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. ??)

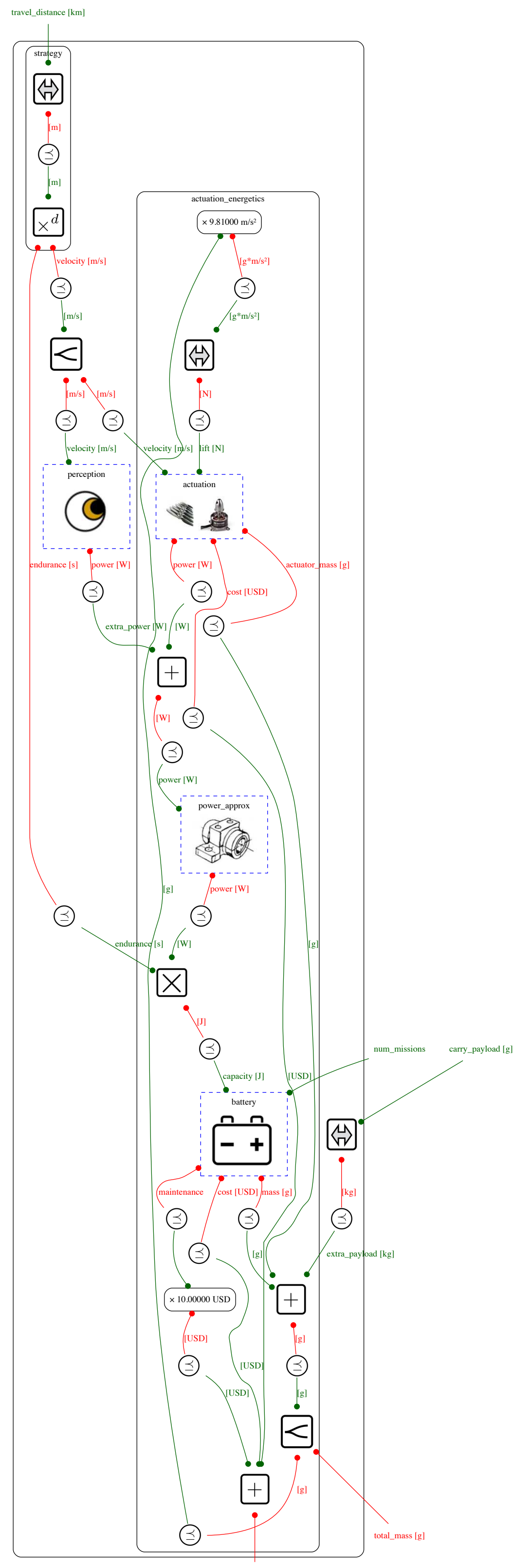
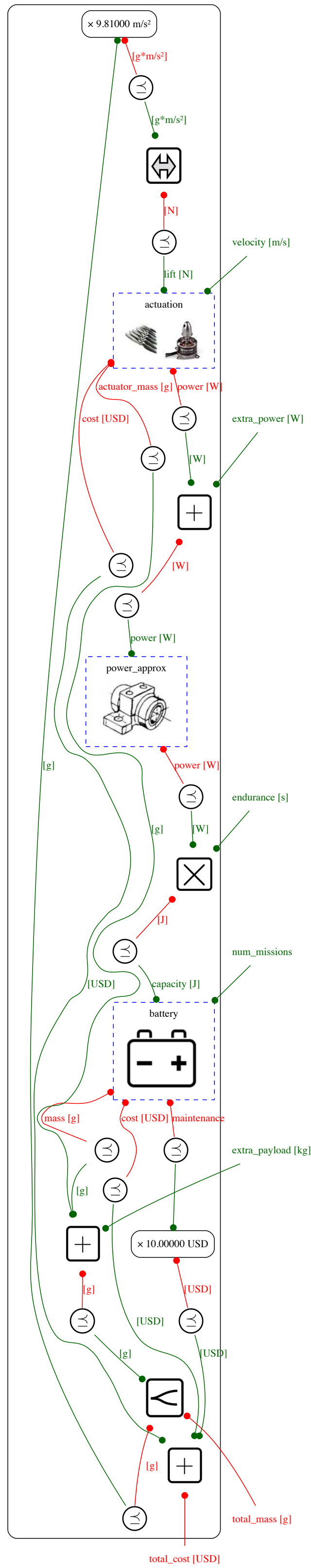


Figure C.4: Graphical representation for the template `ActuationEnergeticsTemplate` (Fig. C.2)



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:

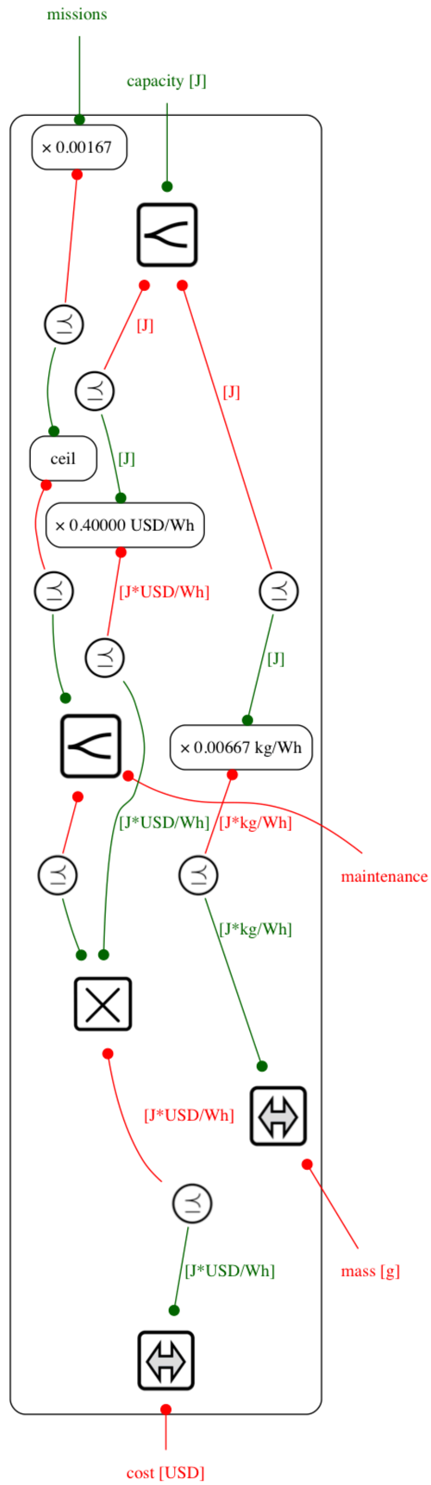
- `capacity` [J];
- `missions` (unitless)

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.

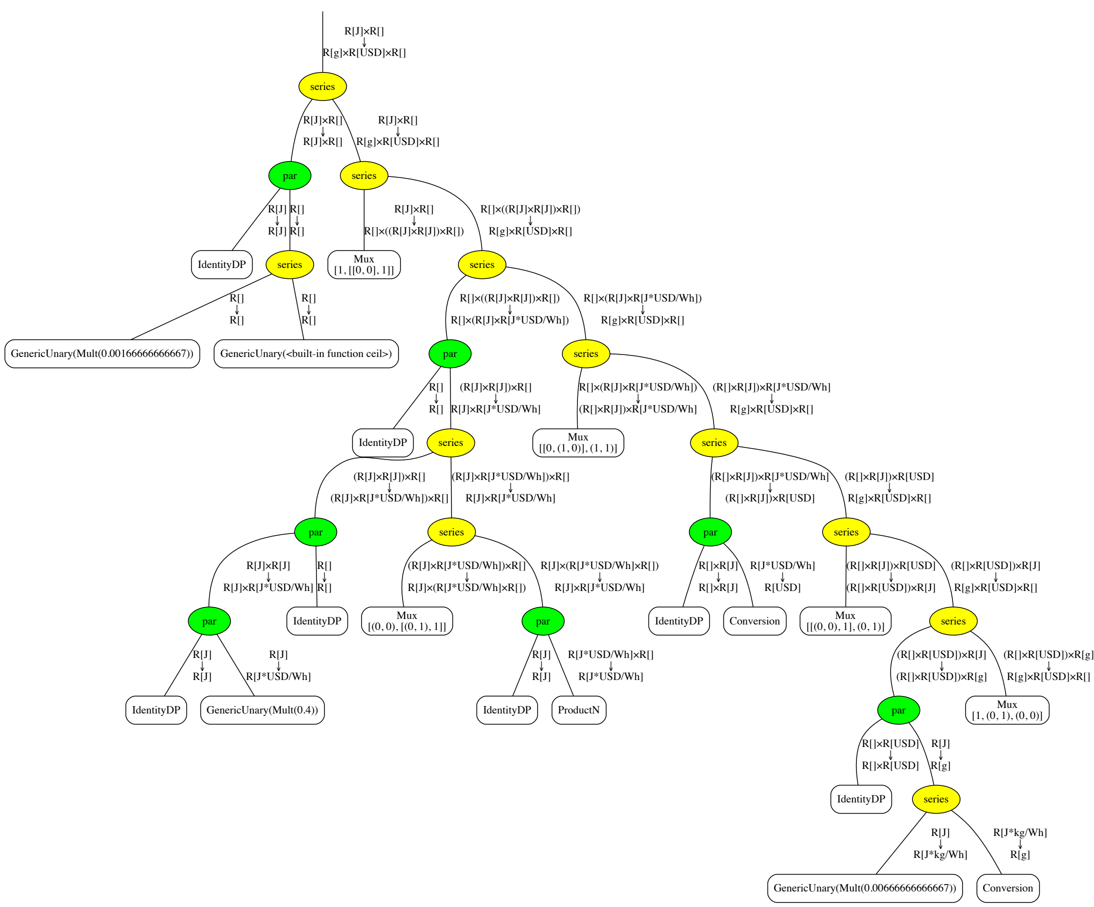
Each edge is the tree is labeled with the signature of the DP, in the form $\mathcal{F} \to \mathcal{R}$. The junctions are one of the $\{\mathsf{series}, \mathsf{par}, \mathsf{loop}\}$ operators. (The operator loop does not appear in this example.)

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



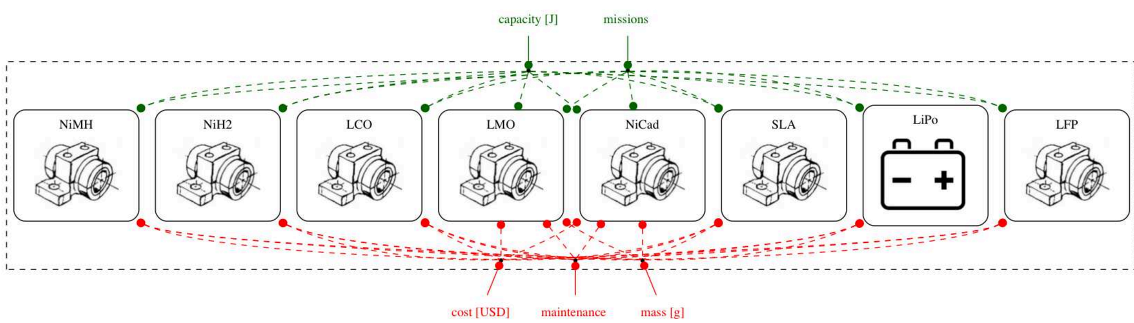
C.3 Choice between different batteries

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)
)
```



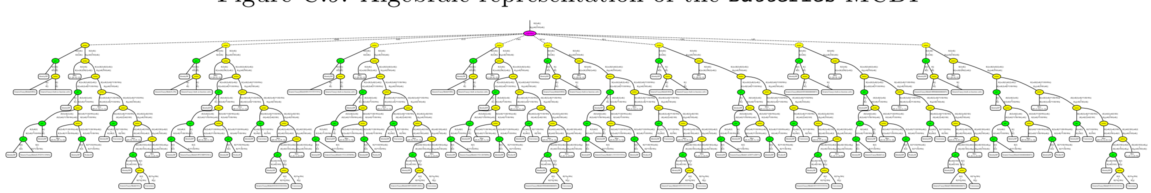
The choice between different batteries is modeled by a *coproduct* operator. This is another type of junction, in addition to series, par, loop that was not described in the paper.

Formally, the coproduct operator it is defined as follows:

$$
\begin{aligned}
h_1 \sqcup \cdots \sqcup h_n : \mathcal{F} &\to \mathsf{A}\mathcal{R}, && (1)\\
\mathsf{f} &\mapsto \operatorname*{Min}_{\preceq_{\mathcal{R}}} \left(h_1(\mathsf{f}) \cup \cdots \cup h_n(\mathsf{f}) \right). && (2)
\end{aligned}
$$

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.

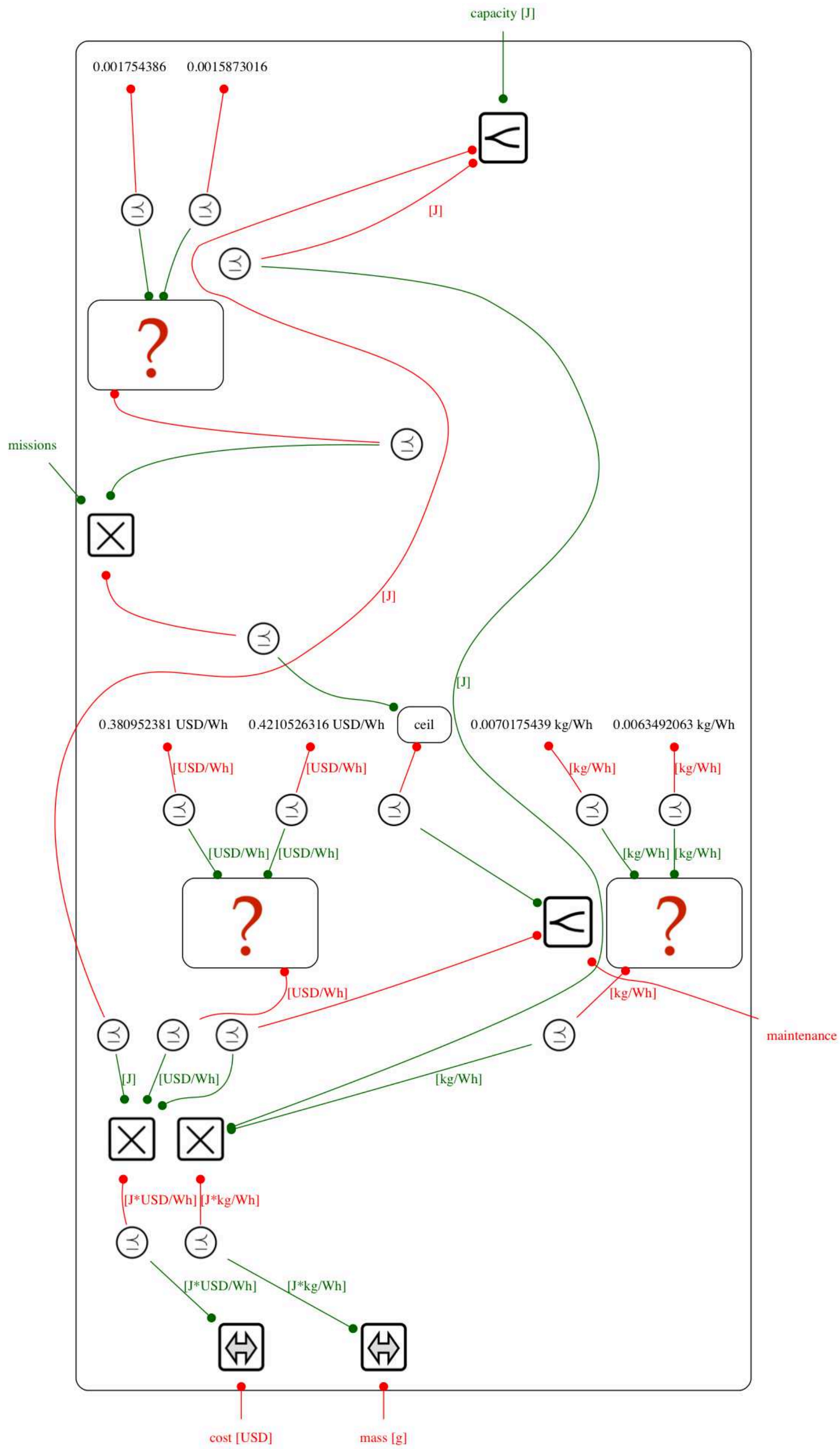
The figures have a 5% uncertainty added to them.

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.

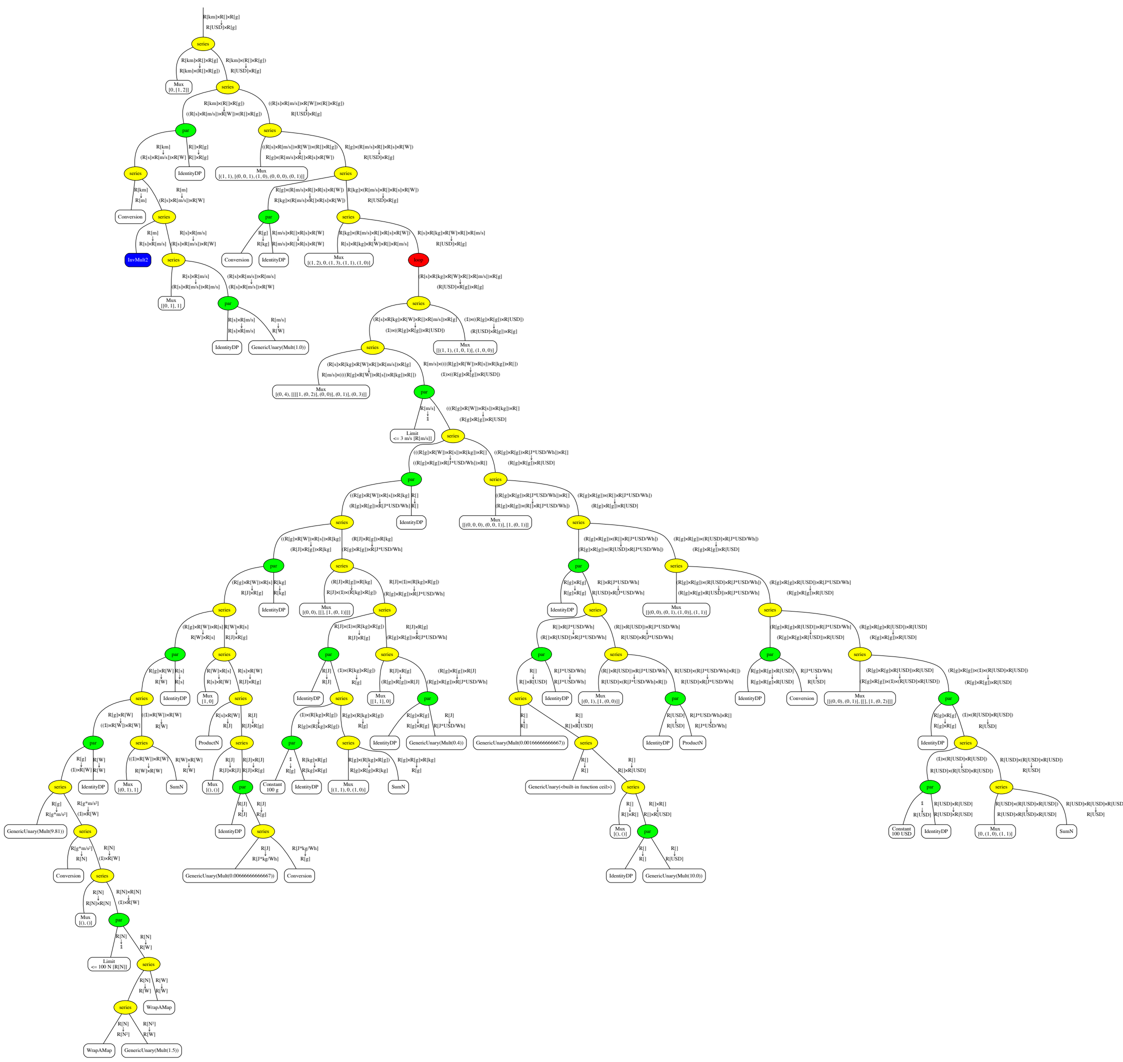
For example, the following code specializes the template using only the `Battery_LiPo` MCDP.

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