a standard operator used in domain theory (see, e.g., [7, pp. II–2.29]).

Definition 12 (Loop operator $\dagger$)**.** For a map $h : \mathcal{F}_1 \times \mathcal{F}_2 \to \mathsf{A}\mathcal{R}$, define

$$\begin{aligned} h^\dagger &: \mathcal{F}_1 \to \mathsf{A}\mathcal{R}, \\ \mathsf{f}_1 &\mapsto \mathsf{lfp}\left(\Psi^h_{\mathsf{f}_1}\right), \end{aligned} \tag{6}$$

where lfp is the least-fixed point operator, and $\Psi^h_{\mathsf{f}_1}$ is defined as

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

B. Solution of MCDPs

Def. 9 gives a way to evaluate the map h for the graph, given the maps $\{h_a \mid a \in \mathcal{A}\}$ for the leaves. Following those instructions, we can compute $h(\mathsf{f})$, and thus find the minimal resources needed for the entire MCDP.

Example 13. The MCDP in Fig. 9 is so small that we can do this explicitly. From Def. 9, we can compute the semantics as follows:

$$\begin{aligned} h &= \varphi [\![\langle \mathcal{A}, \mathsf{loop}(\mathsf{series}(a, \mathsf{par}(b, c)), \boldsymbol{v} \rangle]\!] \\ &= (h_a \circledcirc (h_b \otimes h_c))^\dagger . \end{aligned}$$

Substituting the definitions 10–12 above, one finds that $h(\mathsf{f}) = \mathsf{lfp}\left(\Psi_{\mathsf{f}}\right)$, with

$$\begin{aligned} \Psi_{\mathsf{f}} &: \mathsf{A}\mathcal{R} \to \mathsf{A}\mathcal{R}, \\ R &\mapsto \bigcup_{\mathsf{r} \in R} \left[\operatorname*{Min}_{\preceq} \uparrow \bigcup_{s \in h_a(\mathsf{f}_1, \mathsf{r})} h_b(s) \times h_c(s) \right] \cap \uparrow \mathsf{r}. \end{aligned}$$

The least fixed point equation can be solved using Kleene's algorithm [6, CPO Fixpoint theorem I, 8.15]. A dynamical system that computes the set of solutions is given by

$$\begin{cases} R_0 & \leftarrow \{\bot_{\mathcal{R}}\}, \\ R_{k+1} & \leftarrow \Psi_{\mathsf{f}}(R_k). \end{cases}$$

The limit $\sup R_k$ is the set of minimal solutions, which might be an empty set if the problem is unfeasible.

This dynamical system is a proper algorithm only if each step can be performed with bounded computation. An example in which this is not the case are relations that give an infinite number of solutions for each functionality. For example, the very first DP appearing in Fig. 1 corresponds to the relation travel distance $\leq$ velocity $\times$ endurance, for which there are infinite numbers of pairs $\langle$velocity, endurance$\rangle$ for each value of travel distance. The machinery developed in this paper will make it possible to deal with these infinite-cardinality relations by relaxation.

IV. Uncertain Design Problems

We now consider the introduction of uncertainty. This section describes objects called Uncertain DPs (UDPs), which are an ordered pair of DPs. Each pair can be interpreted as upper and lower bounds for resource consumptions (Fig. 14).

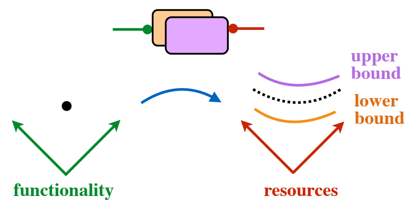


Fig. 14.

We will be able to propagate this interval uncertainty through an arbitrary interconnection of DPs. The result presented to the user will be a *pair* of antichains — a lower and an upper bound for the resource consumption.

A. Partial order $\preceq_{\mathsf{DP}}$

Being able to provide both upper and lower bounds comes from the fact that in this framework everything is ordered – there are a poset of resources, lifted to posets of antichains, which is lifted to posets of DPs, and finally, to the poset of uncertain DPs.

The first step is defining a partial order $\preceq_{\mathsf{DP}}$ on DP.

Definition 14 (Partial order $\preceq_{\mathsf{DP}}$)**.** Consider two DPs $h_1, h_2 : \mathcal{F} \to \mathsf{A}\mathcal{R}$. The DP h_1 precedes h_2 if it requires fewer resources for all functionality f:

$$h_1 \preceq_{\mathsf{DP}} h_2 \quad \equiv \quad h_1(\mathsf{f}) \preceq_{\mathsf{A}\mathcal{R}} h_2(\mathsf{f}), \text{ for all } \mathsf{f} \in \mathcal{F}.$$

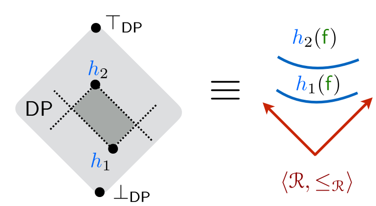


Fig. 15.

In this partial order, there is both a top $\top_{\mathsf{DP}}$ and a bottom $\bot_{\mathsf{DP}}$, defined as follows:

$$\begin{aligned} \bot_{\mathsf{DP}} &: \mathcal{F} \to \mathsf{A}\mathcal{R}, & \top_{\mathsf{DP}} &: \mathcal{F} \to \mathsf{A}\mathcal{R}, \\ \mathsf{f} &\mapsto \{\bot_{\mathcal{R}}\}. & \mathsf{f} &\mapsto \emptyset. \end{aligned} \tag{7}$$

$\bot_{\mathsf{DP}}$ means that any functionality can be done with zero resources, and $\top_{\mathsf{DP}}$ means that the problem is always infeasible ("the set of feasible resources is empty").

B. Uncertain DPs (UDPs)

Definition 15 (Uncertain DPs)**.** An Uncertain DP (UDP) $\boldsymbol{u}$ is a pair of DPs $\langle \mathsf{L}\boldsymbol{u}, \mathsf{U}\boldsymbol{u} \rangle$ such that $\mathsf{L}\boldsymbol{u} \preceq_{\mathsf{DP}} \mathsf{U}\boldsymbol{u}$.

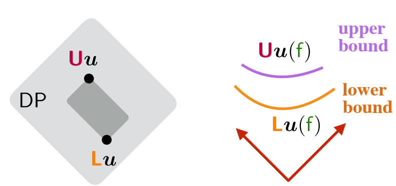


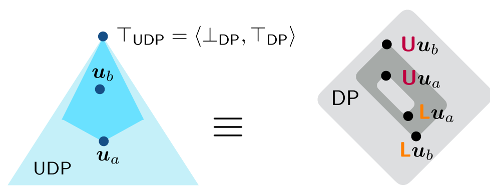
Fig. 16.

C. Order on UDP

Definition 16 (Partial order $\preceq_{\mathsf{UDP}}$)**.** A UDP $\boldsymbol{u}_a$ precedes another UDP $\boldsymbol{u}_b$ if the interval $[\mathbf{L}\boldsymbol{u}_a, \mathbf{U}\boldsymbol{u}_a]$ is contained in the interval $[\mathbf{L}\boldsymbol{u}_a, \mathbf{U}\boldsymbol{u}_a]$ (Fig. 17):

$$\boldsymbol{u}_a \preceq_{\mathsf{UDP}} \boldsymbol{u}_b \quad \equiv \quad \mathbf{L}\boldsymbol{u}_b \preceq_{\mathsf{DP}} \mathbf{L}\boldsymbol{u}_a \preceq_{\mathsf{DP}} \mathbf{U}\boldsymbol{u}_a \preceq_{\mathsf{DP}} \mathbf{U}\boldsymbol{u}_b.$$

Fig. 17.



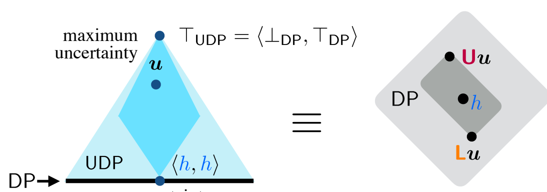
The partial order $\preceq_{\mathsf{UDP}}$ has a top $\top_{\mathsf{UDP}} = \langle \bot_{\mathsf{DP}}, \top_{\mathsf{DP}} \rangle$. This pair describes maximum uncertainty about the DP: we do not know if the DP is feasible with 0 resources ($\bot_{\mathsf{DP}}$), or if it is completely infeasible ($\top_{\mathsf{DP}}$).

D. DPs as degenerate UDPs

A DP h is equivalent to a degenerate UDP $\langle h, h \rangle$.

A UDP $\boldsymbol{u}$ is a bound for a DP h if $\boldsymbol{u} \preceq_{\mathsf{UDP}} \langle h, h \rangle$, or, equivalently, if $\mathbf{L}\boldsymbol{u} \preceq_{\mathsf{UDP}} h \preceq_{\mathsf{UDP}} \mathbf{U}\boldsymbol{u}$.

Fig. 18.



A pair $\langle h, h \rangle$ is a minimal element of UDP, because it cannot be dominated by any other. Thus, we can imagine the space UDP as a pyramid (Fig. 18), with the space DP forming the base. The base represents non-uncertain DPs. The top of the pyramid is $\top_{\mathsf{UDP}}$, which represents maximum uncertainty.

V. Interconnection of Uncertain Design Problems

We now define the interconnection of UDPs, in an equivalent way to the definition of MCDPs. The only difference between Def. 6 and Def. 17 below is that the valuation assigns to each atom an UDP, rather than a DP.

Definition 17 (Algebraic definition of UMCDPs)**.** An Uncertain MCDP (UMCDP) is a tuple $\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v} \rangle$, where $\mathcal{A}$ is a set of atoms, $\mathbf{T} \in \mathsf{Terms}(\{\mathsf{series}, \mathsf{par}, \mathsf{loop}\}, \mathcal{A})$ is the algebraic representation of the graph, and $\boldsymbol{v} : \mathcal{A} \to \mathsf{UDP}$ is a valuation that assigns to each atom a UDP.

Next, the semantics of a UMCDP is defined as a map Φ that computes the UDP. Def. 18 below is analogous to Def. 9.

Definition 18 (Semantics of UMCDPs)**.** Given an UMCDP $\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v} \rangle$, the semantics function Φ computes a UDP

$$\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v} \rangle]\!] \in \mathsf{UDP},$$

and it is recursively defined as follows:

$$\Phi[\![\langle \mathcal{A}, a, \boldsymbol{v} \rangle]\!] = \boldsymbol{v}(a), \qquad \text{for all } a \in \mathcal{A}.$$

$$\mathbf{L}\Phi[\![\langle \mathcal{A}, \mathsf{series}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v} \rangle]\!] = (\mathbf{L}\Phi[\![\langle \mathcal{A}, \mathbf{T}_1, \boldsymbol{v} \rangle]\!]) \circledcirc (\mathbf{L}\Phi[\![\langle \mathcal{A}, \mathbf{T}_2, \boldsymbol{v} \rangle]\!]),$$
$$\mathbf{U}\Phi[\![\langle \mathcal{A}, \mathsf{series}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v} \rangle]\!] = (\mathbf{U}\Phi[\![\langle \mathcal{A}, \mathbf{T}_1, \boldsymbol{v} \rangle]\!]) \circledcirc (\mathbf{U}\Phi[\![\langle \mathcal{A}, \mathbf{T}_2, \boldsymbol{v} \rangle]\!]),$$

$$\mathbf{L}\Phi[\![\langle \mathcal{A}, \mathsf{par}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v} \rangle]\!] = (\mathbf{L}\Phi[\![\langle \mathcal{A}, \mathbf{T}_1, \boldsymbol{v} \rangle]\!]) \otimes (\mathbf{L}\Phi[\![\langle \mathcal{A}, \mathbf{T}_2, \boldsymbol{v} \rangle]\!]),$$
$$\mathbf{U}\Phi[\![\langle \mathcal{A}, \mathsf{par}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v} \rangle]\!] = (\mathbf{U}\Phi[\![\langle \mathcal{A}, \mathbf{T}_1, \boldsymbol{v} \rangle]\!]) \otimes (\mathbf{U}\Phi[\![\langle \mathcal{A}, \mathbf{T}_2, \boldsymbol{v} \rangle]\!]),$$

$$\mathbf{L}\Phi[\![\langle \mathcal{A}, \mathsf{loop}(\mathbf{T}), \boldsymbol{v} \rangle]\!] = (\mathbf{L}\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v} \rangle]\!])^{\dagger},$$
$$\mathbf{U}\Phi[\![\langle \mathcal{A}, \mathsf{loop}(\mathbf{T}), \boldsymbol{v} \rangle]\!] = (\mathbf{U}\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v} \rangle]\!])^{\dagger}.$$

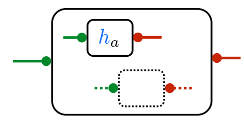
The operators $\dagger, \circledcirc, \otimes$ are defined in Def. 11–Def. 12.

VI. Approximation Results

The main result of this section is a relaxation result stated as Theorem 20 below.

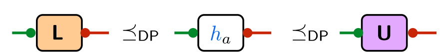
Informal statement: Suppose that we have an MCDP composed of many DPs, and one of those is h_a (Fig. 19).

Fig. 19.



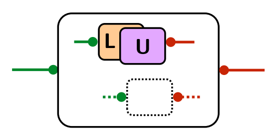
Suppose that we can find two DPs $\mathbf{L}$, $\mathbf{U}$ that bound the DP h_a (Fig. 20).

Fig. 20.



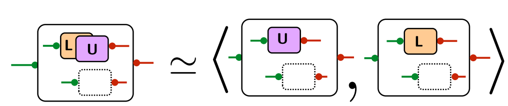
This can model either (a) uncertainty in our knowledge of h_a, or (b) a relaxation that we willingly introduce.
Then we can consider the pair $\mathbf{L}$, $\mathbf{U}$ as a UDP $\langle \mathbf{L}, \mathbf{U} \rangle$ and we can plug it in the original MCDP in place of h_a (Fig. 21).

Fig. 21.



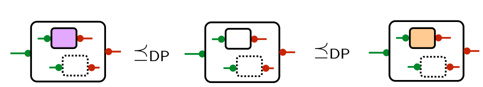
Given the semantics of interconnections of UDPs (Def. 18), this is equivalent to considering a pair of MCDPs, in which we choose either the lower bound or the upper bound (Fig. 22).

Fig. 22.



We can then show that the solution of the original MCDP is bounded below and above by the solution of the new pair of MCDPs (Fig. 23).

Fig. 23.



This result generalizes for any number of substitutions.

Formal statement: First, we define a partial order on the valuations. A valuation precedes another if it gives more information on each DP.

Definition 19 (Partial order $\preceq_V$ on valuations)**.** For two valuations $\boldsymbol{v}_1, \boldsymbol{v}_2 : \mathcal{A} \to \mathsf{UDP}$, say that $\boldsymbol{v}_1 \preceq_V \boldsymbol{v}_2$ if $\boldsymbol{v}_1(a) \preceq_{\mathsf{UDP}} \boldsymbol{v}_2(a)$ for all $a \in \mathcal{A}$.

At this point, we have enough machinery in place that we can simply state the result as "the semantics is monotone in the valuation".

Theorem 20 (Φ is monotone in the valuation)**.** *If $\boldsymbol{v}_1 \preceq_V \boldsymbol{v}_2$, then*

$$\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_1 \rangle]\!] \preceq_{\mathsf{UDP}} \Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_2 \rangle]\!].$$

The proof is given in Appendix A.4 in the supplementary materials.

This result says that we can swap any DP in a MCDP with a UDP relaxation, obtain a UMCDP, which we can solve to obtain inner and outer approximations to the solution of the original MCDP. This shows that considering uncertainty in the MCDP framework is easy; as the problem reduces to solving a pair of problems instead of one. This

The rest of the paper consists of applications of this result.

VII. Applications

This section shows three example applications of the theory:

1) The first example deals with *parametric uncertainty*.
2) The second example deals with the idea of relaxation of a scalar relation. This is equivalent to accepting a tolerance for a given variable, in exchange for a reduced number of iterations.
3) The third example deals with the relaxation of relations with infinite cardinality. In particular it shows how one can obtain consistent estimates with a finite and prescribed amount of computation.

A. Application: Dealing with Parametric Uncertainty

To instantiate the model in Fig. 1, we need to obtain numbers for energy density, specific cost, and operating life for all batteries technologies we want to examine.

By browsing Wikipedia, one can find the figures in Table I.

TABLE I

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	150	2.50	600
LFP	90	1.50	1500

Should we trust those figures? Fortunately, we can easily deal with possible mistrust by introducing uncertain DPs.

Formally, we replace the DPs for *energy density*, *specific cost*, *operating life* in Fig. 1 with the corresponding Uncertain DPs with a configurable uncertainty. We can then solve the UDPs to obtain a lower bound and an upper bound to the solutions that can be presented to the user.

Fig. 24 shows the relation between the provided endurance and the minimal total mass required, when using uncertainty of 5%, 10%, 25% on the numbers above. Each panel shows two curves: the lower bound (best case analysis) and the upper bound (worst case analysis). In some cases, the lower bound is feasible, but the upper bound is not. For example, in panel *b*, for 10% uncertainty, we can conclude that, notwithstanding the uncertainty, there exists a solution for endurance ≤ 1.3 hours, while for higher endurance, because the upper bound is infeasible, we cannot conclude that there is a solution — though, because the lower bound is feasible, we cannot conclude that a solution does not exist (Fig. 24c).

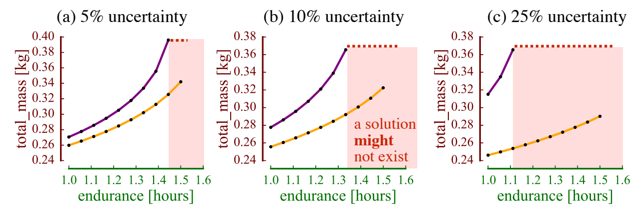


Fig. 24. Uncertain relation between endurance and the minimal total mass required, obtained by solving the example in Fig. 1 for different values of the uncertainty on the characteristics of the batteries. As the uncertainty increases, there are no solutions for the worst case.

B. Application: Introducing Tolerances

Another application of the theory is the introduction of tolerances for any variable in the optimization problem. For example, one might not care about the variations of the battery mass below, say, 1 g. One can then introduce a ± 1 g uncertainty in the definition of the problem by adding a UDP hereby called "uncertain identity".

1) The uncertain identity: Let $\alpha > 0$ be a step size. Define floor_α and ceil_α to be the floor and ceil with step size α (Fig. 25). By construction, $\mathsf{floor}_\alpha \preceq_{\mathsf{DP}} \mathsf{Id} \preceq_{\mathsf{DP}} \mathsf{ceil}_\alpha$.

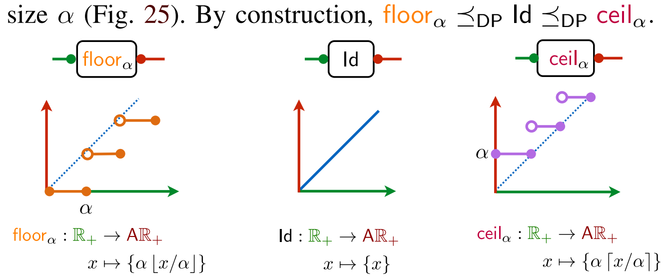


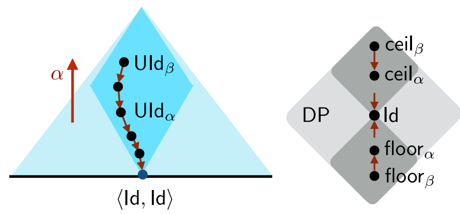
Fig. 25. The identity and its two relaxations floor_α and ceil_α.

Let $\mathsf{UId}_\alpha \doteq \langle \mathsf{floor}_\alpha, \mathsf{ceil}_\alpha \rangle$ be the "uncertain identity". For $0 < \alpha < \beta$, it holds that

$$\mathsf{Id} \prec_{\mathsf{UDP}} \mathsf{UId}_\alpha \prec_{\mathsf{UDP}} \mathsf{UId}_\beta.$$

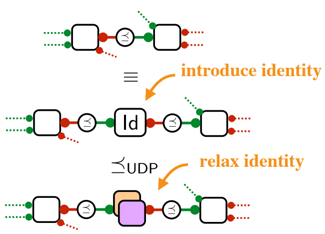
Therefore, the sequence UId_α is a descending chain that converges to Id as $\alpha \to 0$ (Fig. 26).

Fig. 26.



2) Approximations in MCDP: We can take any edge in an MCDP and apply this relaxation. Formally, we first introduce an identity Id and then relax it using UId_α (Fig. 27).

Fig. 27.



Mathematically, given an MCDP $\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}\rangle$, we generate a UMCDP $\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_\alpha\rangle$, where the new valuation $\boldsymbol{v}_\alpha$ agrees with $\boldsymbol{v}$ except on a particular atom $a \in \mathcal{A}$, which is replaced by the series of the original $\boldsymbol{v}(a)$ and the approximation UId_α:

$$\boldsymbol{v}_\alpha(a) \doteq \mathsf{series}(\mathsf{UId}_\alpha, \boldsymbol{v}(a))$$

Call the original and approximated DPs dp and dp_α:

$$\mathsf{dp} \doteq \Phi\left[\!\left[\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}\rangle\right]\!\right], \qquad \mathsf{dp}_\alpha \doteq \Phi\left[\!\left[\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_\alpha\rangle\right]\!\right].$$

Because $\boldsymbol{v} \preceq_V \boldsymbol{v}_\alpha$ (in the sense of Def. 19), Theorem 20 implies that

$$\mathsf{dp} \preceq_{\mathsf{UDP}} \mathsf{dp}_\alpha.$$

This means that we can solve Ldp_α and Udp_α and obtain upper and lower bounds for dp. Furthermore, by varying α, we can construct an approximating sequence of DPs whose solution will converge to the solution of the original MCDP.

Numerical results: This procedure was applied to the example model in Fig. 1 by introducing a tolerance to the "power" variable for the actuation. The tolerance α is chosen at logarithmic intervals between 0.01 mW and 1 W. Fig. 28a shows the solutions of the minimal mass required for Ldp_α and Udp_α, as a function of α. Fig. 28a confirms the consistency results predicted by the theory. First, if the solutions for both Ldp_α and Udp_α exist, then they are ordered ($\mathsf{Ldp}_\alpha(\mathsf{f}) \preceq \mathsf{Udp}_\alpha(\mathsf{f})$). Second, as α decreases, the interval shrinks. Third, the bounds are consistent (the solution for the original DP is always contained in the bound).

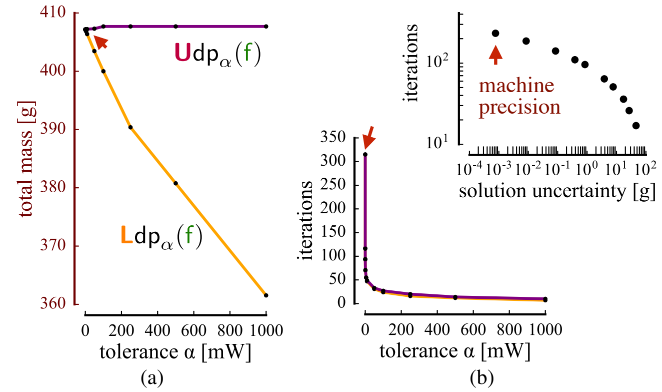


Fig. 28. Results of model in Fig. 1 when tolerance is applied to the actuation power resource. Please see the supplementary materials for more details.

Next, it is interesting to consider the computational complexity. Fig. 28b shows the number of iterations as a function of the resolution α, and the trade-off of the uncertainty of the solution and the computational resources spent. This shows that this approximation scheme is an effective way to reduce the computation load while maintaining a consistent estimate.

C. Application: Relaxation for relations with infinite cardinality

Another way in which uncertain DPs can be used is to construct approximations of DPs that would be too expensive to solve exactly. For example, consider a relation like

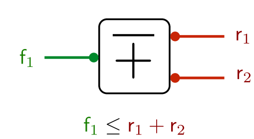
$$\text{travel\_distance} \leq \text{velocity} \times \text{endurance}, \tag{8}$$

which appears in the model in Fig. 1. If we take these three quantities in (8) as belonging to $\mathbb{R}$, then, for each value of the travel distance, there are infinite pairs of $\langle \text{velocity}, \text{endurance}\rangle$ that are feasible. (On a computer, where the quantities could be represented as floating point numbers, the combinations are properly not "infinite", but, still, extremely large.)

We can avoid considering all combinations by creating a sequence of uncertain DPs that use finite and prescribed computation.

1) Relaxations for addition: Consider a monotone relation between some functionality $\mathsf{f}_1 \in \mathbb{R}_+$ and resources $\mathsf{r}_1, \mathsf{r}_2 \in \mathbb{R}_+$ described by the constraint that $\mathsf{f}_1 \leq \mathsf{r}_1 + \mathsf{r}_2$ (Fig. 29). For example, this could represent the case where there are two batteries providing the power f_1, and we need to decide how much to allocate to the first (r_1) or the second (r_2).

Fig. 29.



The formal definition of this constraint as an DP is

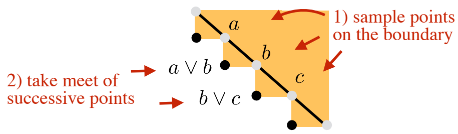
$$\begin{aligned} \mp : \mathbb{R}_+ &\to \mathsf{A}(\mathbb{R}_+ \times \mathbb{R}_+), \\ \mathsf{f}_1 &\mapsto \{\langle x, \mathsf{f}_1 - x\rangle \mid x \in \mathbb{R}_+\}. \end{aligned}$$

Note that, for each value f_1, $\mp(\mathsf{f}_1)$ is a set of infinite cardinality.

We will now define two sequences of relaxations for $\mp$ with a fixed number of solutions $n \geq 1$.

Using uniform sampling: We will first define a sequence of UDPs S_n based on uniform sampling. Let $\mathsf{U}S_n$ consist of n points sampled on the segment with extrema $\langle 0, \mathsf{f}_1\rangle$ and $\langle \mathsf{f}_1, 0\rangle$. For $\mathsf{L}S_n$, sample $n+1$ points on the segment and take the *meet* of successive points (Fig. 30).

Fig. 30.



The first elements of the sequences are shown in Fig. 31. One can easily prove that $\mathsf{L}S_n \preceq_{\mathsf{DP}} \mp \preceq_{\mathsf{DP}} \mathsf{U}S_n$, and thus S_n is a relaxation of $\mp$, in the sense that $\mp \preceq_{\mathsf{UDP}} S_n$. Moreover, S_n converges to $\mp$ as $n \to \infty$.

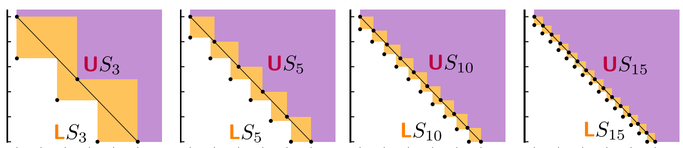


Fig. 31. Approximations to $\mp$ using the uniform sampling sequence S_n.

However, note that the convergence is not monotonic: $S_{n+1} \not\preceq_{\mathsf{UDP}} S_n$. The situation can be represented graphically as in Fig. 34a. The sequence S_n eventually converges to $\mp$, but it is not a descending chain. This means that it is not true, in general, that the solution to the MCDP obtained by plugging in S_{n+1} gives smaller bounds than S_n.

Relaxation based on Van Der Corput sequence: We can easily create an approximation sequence $V : \mathbb{N} \to \mathsf{UDP}$ that converges monotonically using Var Der Corput (VDC) sampling [13, Section 5.2]. Let $\mathsf{vdc}(n)$ be the VDC sequence of n elements in the interval $[0,1]$. The first elements of the VDC are $0, 0.5, 0.25, 0.75, 0.125, \dots$. The sequence is guaranteed to satisfy $\mathsf{vdc}(n) \subseteq \mathsf{vdc}(n+1)$ and to minimize the discrepancy. The upper bound $\mathsf{U}V_n$ is defined as sampling the segment with extrema $\langle 0, \mathsf{f}_1\rangle$ and $\langle \mathsf{f}_1, 0\rangle$ using the VDC sequence:

$$\mathsf{U}V_n : \mathsf{f}_1 \mapsto \{\langle \mathsf{f}_1 x, \mathsf{f}_1(1-x)\rangle \mid x \in \mathsf{vdc}(n)\}.$$

The lower bound $\mathsf{L}V_n$ is defined by taking meets of successive points, according to the procedure in Fig. 30.

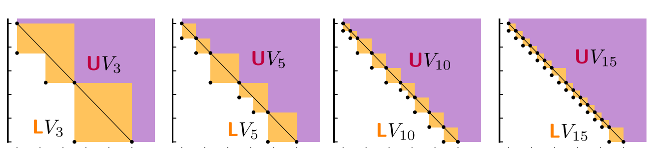
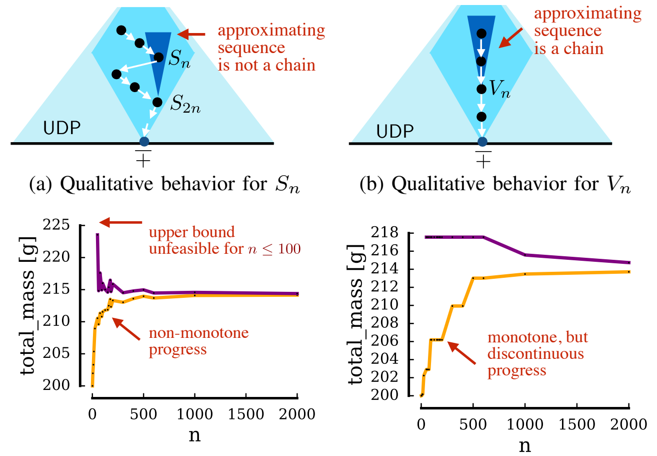


Fig. 32. Approximations to $\mp$ using the Van Der Corput sequence V_n.

For this sequence, one can prove that not only $\mp \preceq_{\mathsf{UDP}} V_n$, but also that the convergence is uniform, in the sense that $\mp \preceq_{\mathsf{UDP}} V_{n+1} \preceq_{\mathsf{UDP}} V_n$. The situation is represented graphically in Fig. 34b: the sequence is a descending chain that converges to $\mp$.



(a) Qualitative behavior for S_n (b) Qualitative behavior for V_n

(c) Numerical results for S_n (d) Numerical results for V_n

Fig. 34. Solutions to the example in Fig. 1, applying relaxations for the relation travel_distance $\leq$ velocity $\times$ endurance using the uniform sampling sequence and the VDC sampling sequence. The uniform sampling sequence S_n does not converge monotonically (panel *a*); therefore the progress is not monotonic (panel *c*). Conversely, the Van Der Corput sequence V_n is a descending chain (panel *b*), which results in monotonic progress (panel *d*).

2) Inverse of multiplication: The case of multiplication can be treated analogously to the case of addition. By taking the logarithm, the inequality $\mathsf{f}_1 \leq \mathsf{r}_1\mathsf{r}_2$ can be rewritten as $\log(\mathsf{f}_1) \leq \log(\mathsf{r}_1) + \log(\mathsf{r}_2)$. So we can repeat the constructions done for addition. The VDC sequence are shown in Fig. 33.

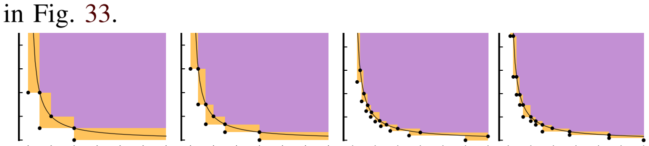


Fig. 33. Van Der Corput relaxations for the relation $\mathsf{f}_1 \leq \mathsf{r}_1\mathsf{r}_2$.

3) Numerical example: We have applied this relaxation to the relation travel distance $\leq$ velocity $\times$ endurance in the MCDP in Fig. 1. Thanks to this theory, we can obtain estimates of the solutions using bounded computation, even though that relation has infinite cardinality.

Fig. 34c shows the result using uniform sampling, and Fig. 34d shows the result using VDC sampling. As predicted by the theory, uniform sampling does not give monotone convergence, while VDC sampling does.

VIII. Conclusions and future work

Monotone Co-Design Problems (MCDPs) provide a compositional theory of "co-design" that describes co-design constraints among different subsystems in a complex system, such as a robotic system.

This paper dealt with the introduction of uncertainty in the framework, specifically, interval uncertainty.

Uncertainty can be used in two roles. First, it can be used to describe limited knowledge in the models. For example, in Section VII-A, we have seen how this can be applied to model mistrust about numbers from Wikipedia. Second, uncertainty allows to generate relaxations of the problem. We have seen two applications: introducing an allowed tolerance in one particular variable (Section VII-B), and dealing with relations with infinite cardinality using bounded computation resources (Section VII-C).

Future work includes strengthening these results. For example, we are not able to predict the resulting uncertainty in the solution before actually computing it; ideally, one would like to know how much computation is needed (measured by the number of points in the antichain approximation) for a given value of the uncertainty that the user can accept.

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 (ACC)*. 2016.

[3] A. Censi. "A Mathematical Theory of Co-Design". In: *CoRR* abs/1512.08055 (2015). URL: http://arxiv.org/abs/1512.08055.

[4] D. Bertsimas, D. B. Brown, and C. Caramanis. "Theory and Applications of Robust Optimization." In: *SIAM Review* 53.3 (2011), pp. 464–501. DOI: 10.1137/080734510.

[5] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. *Robust Optimization*. Walter de Gruyter GmbH, 2009. DOI: 10.1515/9781400831050.

[6] B. Davey and H. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002. ISBN: 9780521784511. DOI: 10.1017/cbo9780511809088.

[7] 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.

[8] R. Duffin. "Topology of series-parallel networks". In: *Journal of Mathematical Analysis and Applications* 10.2 (1965), pp. 303–318. DOI: 10.1016/0022-247x(65)90125-3.

[9] A. Joyal, R. Street, and D. Verity. "Traced monoidal categories". In: *Math. Proc. Camb. Phil. Soc.* 119.03 (1996), p. 447. DOI: 10.1017/s0305004100074338.

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

[11] G. Ştefănescu. *Network Algebra*. Springer Science + Business Media, 2000. DOI: 10.1007/978-1-4471-0479-7.

[12] J. Jezek. *Universal Algebra*. 2008.

[13] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, May 2006. URL: http://planning.cs.uiuc.edu/.

Appendix

A. Proofs

1) Proofs of well-formedness of Def. 18: As some preliminary business, we need to prove that Def. 18 is well formed, in the sense that the way the semantics function Φ is defined, it returns a UDP for each argument. This is not obvious from Def. 18.

For example, for $\Phi[\![\mathcal{A}, \mathsf{series}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v}]\!]$, the definition gives values for $\mathbf{L}\Phi[\![\mathcal{A}, \mathsf{series}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v}]\!]$ and $\mathbf{U}\Phi[\![\mathcal{A}, \mathsf{series}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v}]\!]$ separately, without checking that

$$\mathbf{L}\Phi[\![\mathcal{A}, \mathsf{series}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v}]\!] \preceq_{\mathsf{DP}} \mathbf{U}\Phi[\![\mathcal{A}, \mathsf{series}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v}]\!].$$

The following lemma provides the proof for that.

Lemma 21. *Def. 18 is well formed, in the sense that*

$$\mathbf{L}\Phi[\![\langle\mathcal{A}, \mathsf{series}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v}\rangle]\!] \preceq_{\mathsf{DP}} \mathbf{U}\Phi[\![\langle\mathcal{A}, \mathsf{series}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v}\rangle]\!], \tag{9}$$

$$\mathbf{L}\Phi[\![\langle\mathcal{A}, \mathsf{par}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v}\rangle]\!] \preceq_{\mathsf{DP}} \mathbf{U}\Phi[\![\langle\mathcal{A}, \mathsf{par}(\mathbf{T}_1, \mathbf{T}_2), \boldsymbol{v}\rangle]\!], \tag{10}$$

$$\mathbf{L}\Phi[\![\langle\mathcal{A}, \mathsf{loop}(\mathbf{T}), \boldsymbol{v}\rangle]\!] \preceq_{\mathsf{DP}} \mathbf{U}\Phi[\![\langle\mathcal{A}, \mathsf{loop}(\mathbf{T}), \boldsymbol{v}\rangle]\!]. \tag{11}$$

Proof. Proving (9)—(11) can be reduced to proving the following three results, for any $x, y \in \mathsf{UDP}$:

$$(\mathbf{L}x \circledcirc \mathbf{L}y) \preceq_{\mathsf{DP}} (\mathbf{U}x \circledcirc \mathbf{U}y),$$
$$(\mathbf{L}x \otimes \mathbf{L}y) \preceq_{\mathsf{DP}} (\mathbf{U}x \otimes \mathbf{U}y),$$
$$(\mathbf{L}x)^{\dagger} \preceq_{\mathsf{DP}} (\mathbf{U}x)^{\dagger}.$$

These are given in Lemma 22, Lemma 23, Lemma 24.

Lemma 22. $(\mathbf{L}x \circledcirc \mathbf{L}y) \preceq_{\mathsf{DP}} (\mathbf{U}x \circledcirc \mathbf{U}y)$.

Proof. First prove that $\circledcirc$ is monotone in each argument (proved as Lemma 25). Then note that

$$(\mathbf{L}x \circledcirc \mathbf{L}y) \preceq_{\mathsf{DP}} (\mathbf{L}x \circledcirc \mathbf{U}y) \preceq_{\mathsf{DP}} (\mathbf{U}x \circledcirc \mathbf{U}y).$$

Lemma 23. $(\mathbf{L}x \otimes \mathbf{L}y) \preceq_{\mathsf{DP}} (\mathbf{U}x \otimes \mathbf{U}y)$.

Proof. The proof is entirely equivalent to the proof of Lemma 22. First prove that par is monotone in each argument (proved as Lemma 26). Then note that

$$(\mathbf{L}x \otimes \mathbf{L}y) \preceq_{\mathsf{DP}} (\mathbf{L}x \otimes \mathbf{U}y) \preceq_{\mathsf{DP}} (\mathbf{U}x \otimes \mathbf{U}y).$$

Lemma 24. $(\mathbf{L}x)^{\dagger} \preceq_{\mathsf{DP}} (\mathbf{U}x)^{\dagger}$.

Proof. This follows from the fact that $\dagger$ is monotone (Lemma 28).

2) Monotonicity lemmas for DP: These lemmas are used in the proofs above.

Lemma 25. $\circledcirc : \mathsf{DP} \times \mathsf{DP} \to \mathsf{DP}$ *is monotone on* $\langle \mathsf{DP}, \preceq_{\mathsf{DP}} \rangle$.

Proof. In Def. 11, $\circledcirc$ is defined as follows for two maps $h_1 : \mathcal{F}_1 \to \mathsf{A}\mathcal{R}_1$ and $h_2 : \mathcal{F}_2 \to \mathsf{A}\mathcal{R}_2$:

$$h_1 \circledcirc h_2 = \underset{\preceq_{\mathcal{R}_2}}{\operatorname{Min}} \uparrow \bigcup_{s \in h_1(\mathsf{f})} h_2(s).$$

It is useful to decompose this expression as the composition of three maps:

$$h_1 \circledcirc h_2 = m \circ g[h_2] \circ h_1,$$

where "$\circ$" is the usual map composition, and g and m are defined as follows:

$$\begin{aligned} g[h_2] : \mathsf{A}\mathcal{R}_1 &\to \mathsf{U}\mathcal{R}_2, \\ R &\mapsto \uparrow \bigcup_{s \in R} h_2(s), \end{aligned}$$

and

$$\begin{aligned} m : \mathsf{U}\mathcal{R}_2 &\to \mathsf{A}\mathcal{R}_2, \\ R &\mapsto \underset{\preceq_{\mathcal{R}_2}}{\operatorname{Min}} R. \end{aligned}$$

From the following facts:

- m is monotone.
- $g[h_2]$ is monotone in h_2.
- $f_1 \circ f_2$ is monotone in each argument if the other argument is monotone.

Then the thesis follows.

Lemma 26. $\otimes : \mathsf{DP} \times \mathsf{DP} \to \mathsf{DP}$ *is monotone on* $\langle \mathsf{DP}, \preceq_{\mathsf{DP}} \rangle$.

Proof. The definition of $\otimes$ (Def. 10) is:

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

Because of symmetry, it suffices to prove that $\otimes$ is monotone in the first argument, leaving the second fixed.

We need to prove that for any two DPs h_a, h_b such that

$$h_a \preceq_{\mathsf{DP}} h_b, \tag{12}$$

and for any fixed $\overline{h}$, then

$$h_a \otimes \overline{h} \preceq_{\mathsf{DP}} h_b \otimes \overline{h}.$$

Let $R = \overline{h}(\mathsf{f}_2)$. Then we have that

$$[h_a \otimes \overline{h}](\mathsf{f}_1, \mathsf{f}_2) = h_a(\mathsf{f}_1) \times R,$$
$$[h_b \otimes \overline{h}](\mathsf{f}_1, \mathsf{f}_2) = h_b(\mathsf{f}_1) \times R.$$

Because of (12), we know that

$$h_a(\mathsf{f}_1) \preceq_{\mathsf{A}\mathcal{R}_1} h_b(\mathsf{f}_1).$$

So the thesis follows from proving that the product of antichains is monotone (Lemma 27).

Lemma 27. *The product of antichains* $\times : \mathsf{A}\mathcal{R}_1 \times \mathsf{A}\mathcal{R}_2 \to \mathsf{A}(\mathcal{R}_1 \times \mathcal{R}_2)$ *is monotone.*

Lemma 28. $\dagger : \mathsf{DP} \to \mathsf{DP}$ *is monotone on* $\langle \mathsf{DP}, \preceq_{\mathsf{DP}} \rangle$.

Proof. Let $h_1 \preceq_{\mathsf{DP}} h_2$. Then we can prove that $h_1^{\dagger} \preceq_{\mathsf{DP}} h_2^{\dagger}$. From the definition of $\dagger$ (Def. 12), we have that

$$h_1^{\dagger}(\mathsf{f}_1) = \mathsf{lfp}(\Psi_{\mathsf{f}}^{h_1}),$$
$$h_2^{\dagger}(\mathsf{f}_2) = \mathsf{lfp}(\Psi_{\mathsf{f}}^{h_2}),$$

with $\Psi_{\mathsf{f}_1}^{h}$ defined as

$$\begin{aligned} \Psi_{\mathsf{f}_1}^{h} : \mathsf{A}\mathcal{R} &\to \mathsf{A}\mathcal{R}, \\ R &\mapsto \underset{\preceq_{\mathcal{R}}}{\operatorname{Min}} \bigcup_{\mathsf{r} \in R} h(\mathsf{f}_1, \mathsf{r}) \cap \uparrow \mathsf{r}. \end{aligned}$$

The least fixed point operator lfp is monotone, so we are left to check that the map

$$h \mapsto \Psi^{h}_{\mathsf{f}_1}$$

is monotone. That is the case, because if $h_1 \preceq_{\mathsf{DP}} h_2$ then

$$\left[\bigcup_{\mathsf{r}\in R} h_1(\mathsf{f}_1, \mathsf{r}) \cap \uparrow \mathsf{r}\right] \preceq_{\mathsf{A}\mathcal{R}} \left[\bigcup_{\mathsf{r}\in R} h_2(\mathsf{f}_1, \mathsf{r}) \cap \uparrow \mathsf{r}\right].$$

3) Monotonicity of semantics φ:

Lemma 29 (φ is monotone in the valuation)**.** *Suppose that $\boldsymbol{v}_1, \boldsymbol{v}_2 : \mathcal{A} \to \mathsf{DP}$ are two valuations for which it holds that $\boldsymbol{v}_1(a) \preceq_{\mathsf{DP}} \boldsymbol{v}_2(a)$. Then $\varphi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_1\rangle]\!] \preceq_{\mathsf{DP}} \varphi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_2\rangle]\!]$.*

Proof. Given the recursive definition of Def. 9, we need to prove this just for the base case and for the recursive cases.

The base case, given in (2), is

$$\varphi[\![\langle \mathcal{A}, a, \boldsymbol{v}\rangle]\!] \doteq \boldsymbol{v}(a), \qquad \text{for all } a \in \mathcal{A}.$$

We have

$$\begin{aligned} \varphi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_1\rangle]\!] &= \boldsymbol{v}_1(a) \\ \varphi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_2\rangle]\!] &= \boldsymbol{v}_2(a) \end{aligned}$$

and $\boldsymbol{v}_1(a) \preceq_{\mathsf{DP}} \boldsymbol{v}_2(a)$ by assumption.

For the recursive cases, (3)–(5), the thesis follows from the monotonicity of $\circledcirc$, $\otimes$, $\dagger$, proved in Lemma 26, Lemma 25, Lemma 28.

4) Proof of the main result, Theorem 20: We restate the theorem.

Theorem 20. *If*

$$\boldsymbol{v}_1 \preceq_V \boldsymbol{v}_2$$

then

$$\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_1\rangle]\!] \preceq_{\mathsf{UDP}} \Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_2\rangle]\!].$$

Proof. From the definition of Φ and φ, we can derive that

$$\mathbf{L}\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}\rangle]\!] = \varphi[\![\langle \mathcal{A}, \mathbf{T}, \mathbf{L} \circ \boldsymbol{v}\rangle]\!]. \tag{13}$$

In particular, for $\boldsymbol{v} = \boldsymbol{v}_1$,

$$\mathbf{L}\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_1\rangle]\!] = \varphi[\![\langle \mathcal{A}, \mathbf{T}, \mathbf{L} \circ \boldsymbol{v}_1\rangle]\!]. \tag{14}$$

Because $\boldsymbol{v}_1(a) \preceq_{\mathsf{UDP}} \boldsymbol{v}_2(a)$, from Lemma 29,

$$\varphi[\![\langle \mathcal{A}, \mathbf{T}, \mathbf{L} \circ \boldsymbol{v}_1\rangle]\!] \preceq_{\mathsf{DP}} \varphi[\![\langle \mathcal{A}, \mathbf{T}, \mathbf{L} \circ \boldsymbol{v}_2\rangle]\!]. \tag{15}$$

From (13) again,

$$\varphi[\![\langle \mathcal{A}, \mathbf{T}, \mathbf{L} \circ \boldsymbol{v}_2\rangle]\!] = \mathbf{L}\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_2\rangle]\!]. \tag{16}$$

From (14), (15), (16) together,

$$\mathbf{L}\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_1\rangle]\!] \preceq_{\mathsf{DP}} \mathbf{L}\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_2\rangle]\!].$$

Repeating the same reasoning for $\mathbf{U}$, we have

$$\mathbf{U}\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_2\rangle]\!] \preceq_{\mathsf{DP}} \mathbf{U}\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_1\rangle]\!].$$

Therefore

$$\Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_1\rangle]\!] \preceq_{\mathsf{UDP}} \Phi[\![\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v}_2\rangle]\!].$$

B. Source code

The implementation is available at the repository http://github.com/AndreaCensi/mcdp/, in the branch "uncertainty_sep16".

C. Virtual machine

A VMWare virtual machine is available to reproduce the experiments at the URL https://www.dropbox.com/sh/nfpnfgjh9hpcgvh/AACVZfdVXxMoVqTYiHWaOwHAa?dl=0.

To reproduce the figures, log in with user password "mcdp"/"mcdp". Then execute the following commands:

```
$ cd ~/mcdp
$ source environment.sh
$ cd libraries/examples/uav_energetics/
        droneD_complete_templates.mcdplib
$ make clean
$ make paper-figures
```

C Supplementary materials: model definition

The figures contained in this Appendix describe a subset of the models used for optimization.

The goal is to give an idea of how a Monotone Co-Design Problem (MCDP) is formalized using the formal language MCDPL.

This Appendix does not explain the syntax of MCDPL. For details, please see the manual available at `http://mcdp.mit.edu`.

C.1 General template

All the MCDPs used in the experiments are instantiation of the same *template*, whose code is shown in Fig. C.1.

Figure C.1: Template `DroneCompleteTemplate`

```
template [
    Battery: `BatteryInterface,
    Actuation: `ActuationInterface,
    Perception: `PerceptionInterface,
    PowerApprox: `PowerApprox]
mcdp {
  provides travel_distance [km]
  provides num_missions [R]
  provides carry_payload [g]

  requires total_cost_ownership [$]
  requires total_mass [g]

  strategy = instance `droneD_complete_v2.Strategy

  actuation_energetics =
    instance specialize [
      Battery: Battery,
      Actuation: Actuation,
      PowerApprox: PowerApprox
    ] `ActuationEnergeticsTemplate

  actuation_energetics.endurance >= strategy.endurance
  actuation_energetics.velocity >= strategy.velocity
  actuation_energetics.num_missions >= num_missions
  actuation_energetics.extra_payload >= carry_payload
  strategy.distance >= travel_distance

  perception = instance Perception
  perception.velocity >= strategy.velocity

  actuation_energetics.extra_power >= perception.power

  required total_mass >= actuation_energetics.total_mass

  total_cost_ownership >=  actuation_energetics.total_cost
}
```

Note the top-level functionality:

- `travel_distance` [km];
- `num_missions` (unitless);
- `carry_payload` [g]

and the top-level resources:

- `total_mass` [g]
- `total_cost_ownership` [USD]

The template has four parameters:

- `Battery`: MCDP for energetics;
- `Actuation`: MCDP for actuation;
- `Perception`: MCDP for perception;
- `PowerApprox`: MCDP describing the tolerance for the power variable. This is used in Section VII-B.

Every experiment chooses different values for the parameters of this template.

The graphical representation of the template is shown in Fig. C.3. The dotted blue containers represent the "holes" that need to be filled to instantiate the template.

In turn, the template contains a specialization call to another template, called `ActuationEnergeticsTemplate`, whose code is shown in Fig. C.2 and whose graphical representation is shown in Fig. C.4.

Figure C.2: Template `ActuationEnergeticsTemplate`

```
template [
  Battery: `BatteryInterface,
  Actuation: `ActuationInterface,
  PowerApprox: `PowerApprox
] mcdp {
  provides endurance     [s]
  provides extra_payload [kg]
  provides extra_power   [W]
  provides num_missions [R]
  provides velocity  [m/s]

  requires total_cost [$]

  battery = instance  Battery
  actuation = instance Actuation

  total_power0 = power required by actuation + extra_power

  power_approx = instance PowerApprox
  total_power0 <= power_approx.power
  total_power = power required by power_approx

  capacity provided by battery >= provided endurance * total_power

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

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

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

  labor_cost = (10 $) * (maintenance required by battery)

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

  battery.missions >= num_missions

  requires total_mass >= total_mass
}
```

The template has functionality `endurance`, `extra_payload`, `extra_power`, `num_missions`, `velocity`, and two resources, `total_mass` and `total_cost`

Figure C.3: Graphical representation for the template `DroneCompleteTemplate` (Fig. ??)

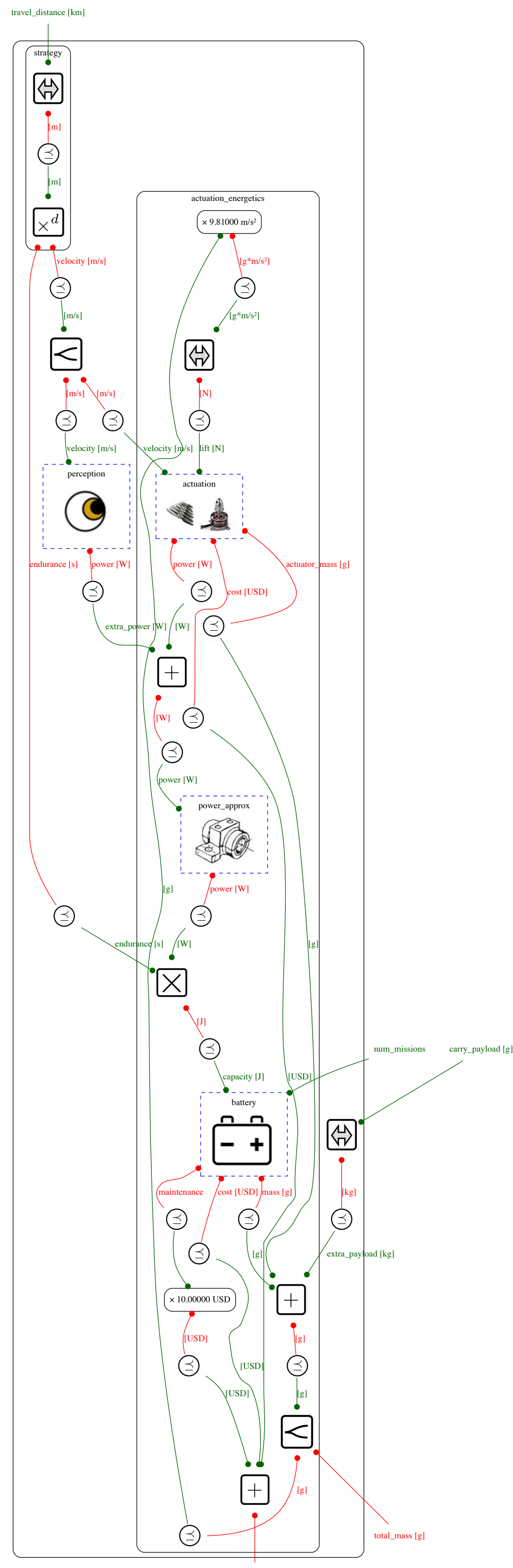
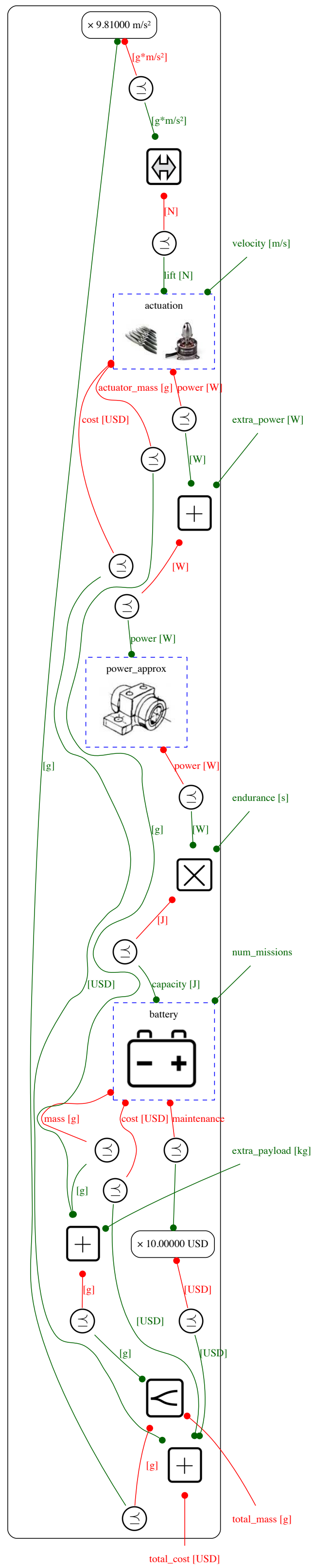


Figure C.4: Graphical representation for the template `ActuationEnergeticsTemplate` (Fig. C.2)



C.2 MCDP defining batteries properties

Fig. C.5 shows the definition of a single battery technology in terms of specific energy, specific cost, and lifetime (number of cycles).

Figure C.5: Definition of `Battery_LiPo` MCDP

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

    requires mass     [g]
    requires cost     [$]

    # Number of replacements
    requires maintenance [R]

    # Battery properties
    specific_energy = 150 Wh/kg
    specific_cost =  2.50 Wh/$
    cycles = 600 []

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

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

    # Cost is proportional to number of replacements
    cost >= (capacity / specific_cost) * num_replacements
}
```

Here a battery is abstracted as a DP with functionality:

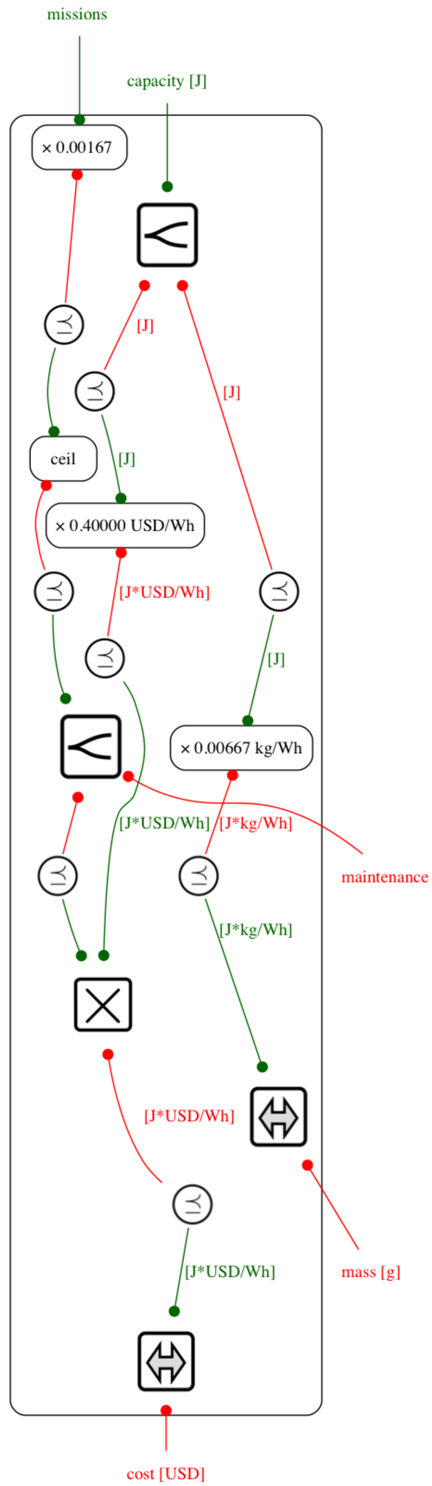
- `capacity` [J];
- `missions` (unitless)

and with resources:

- `mass` [g]
- `cost` [USD]

The corresponding graphical representation is shown in Fig. C.6.

Figure C.6: Definition of battery technology



Because this MCDP is completely specified, as opposed to the two *templates* shown earlier, we can show its algebraic representation, as defined in Def. 6.

The MCDP interpreter takes the code shown in Fig. C.5 and then builds an intermediate graphical representation like the one shown in Fig. C.6. Finally, it is compiled to an algebraic representation $\langle \mathcal{A}, \mathbf{T}, \boldsymbol{v} \rangle$, where $\mathbf{T}$ is a tree in the $\{\mathsf{series}, \mathsf{par}, \mathsf{loop}\}$ algebra.

A representation of $\mathbf{T}$ for this example is shown in Fig. C.7.

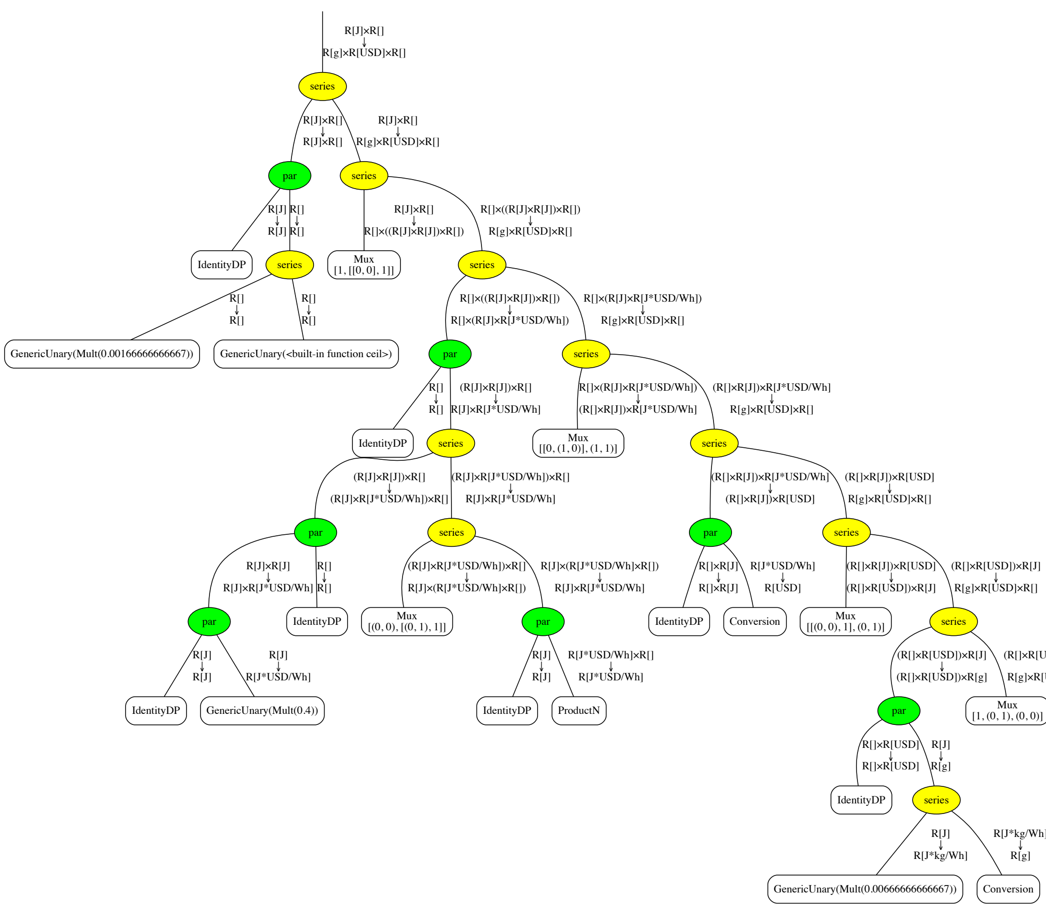
Each edge is the tree is labeled with the signature of the DP, in the form $\mathcal{F} \to \mathcal{R}$. The junctions are one of the $\{\mathsf{series}, \mathsf{par}, \mathsf{loop}\}$ operators. (The operator loop does not appear in this example.)

The leaves are labeled with a representation of the Python class that implements them. In particular, the frequently-appearing `Mux` type represents various multiplexing operations, such as

$$\langle x, y, z \rangle \mapsto \langle \langle z, y \rangle, x \rangle.$$

These are necessary to transform a graph into a tree representation.

Figure C.7: Algebraic representation for the example in Fig. C.5



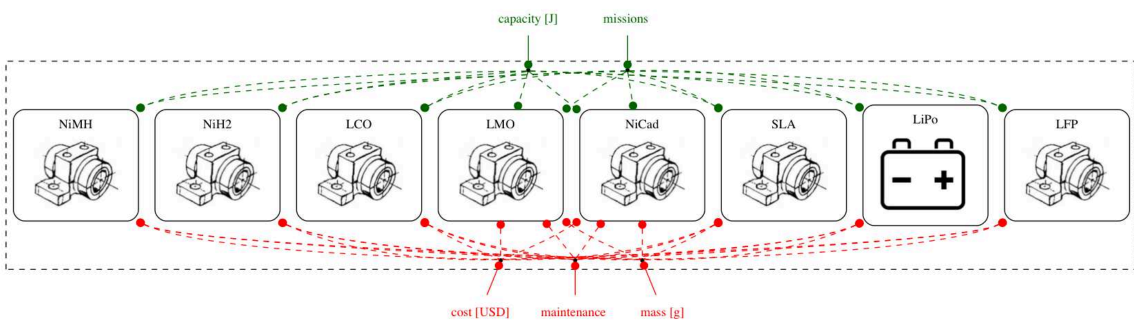
C.3 Choice between different batteries

Just like we defined `Battery_LiPo` (Fig. C.5), other batteries technologies are similarly defined, such as `Battery_NiMH`, `Battery_LCO`, etc.

Then we can easily express the choice between any of them using the keyword **`choose`**, as in Fig. C.8.

Figure C.8: Definition of the `Batteries` MCDP

```
choose(
          NiMH: (load Battery_NiMH),
          NiH2: (load Battery_NiH2),
           LCO: (load Battery_LCO),
           LMO: (load Battery_LMO),
         NiCad: (load Battery_NiCad),
           SLA: (load Battery_SLA),
          LiPo: (load Battery_LiPo),
           LFP: (load Battery_LFP)
)
```



The choice between different batteries is modeled by a *coproduct* operator. This is another type of junction, in addition to series, par, loop that was not described in the paper.

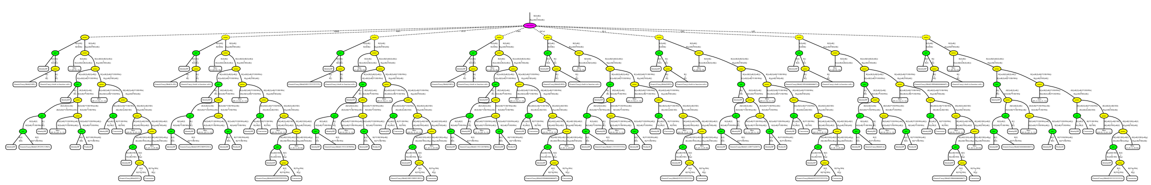
Formally, the coproduct operator it is defined as follows:

$$h_1 \sqcup \cdots \sqcup h_n : \mathcal{F} \to \mathsf{A}\mathcal{R}, \tag{1}$$

$$\mathrm{f} \mapsto \underset{\preceq_{\mathcal{R}}}{\mathrm{Min}} \left(h_1(\mathrm{f}) \cup \cdots \cup h_n(\mathrm{f}) \right). \tag{2}$$

The algebraic representation (Fig. C.9) contains then one branch for each type of battery.

Figure C.9: Algebraic representation of the `Batteries` MCDP



C.4 Describing uncertainty

This is a description of the Uncertain MCDPs used in the experiments in Section VII-A.

MCDPL has an `Uncertain` operator that can describe interval uncertainty.

For example, the MCDP in Fig. C.5 is rewritten with uncertainty to obtain the code in Fig. C.10.

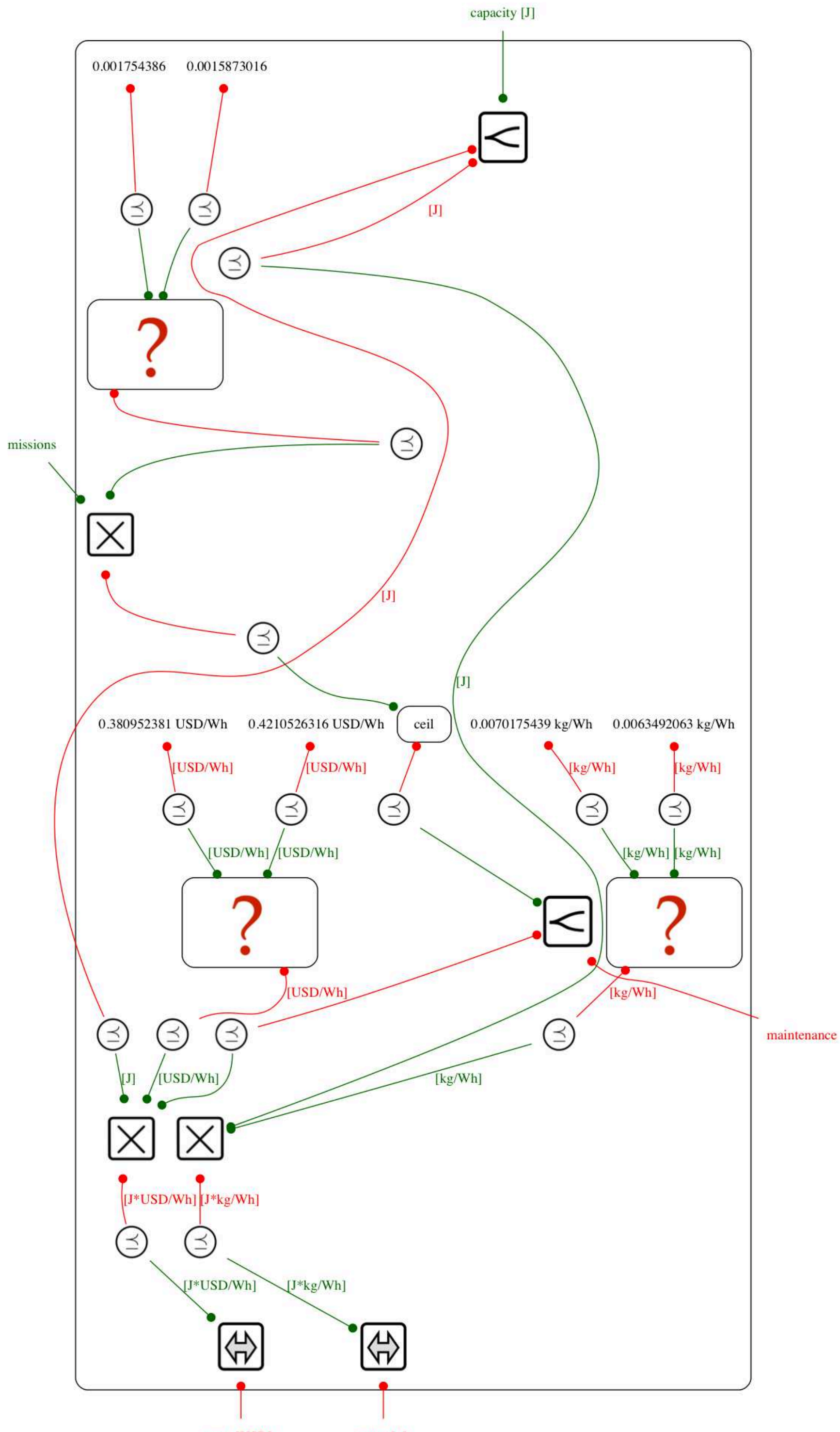
The figures have a 5% uncertainty added to them.

Figure C.10: Definition of `Battery_LiPo` MCDP with 5% uncertainty on parameters

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

    requires mass     [g]
    requires cost     [USD]

    # Number of replacements
    requires maintenance [R]

    # Battery properties
    specific_energy_inv = Uncertain(1.0 []/ 157.5 Wh/kg, 1.0 [] /  142.5 Wh/kg)
    specific_cost_inv = Uncertain(1.0 [] / 2.625 Wh/USD, 1.0 [] / 2.375 Wh/USD)
    cycles_inv = Uncertain(1.0 []/630.0 [], 1.0[]/ 570.0 [])

    # Constraint between mass and capacity
    massc = provided capacity * specific_energy_inv

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

    # Cost is proportional to number of replacements
    costc = (provided capacity * specific_cost_inv) * num_replacements

    required cost >= costc
    required mass >= massc
}
```

In the graphical representation, the uncertainty is represented as “uncertainty gates” that have two branches: one for best case and one for worst case (Fig. C.11).

Figure C.11: Graphical representation of uncertain MCDP in Fig. C.10



C.5 Specialization of templates

Once all the single pieces are defined, then the final MCDP is assembled using the `specialize` keyword.

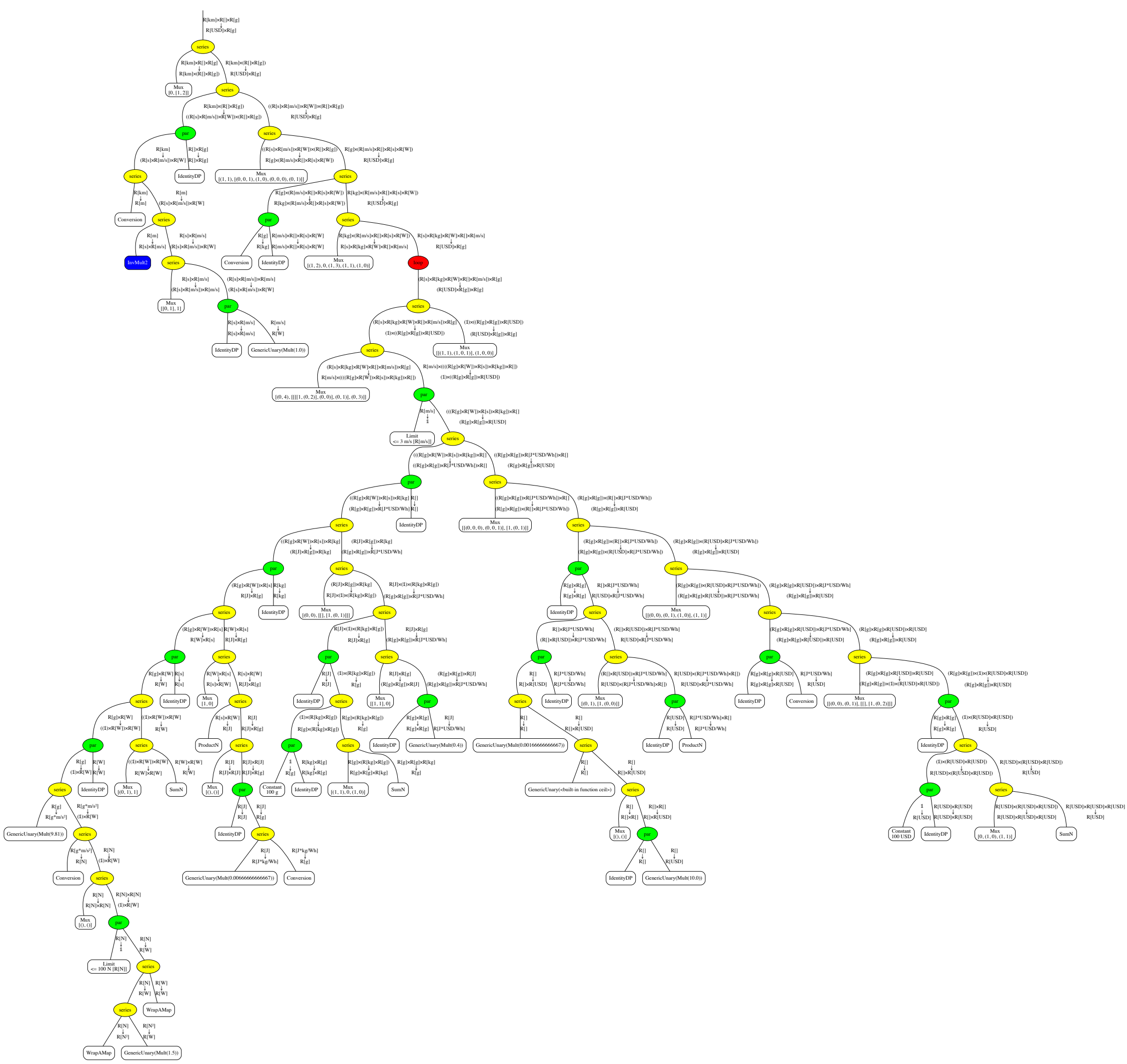
For example, the following code specializes the template using only the `Battery_LiPo` MCDP.

Figure C.12:

```
specialize [
   Battery: `batteries_nodisc.Battery_LiPo,
   Actuation: `droneD_complete_v2.Actuation,
   Perception: `Perception1,
   PowerApprox: `PowerApprox
]
`DroneCompleteTemplate
```

The algebraic representation is shown in Fig. C.13.

Figure C.13: Algebraic representation of MCDP in Fig. C.12



The following code specializes the template using the coproduct of all batteries, each having an uncertain specification.

Figure C.14:

```
specialize [
  Battery: `batteries_uncertain1.batteries,
  Actuation: `droneD_complete_v2.Actuation,
  PowerApprox: mcdp {
    provides power [W]
    requires power [W]

    required power  >= approxu(provided power, 1 mW)
   }
] `ActuationEnergeticsTemplate
```

The algebraic representation is shown in Fig. C.15.

The blue nodes are the uncertain nodes (UDPs).

Figure C.15: Algebraic representation of MCDP in Fig. C.14