Co-Design of Embodied Intelligence: A Structured Approach

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems

G. Zardini, D. Milojevic, A. Censi, and E. Frazzoli

Institute for Dynamic Systems and Control, ETH Zürich
Automotive Powertrain Technologies Lab, EMPA

gzardini@ethz.ch - http://gioele.science

The pain of engineering complex systems

So many **components** (hardware, software, ...), so many choices to make!

Nobody can understand the **whole** thing!

We forget why we made some **choices**, and we are afraid to make **changes**...

These "computer" thingies are not helping us that much for design...

anthropomorphization of 21st century engineering malaise

"My dear, it's simple: you lack a proper theory of co-design!"

Co-design of autonomous systems: from hardware selection to control synthesis

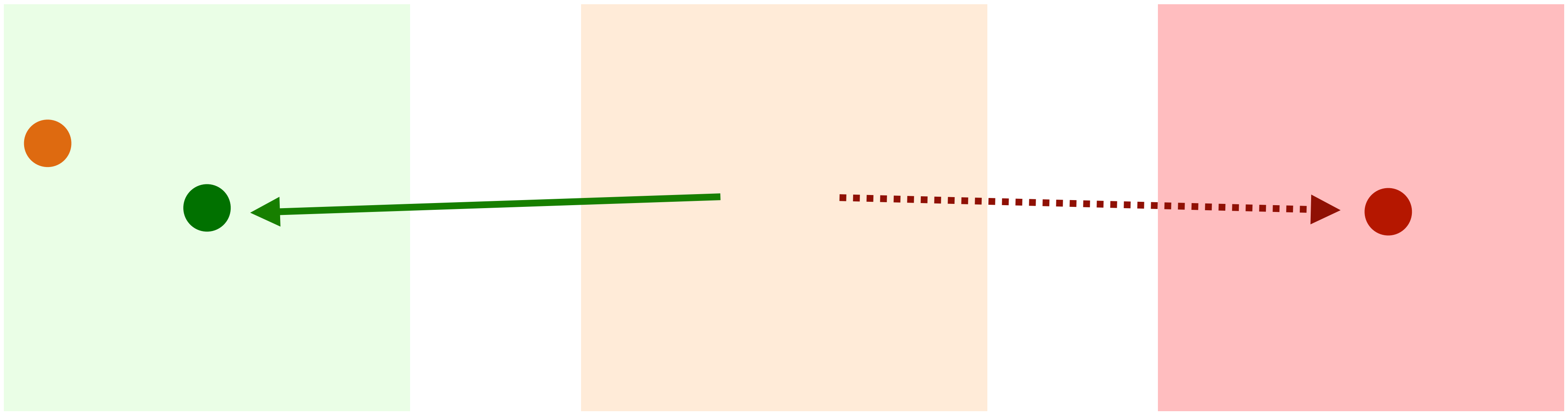
An autonomous robot =

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

- **Takeways** of this talk**:**
  - Using co-design, it is easy to **hierarchical embodied intelligence models**
  - Very **intuitive** modeling approach (no “acrobatics” needed)
  - **Rich modeling capabilities**: analytic models, catalogues, simulations
  - **Compositionality** and **modularity** allow **interdisciplinary collaboration**
  - Co-design produces **actionable information** for designers to **reason** about their problems

An abstract view of design problems

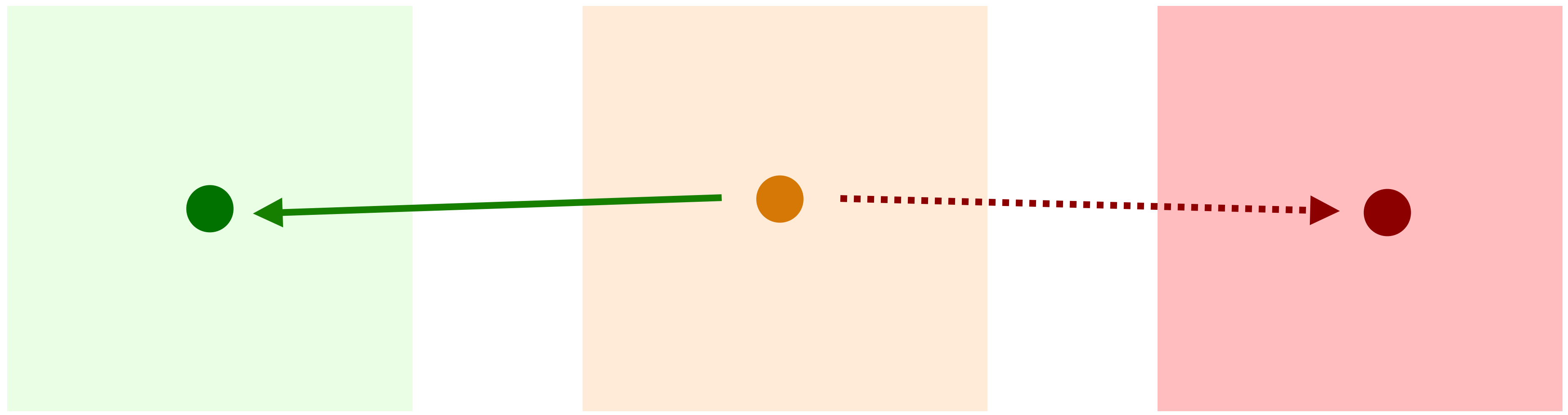
- Across fields, design or synthesis problems are defined with 3 spaces:
  - **implementation space**: the options we can choose from;
  - **functionality space**: what we need to provide/achieve;
  - **requirements/costs space**: the resources we need to have available;



functionality *(provided)*	***implementations***	***costs, resources*** *(required)*
desired behavior	*choices*	
specifications	*plans*	*requirenents*
objectives	*blueprints*	*dependencies*
guarantees	*decision variables*	
“function”	*“form”*	*“function”*
“conclusions”	*“proof”*	*“assumptions”*

An abstract view of design problems

- Across fields, design or synthesis problems are defined with 3 spaces:
  - **implementation space**: the options we can choose from;
  - **functionality space**: what we need to provide/achieve;
  - **requirements/costs space**: the resources we need to have available;



choices

Partially ordered sets

to maximize $\langle \mathbf{F}, \preceq_{\mathbf{F}} \rangle$

to minimize $\langle \mathbf{R}, \preceq_{\mathbf{R}} \rangle$

Partial orders allow to model various trade-offs

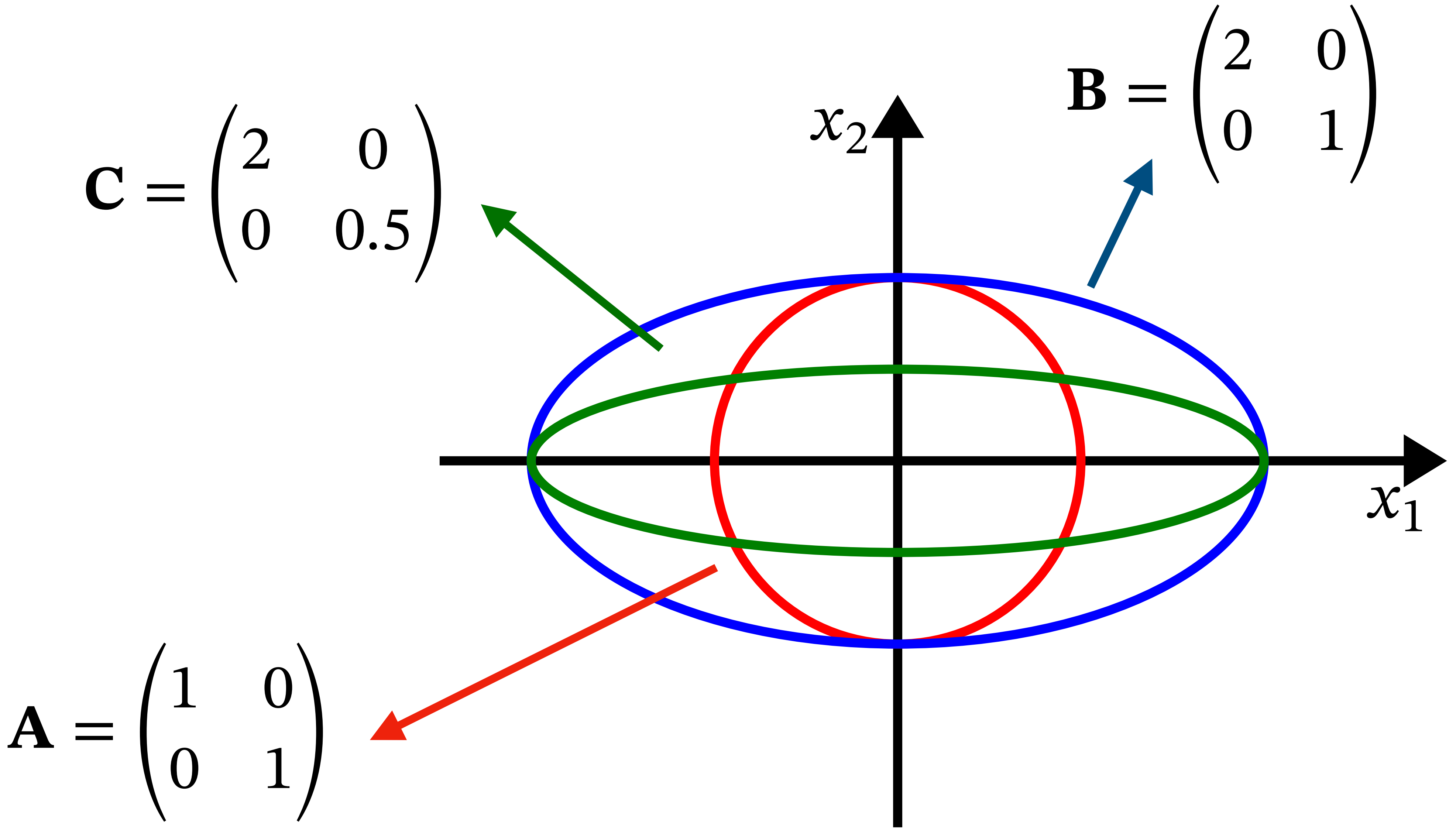
> **Definition.** A *poset* is a tuple $\langle P, \preceq_P \rangle$, where P is a set and $\preceq_P$ is a partial order, defined as a reflexive, transitive, and antisymmetric relation.

- All **totally ordered sets** are particular cases of **partially ordered sets:** $\langle \mathbb{R}_{\geq 0}, \leq \rangle$ $\langle \mathbb{N}, \leq \rangle$

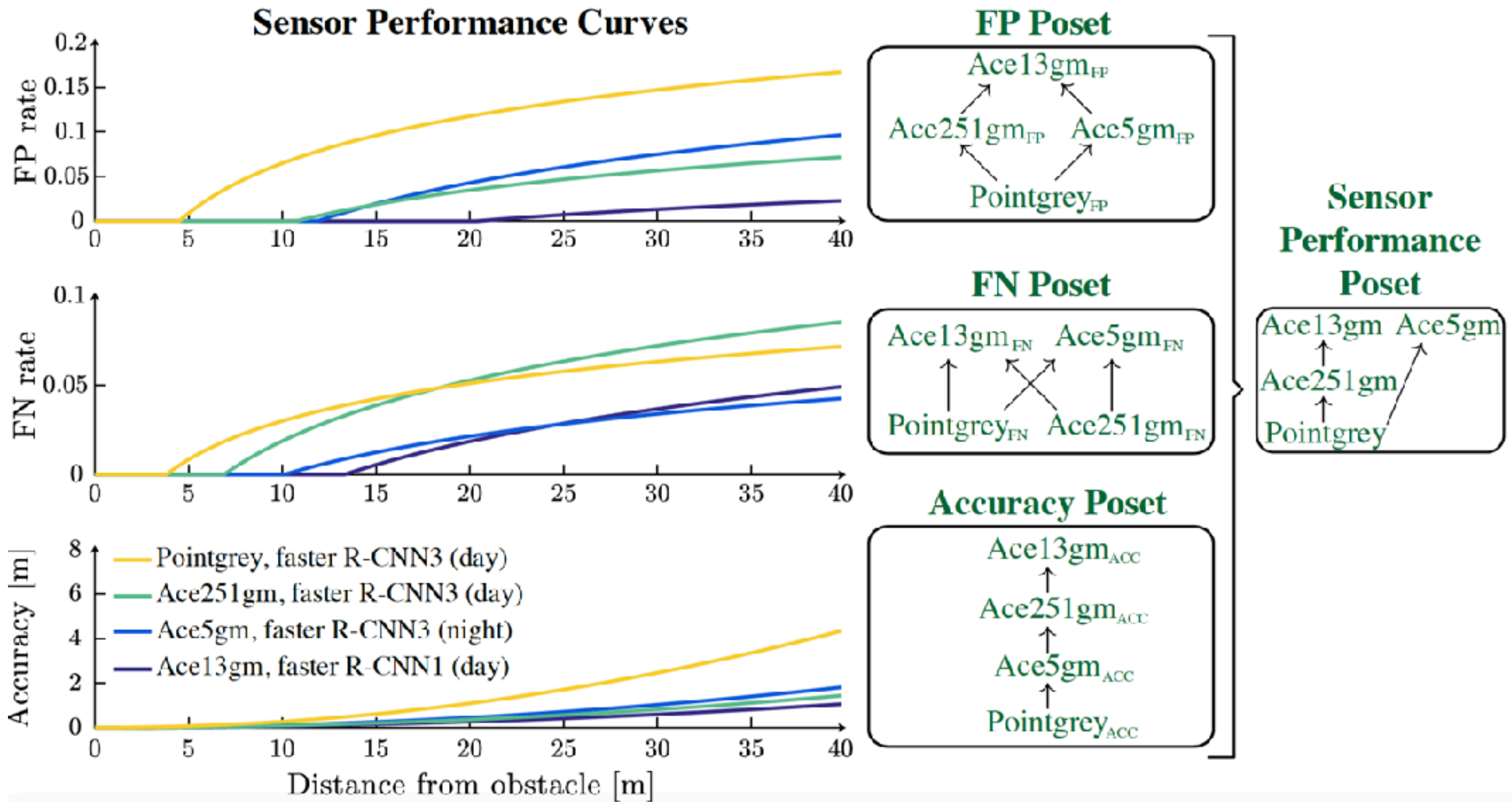
- In this work, among others, we consider

A poset of positive semi-definite matrices

$$\mathbf{A} \preceq \mathbf{B} \Leftrightarrow (\mathbf{A} - \mathbf{B}) \in \mathcal{P}^n, \quad \mathbf{A}, \mathbf{B} \in \mathcal{P}^n$$



A poset of sensors



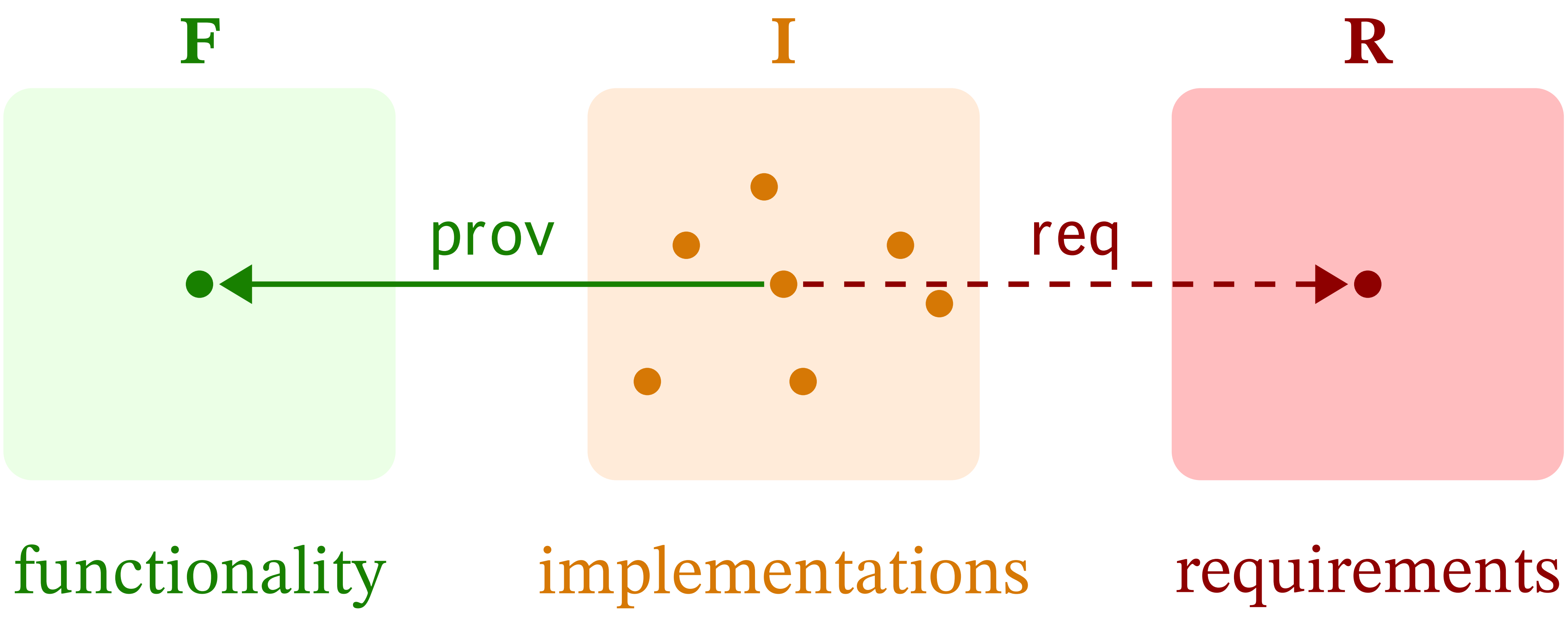
Design problem with implementation (DPIs)

Definition (Design problem with implementation). A *design problem with implementation* (DPI) is a tuple

$$\langle \mathbf{F}, \mathbf{R}, \mathbf{I}, \mathtt{prov}, \mathtt{req} \rangle,$$

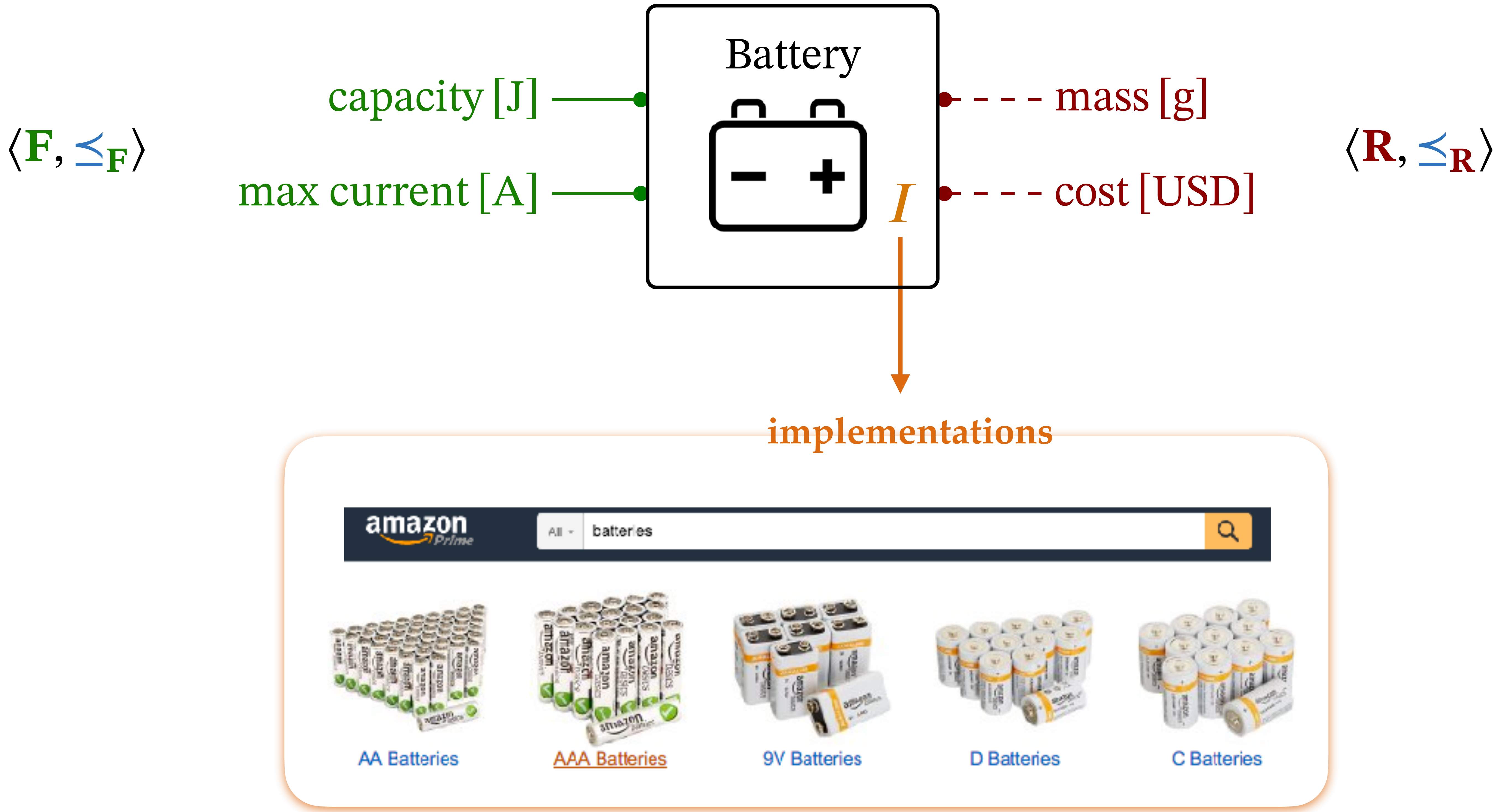
where:

- $\mathbf{F}$ is a poset, called *functionality space*;
- $\mathbf{R}$ is a poset, called *requirements space*;
- $\mathbf{I}$ is a set, called *implementation space*;
- the map $\mathtt{prov} : \mathbf{I} \to \mathbf{F}$ maps an implementation to the functionality it provides;
- the map $\mathtt{req} : \mathbf{I} \to \mathbf{R}$ maps an implementation to the resources it requires.



Graphical notation for DPIs

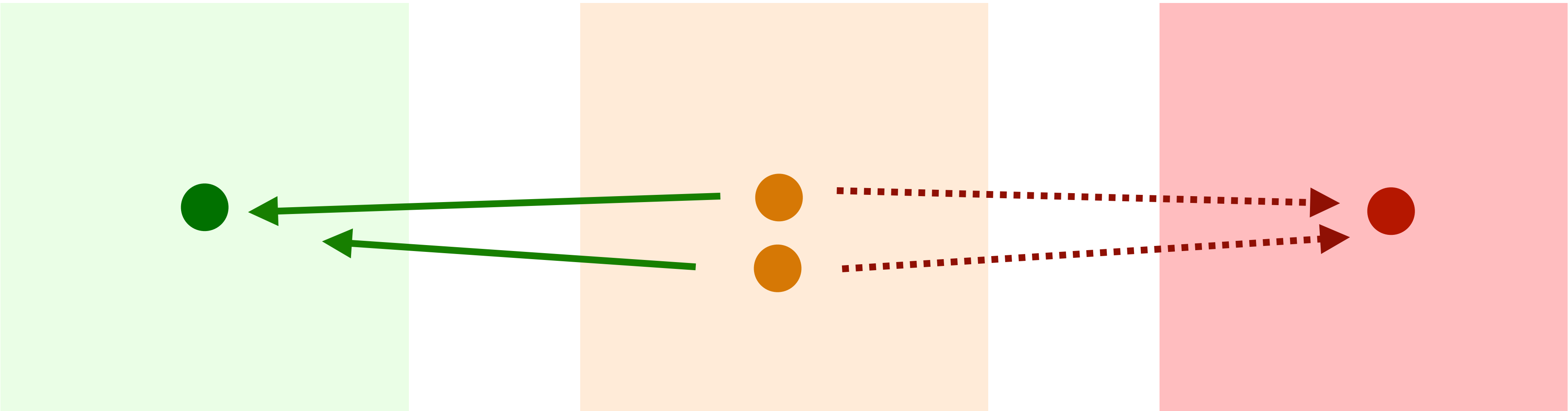
- We use this graphical notation:
  - functionality: **green continuous wires** on the left
  - requirements: **dashed red wires** on the right.



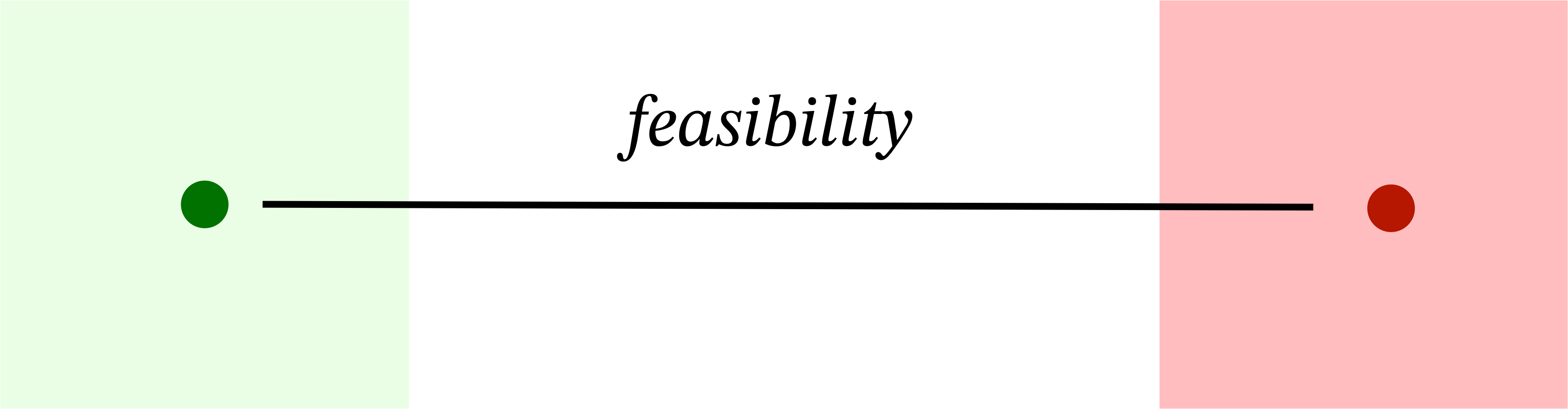
Engineering is constructive

- For the purpose of design, we **need to know how something is done**, not just that it is possible to do something: **engineering is constructive**.

- We need to know what are the implementation(s), if any, that relate functionality and costs.



- For the algorithmic solution of co-design problem, **it is useful to consider a direct feasibility relation** from functionality to costs.



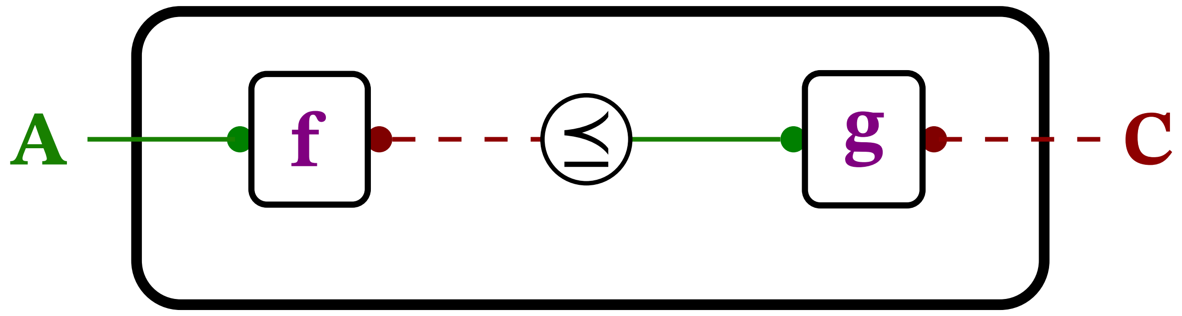
$$\mathbf{d} : \mathbf{F}^{\mathrm{op}} \times \mathbf{R} \to_{\mathbf{Pos}} \mathbf{Bool}$$

$$\langle f^*, r \rangle \mapsto \exists i \in \mathbf{I} : (f \preceq_{\mathbf{F}} \mathsf{prov}(i)) \wedge (\mathsf{req}(i) \preceq_{\mathbf{R}} r)$$

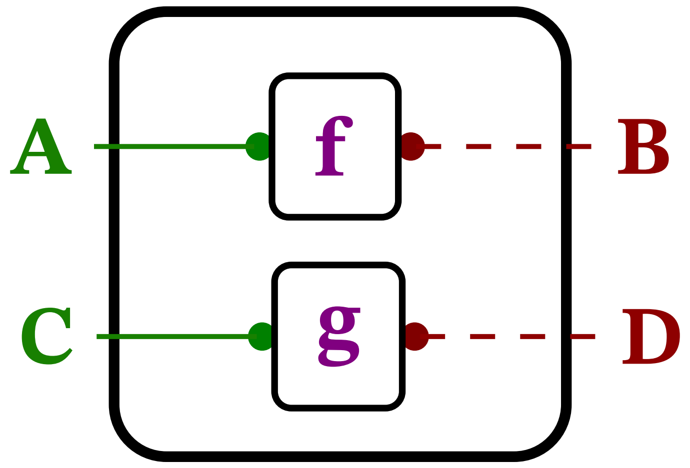
- **Monotone** map: **Lower functionalities** does **not** require **more resources**, **higher resources** do not provide **less functionalities**

Composition operators

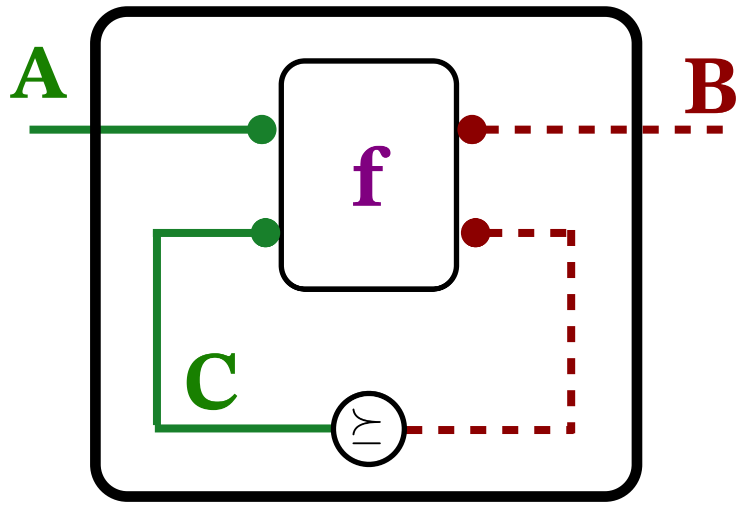
"series"



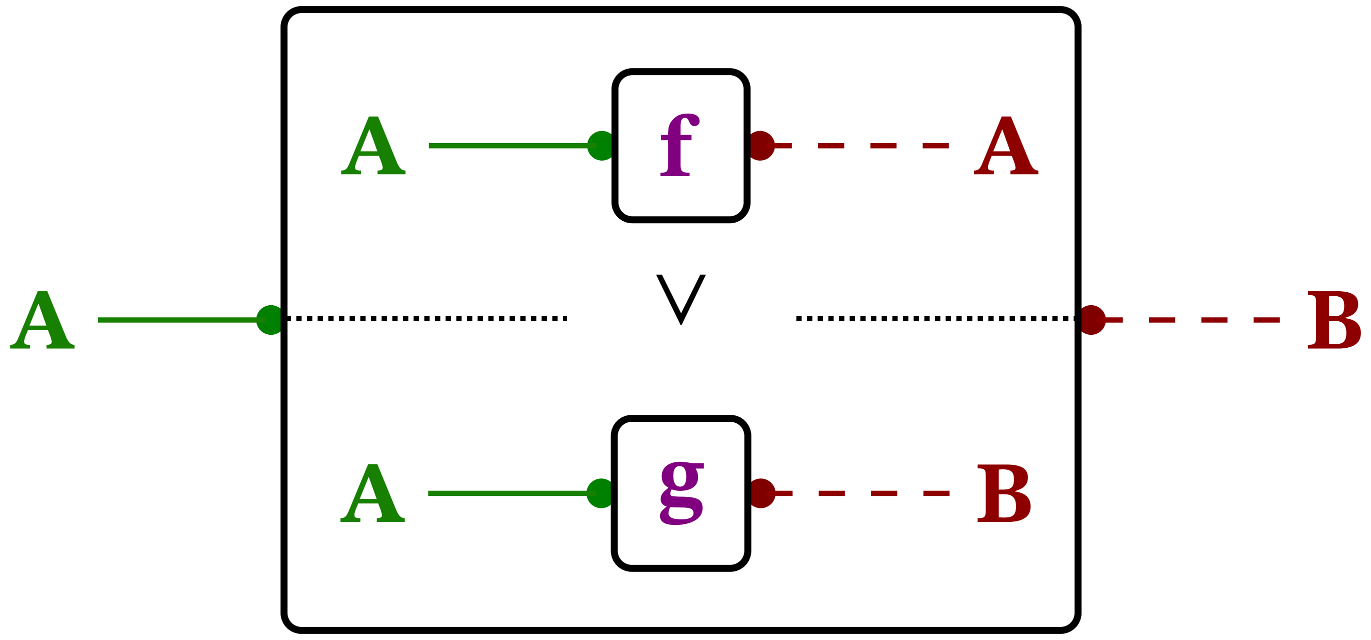
"parallel"



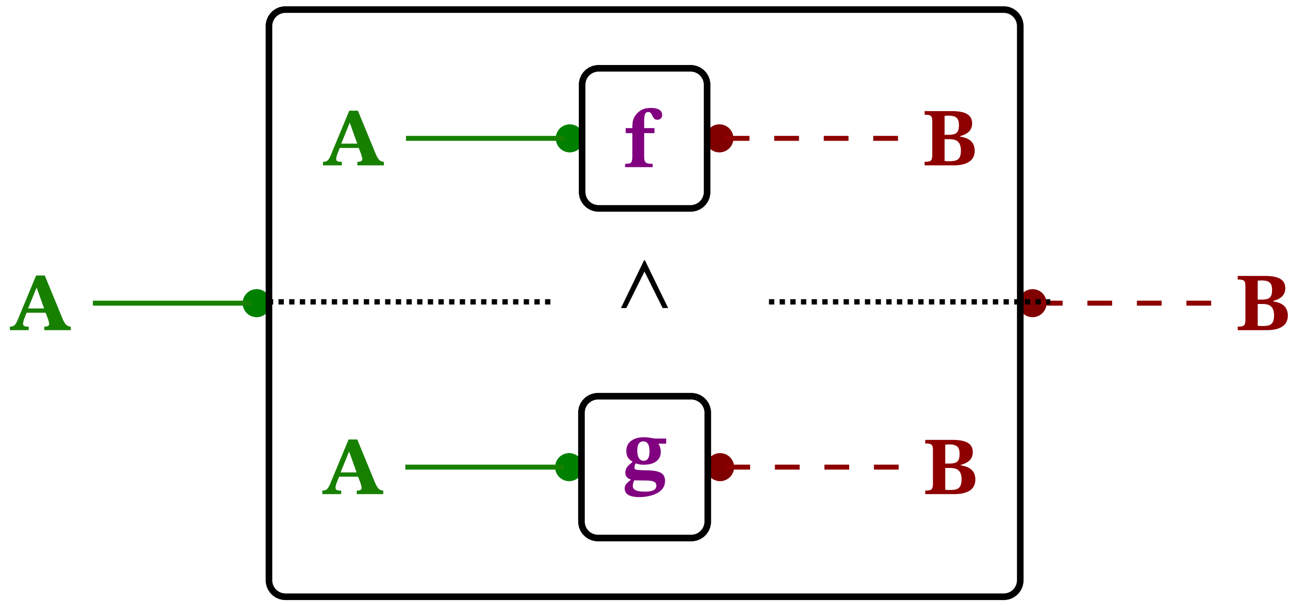
"feedback"



"choose between two options"



"convince two experts"



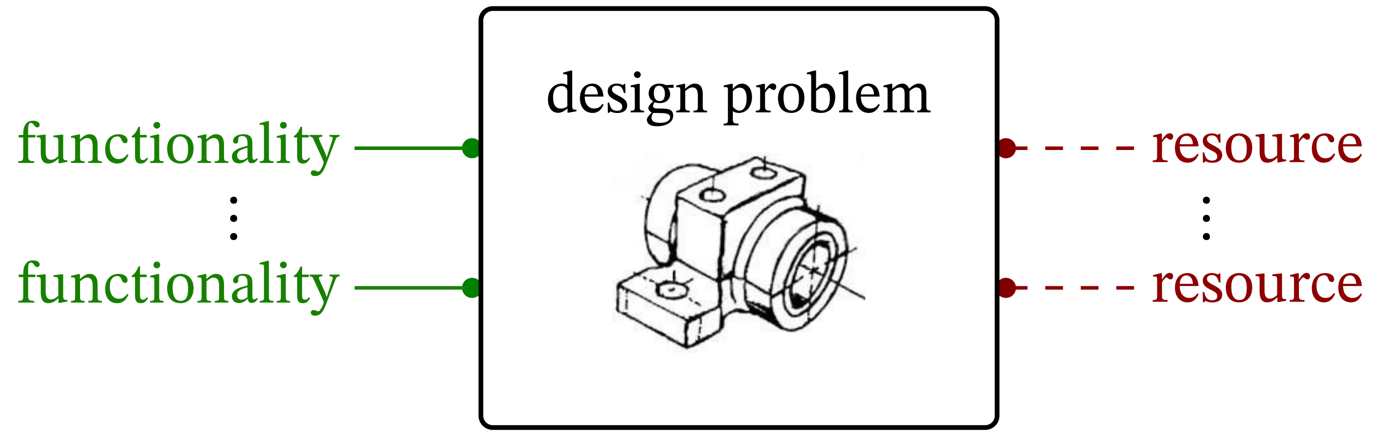
- The **composition** of any two **DPs** returns a **DP** (closure)
- Very practical tool to **decompose** large **problems** into **subproblems**

Design queries

- Two basic design queries are:
  - **FixFunMinReq**: Fixed a lower bound on functionality, minimize the resources.
  - **FixReqMaxFun**: Fixed an upper bound on the resource, maximize the functionality

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

FixFunMinReq →



← **FixReqMaxFun**

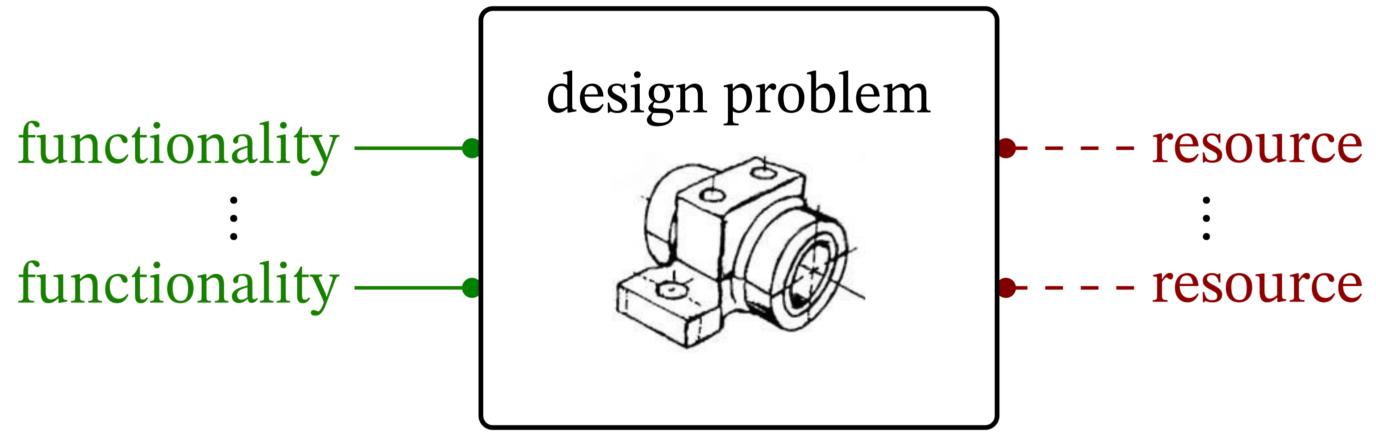
Given the resources that are available, what is
the **maximal functionality** that can be provided?

Design queries

- Two basic design queries are:
  - **FixFunMinReq**: Fixed a lower bound on functionality, minimize the resources.
  - **FixReqMaxFun**: Fixed an upper bound on the resource, maximize the functionality

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

FixFunMinReq



FixReqMaxFun

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

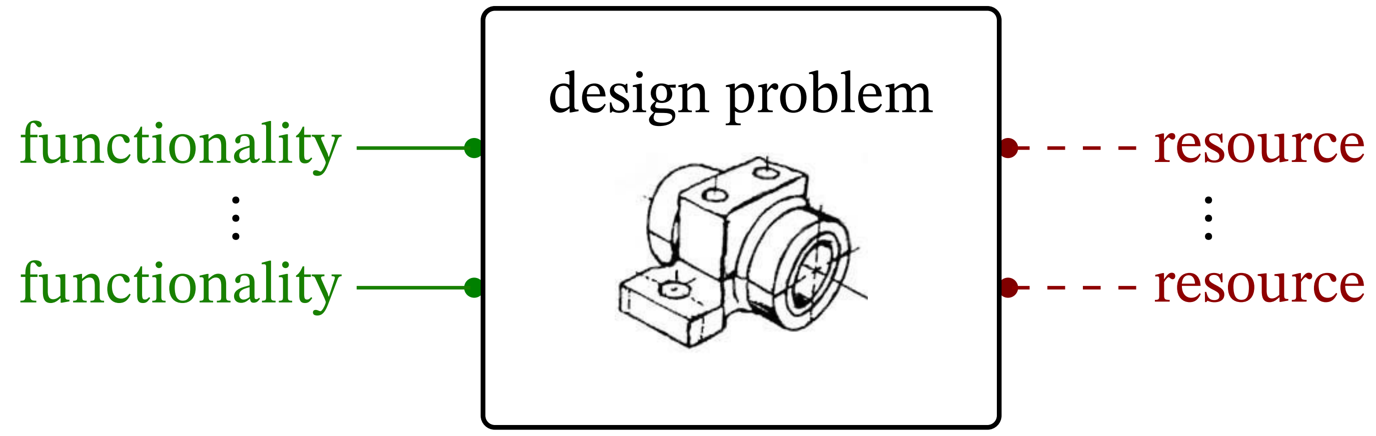
- The two problems are **dual**
- From the solutions, one can retrieve the **implementations** (design choices)

Design queries

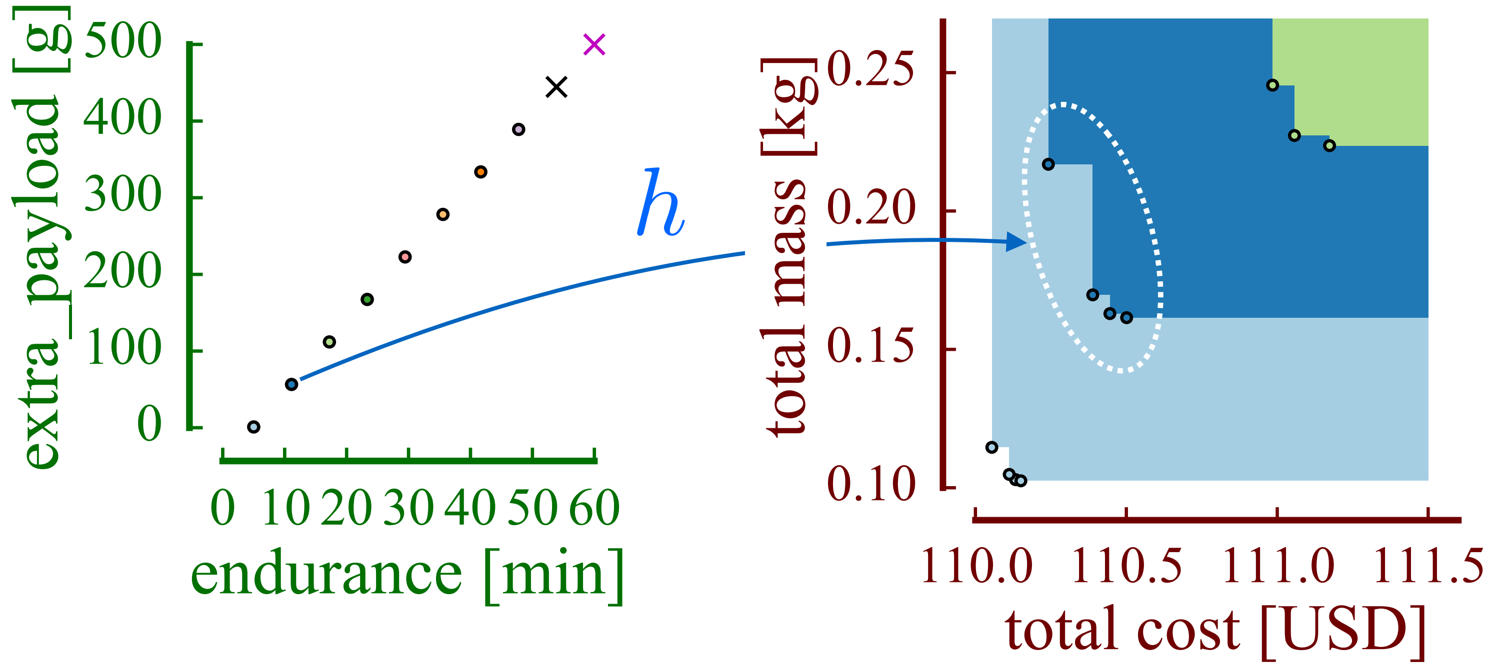
- Two basic design queries are:
  - **FixFunMinReq**: Fixed a lower bound on functionality, minimize the resources.
  - **FixReqMaxFun**: Fixed an upper bound on the resource, maximize the functionality

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

FixFunMinReq →



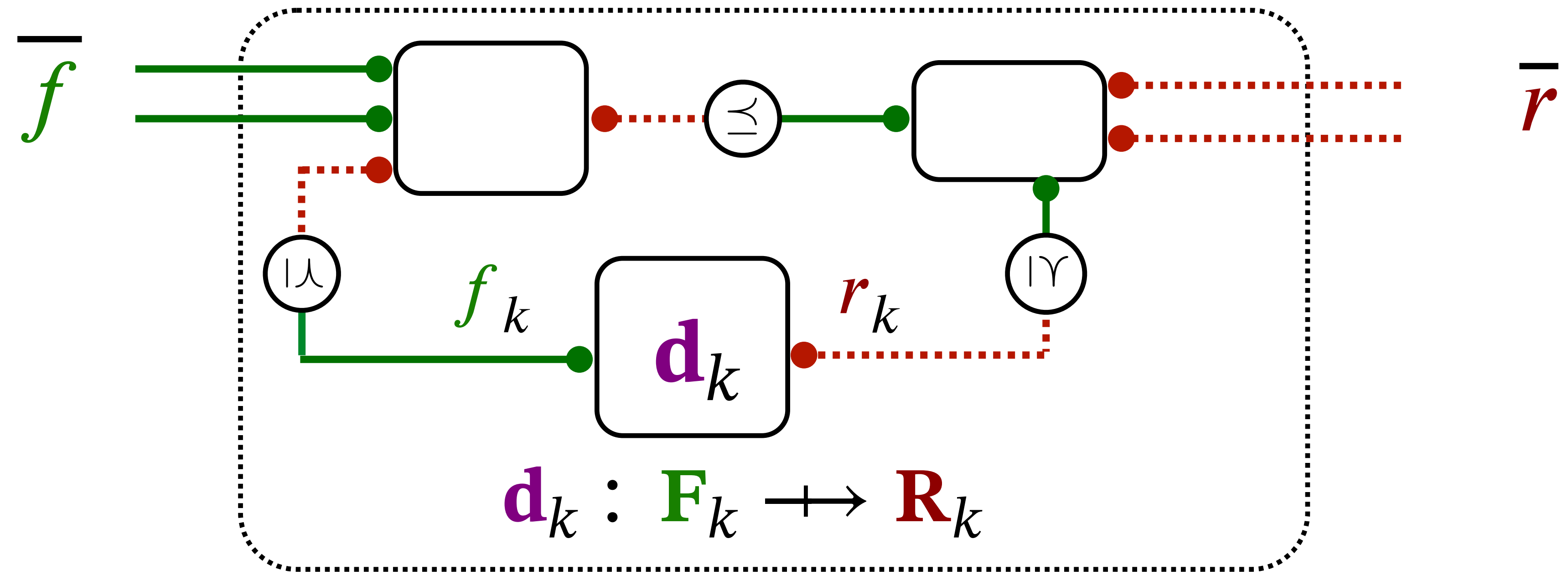
- We are looking for:
  - A map from functionality to upper sets of feasible resources: $h : \mathbf{F} \to \mathcal{U}\mathbf{R}$
  - A map from functionality to antichains of minimal resources: $h : \mathbf{F} \to \mathcal{A}\mathbf{R}$



Optimization semantics

- This is the semantics of **FixFunMinReq** as a **family of optimization problems.**

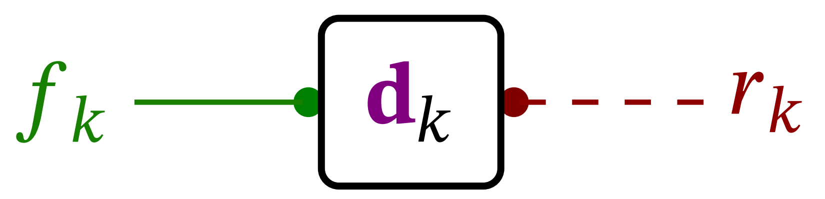
chosen by user

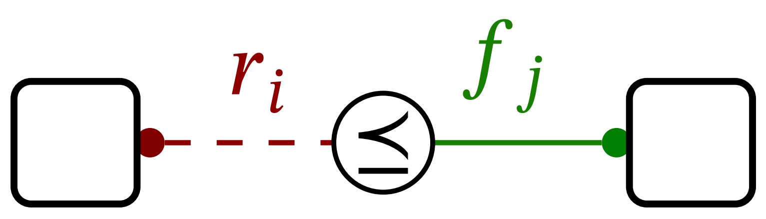


to minimize

variables $r_k \in \langle \mathbf{R}_k, \preceq_{\mathbf{R}_k} \rangle$ $\quad f_k \in \langle \mathbf{F}_k, \preceq_{\mathbf{F}_k} \rangle$

constraints *for* ***each node:*** *for* ***each edge:***





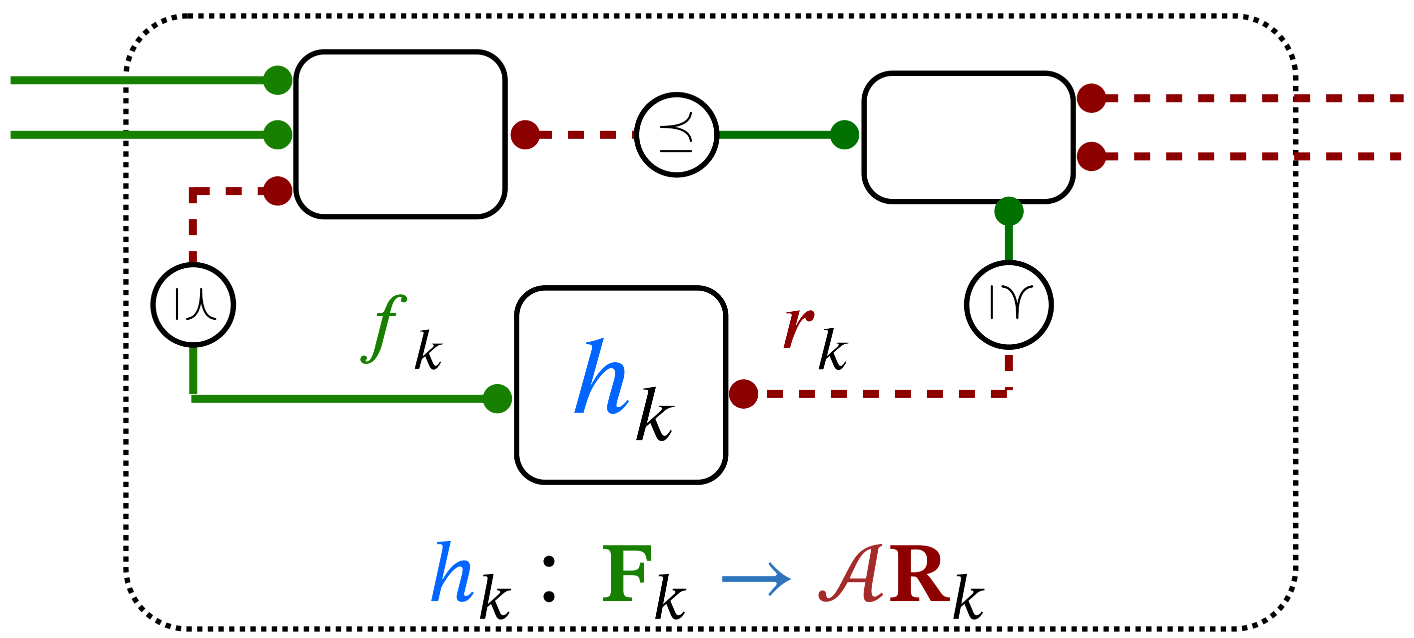
$$\mathbf{d}_k(f_k^*, r_k) = \top \qquad r_i \preceq f_j$$

objective $\underset{\preceq}{\mathrm{Min}}\, \overline{r}$

- ! not convex
- ! not differentiable
- ! not continuous
- ! not even defined on continuous spaces

Solving DP queries

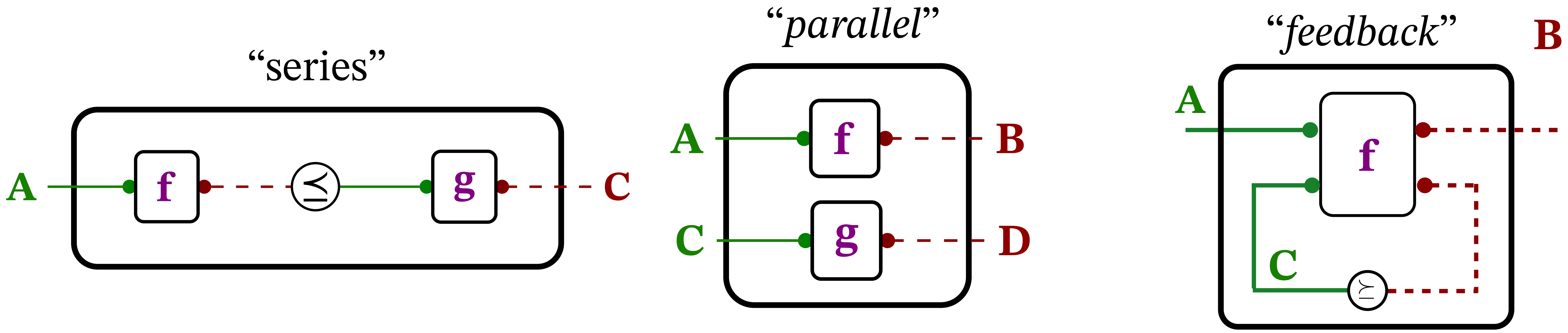
- Suppose we are given the function $h_k : \mathbf{F}_k \to \mathcal{A}\mathbf{R}_k$ for all nodes in the co-design graph.



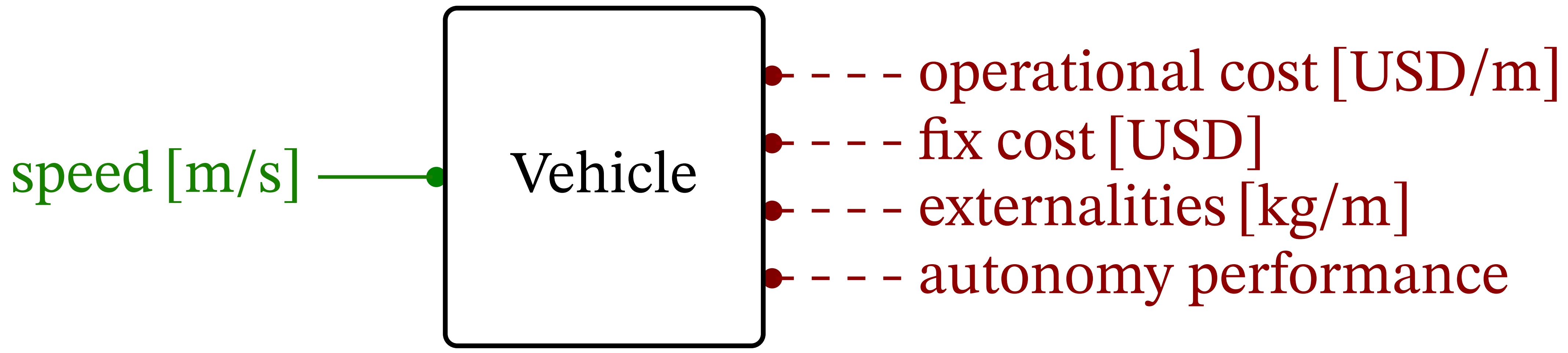
- Can we find the map $h : \mathbf{F} \to \mathcal{A}\mathbf{R}$ for the entire diagram?

- **Recursive approach:** We just need to work out the the composition formulas for all operations we have defined

- The set of **minimal** feasible resources can be obtained as the **least fixed point** of a monotone function in the space of anti-chains.



Co-design of embodied intelligence



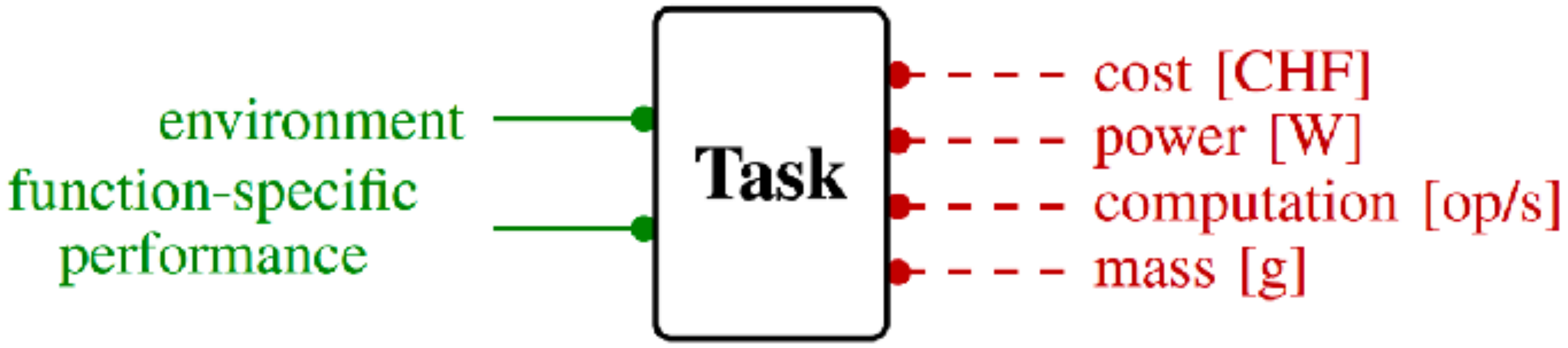
- We propose a **structured** approach to **model** and **solve embodied intelligence** co-design problems
- We take the proxy of **AV design**, from the perspective of the **developers:**
  - The methodology can be applied to other autonomous systems
  - *Proof of concept* implementation

- **Modeling approach:**
  - **Task -** *what do we need to do?*
  - **Functional decomposition -** *how to decompose the system?*
  - **Find components -** *decompose until you find components* (hardware and software)
  - **Find common resources** - In robotics, **size**, **weight**, **power**, **computation**, **latency**

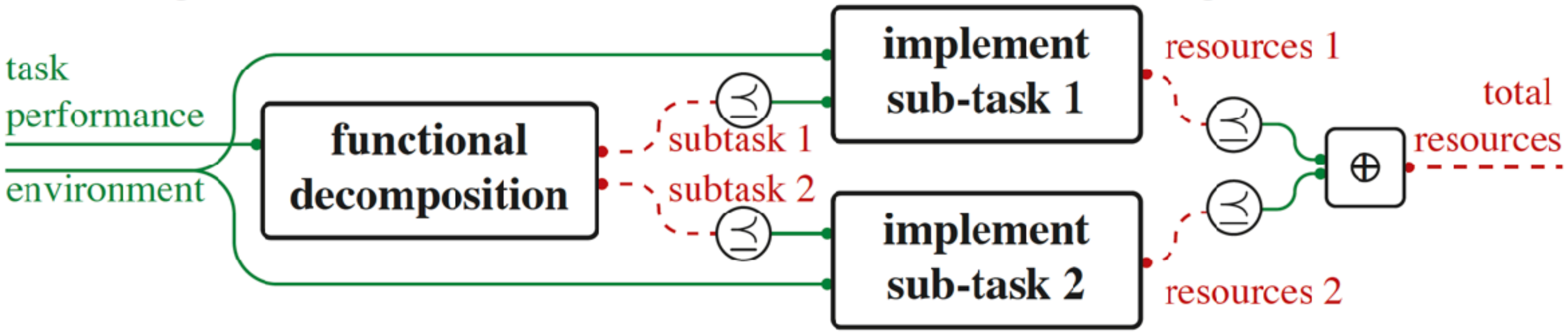
- **Implementation:**
  - **Skeleton** - *write the structure using the formal language and the found dependencies*
  - **Fill-in the holes** - *catalogues, analytic models, simulations*

Task abstraction and functional decomposition in autonomy

- Embodied intelligence tasks can be usually characterized as a **design problem:**

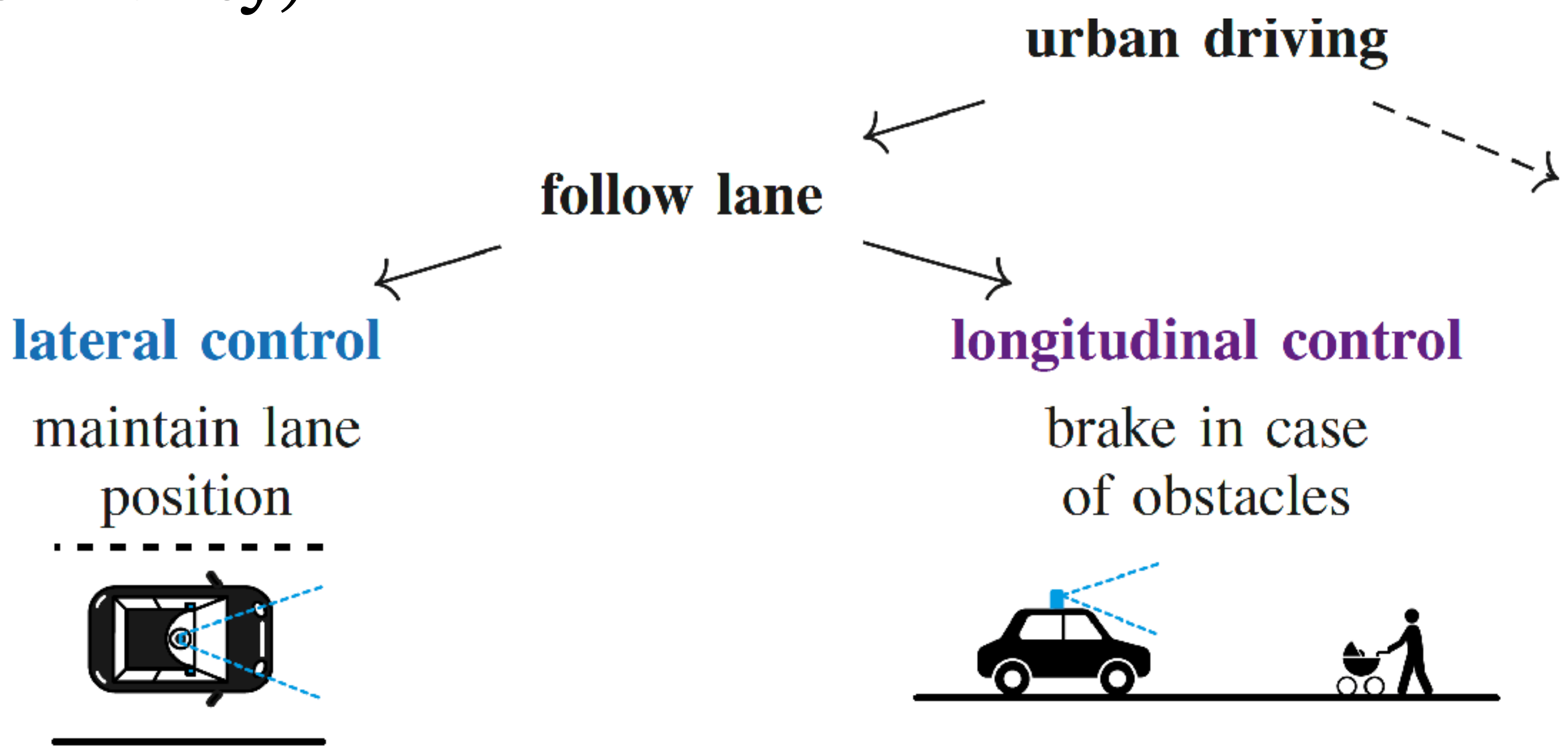


- Given **sub-tasks**, one can interconnect them:



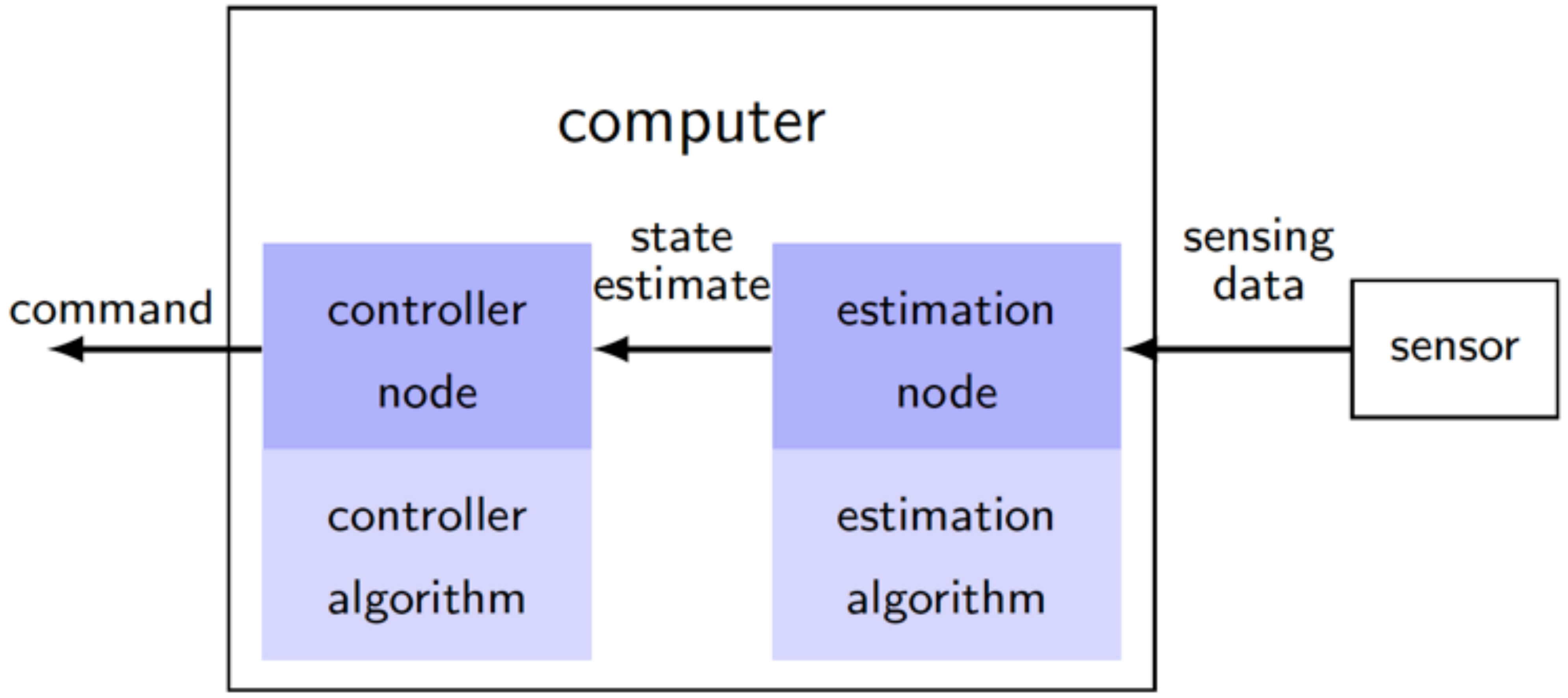
- Note that **composing** tasks returns a **task** (**compositionality**)

- For instance, in **urban driving**:



Finding components: Data flow vs. Logical dependencies

- In robotics, we are used to think about **data flow:**



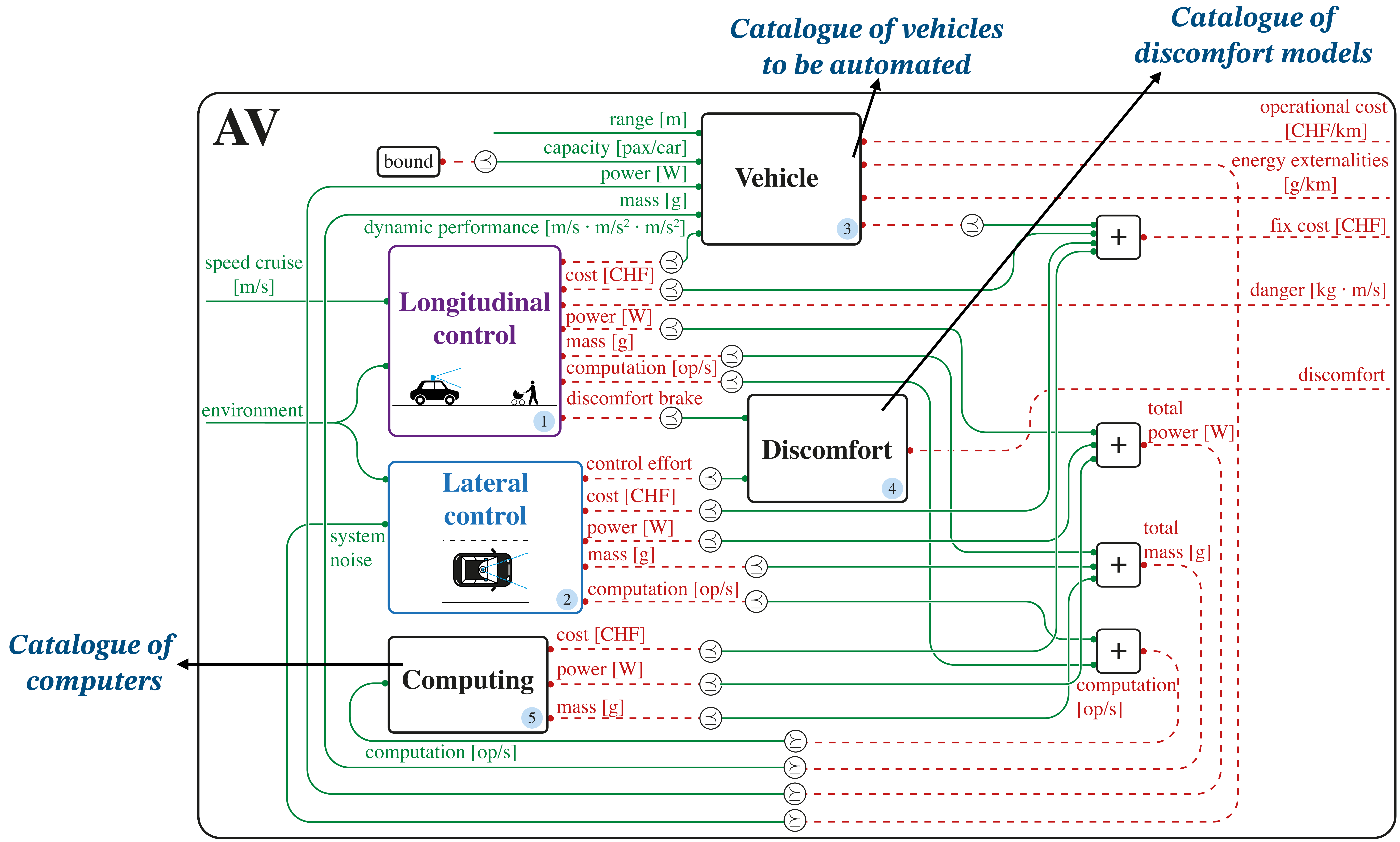
- To find **components**, it helps to reason about **logical dependencies**:

Decision making $\xrightarrow{\text{requires}}$ *State estimation* $\xrightarrow{\text{requires}}$ *Sensing data* $\xrightarrow{\text{requires}}$ *Sensor*

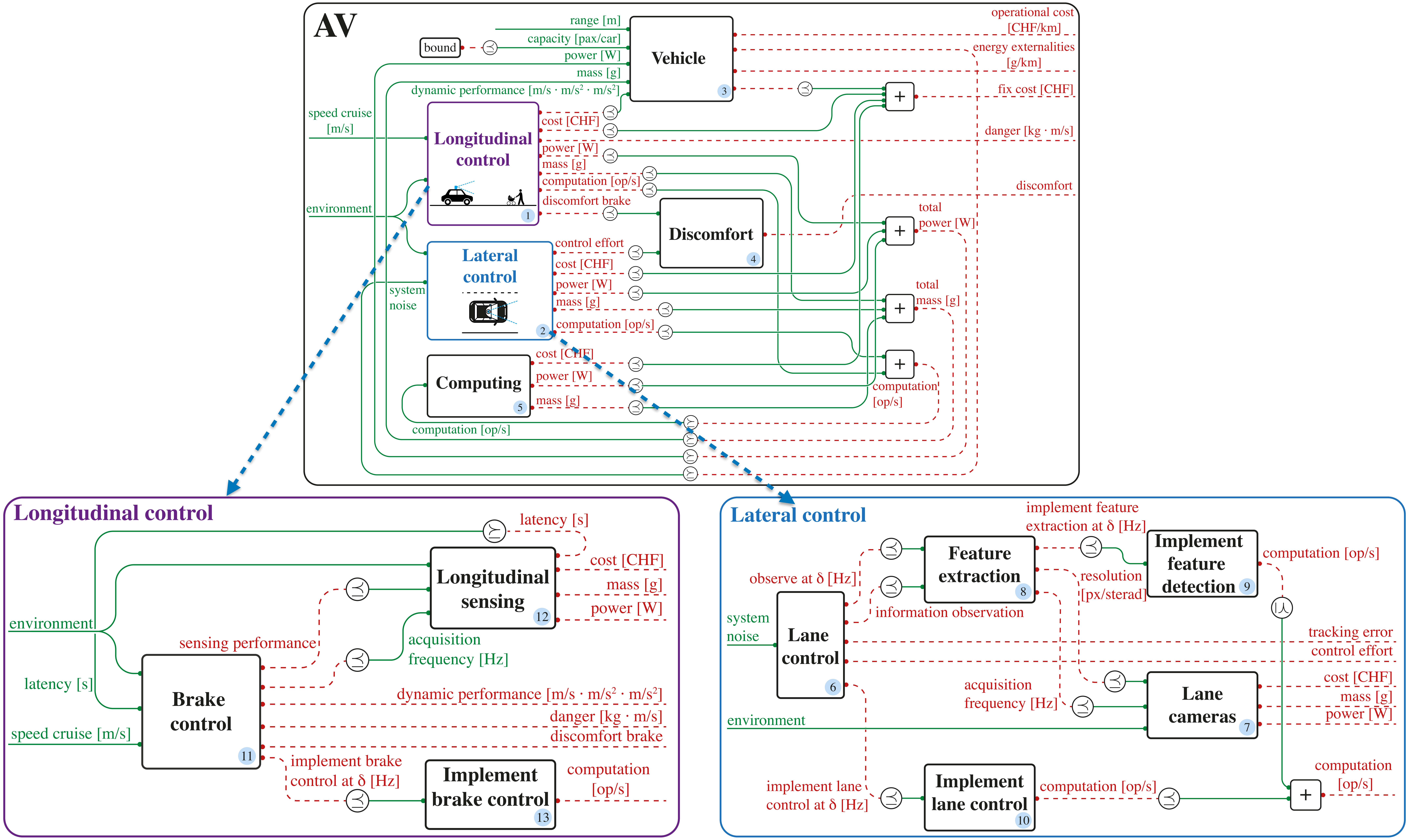
State estimation $\xrightarrow{\text{requires}}$ *algorithm*

State estimation $\xrightarrow{\text{requires}}$ *computation* $\xrightarrow{\text{requires}}$ *Computer*

Co-design of an autonomous vehicle

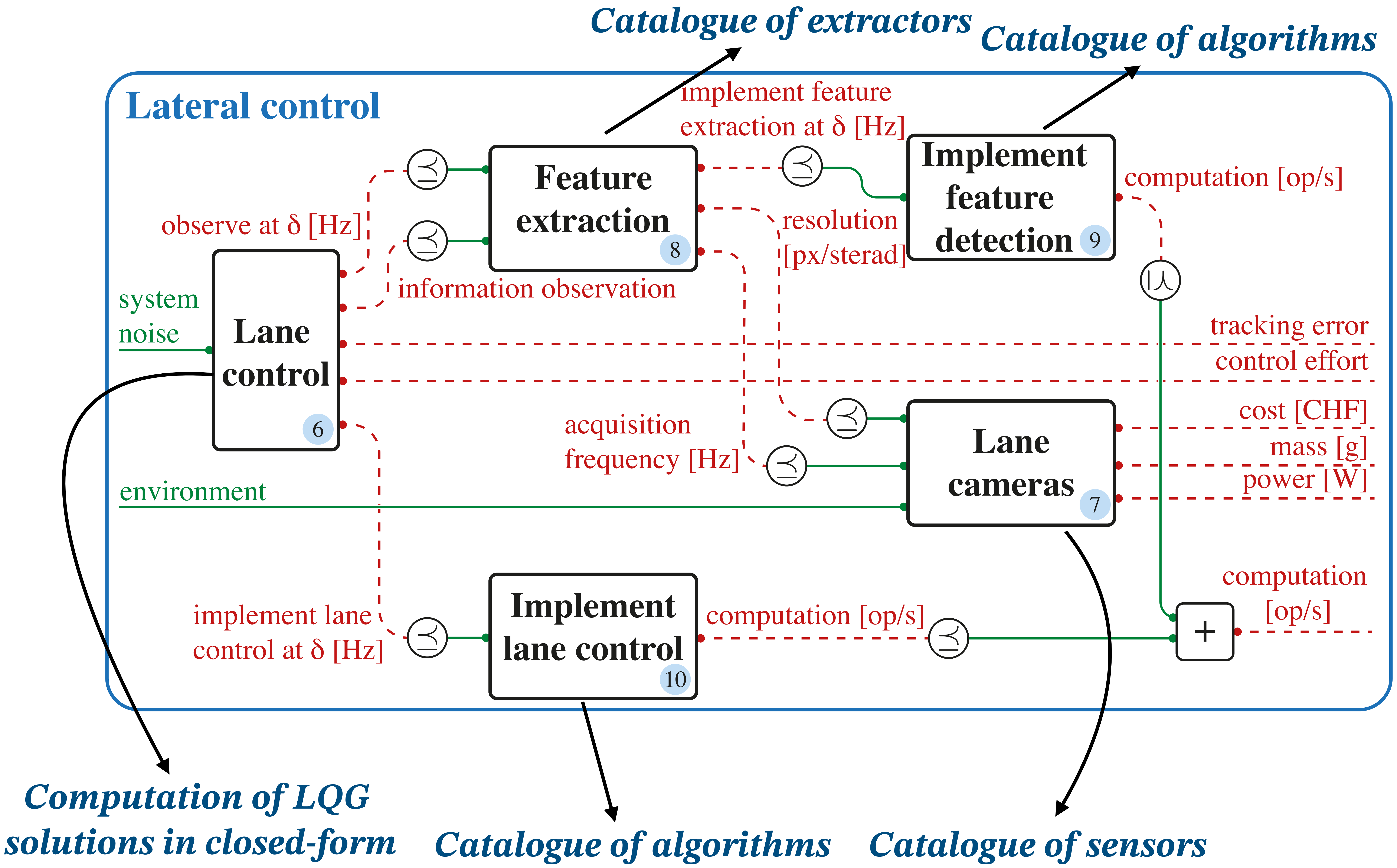


Encapsulating co-design models via functional decomposition



Co-design of lateral control: Closed-form simulations

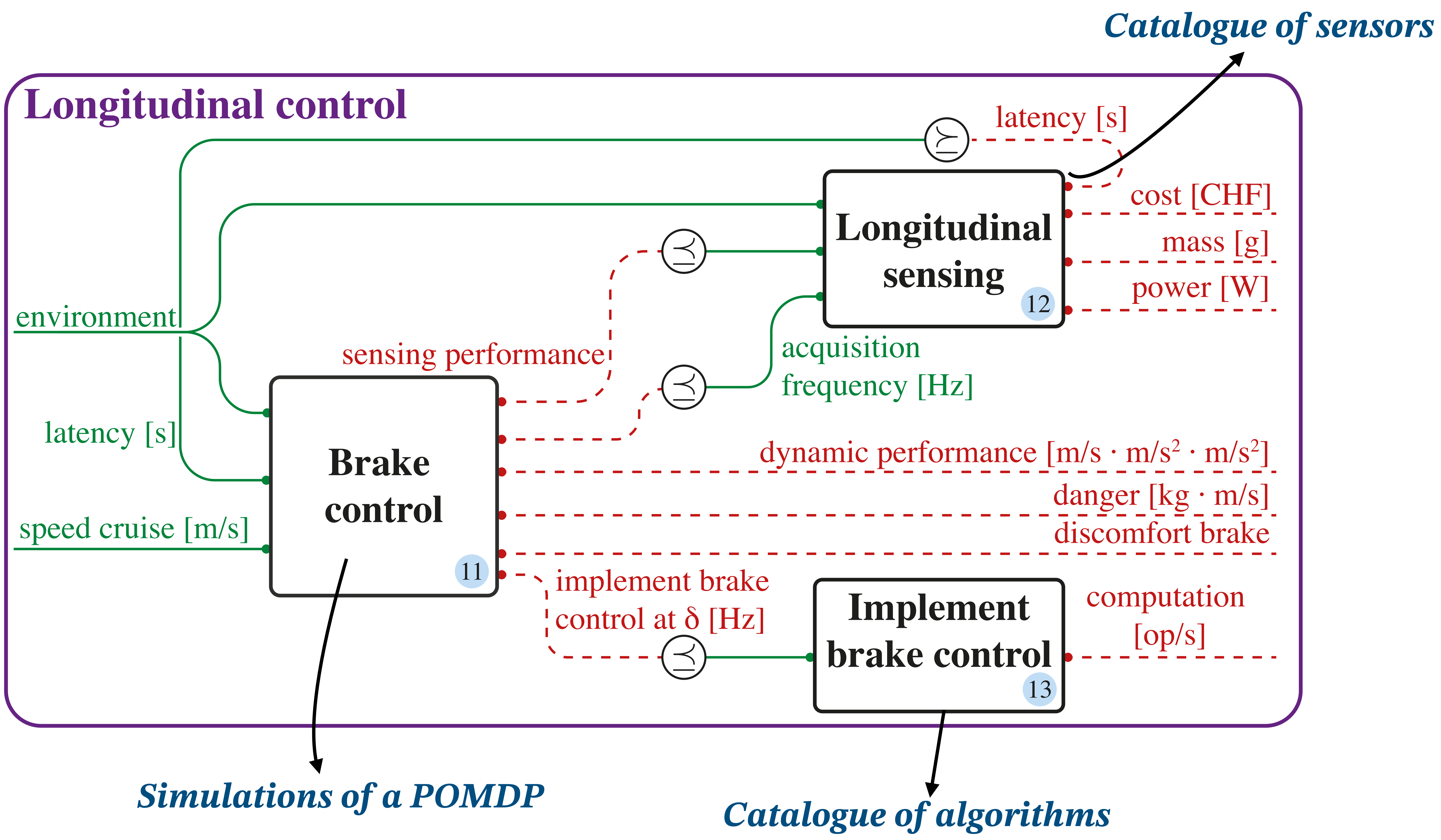
- Lateral control itself can decomposed in **sub-tasks:**



Zardini, Censi, Frazzoli, Co-design of Autonomous Systems: From Hardware Selection to Control Synthesis, *ECC 2021*

Co-design of longitudinal control: Simulations of POMDPs

- Longitudinal control can be decomposed in **sub-tasks**:



User friendly interface to solve complex optimization problems

- The theory comes with a **formal language** and a **solver (MCDP)**
- Very intuitive to use:

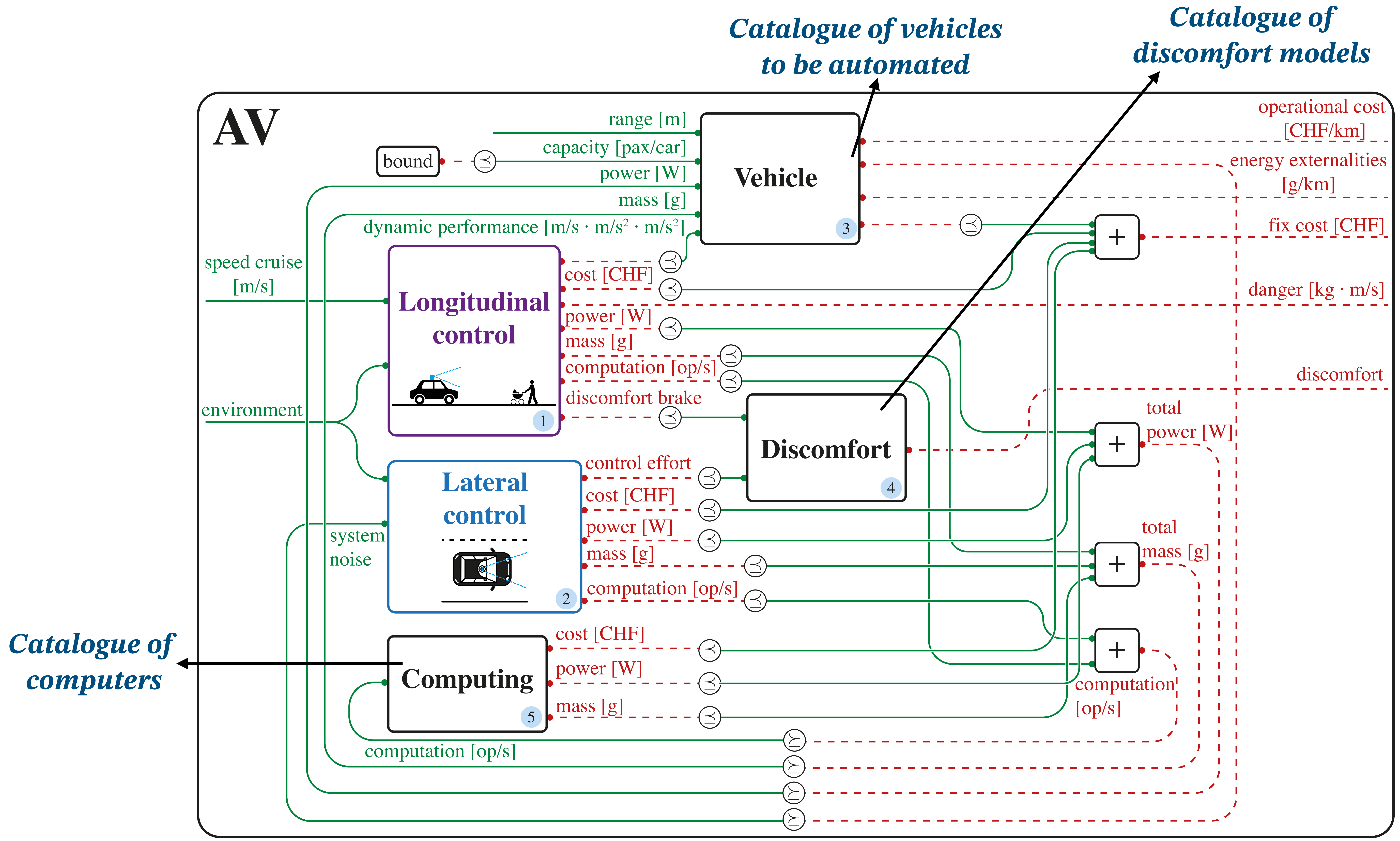
```
mcdp {
    provides computation [op/s]
    requires cost [CHF]
    requires mass [g]
    requires power [W]

}
```

```
choose(
        SedanS: (load Car_SedanS),
        SedanM: (load Car_SedanM),
        SedanL: (load Car_SedanL),
        SUVS: (load Car_SuvS),
        SUVM: (load Car_SuvM),
        Minivan: (load Car_Minivan),
        Shuttle: (load Car_Shuttle),
        Hybrid: (load Car_Hybrid),
        BEV: (load Car_BEV)
)
```

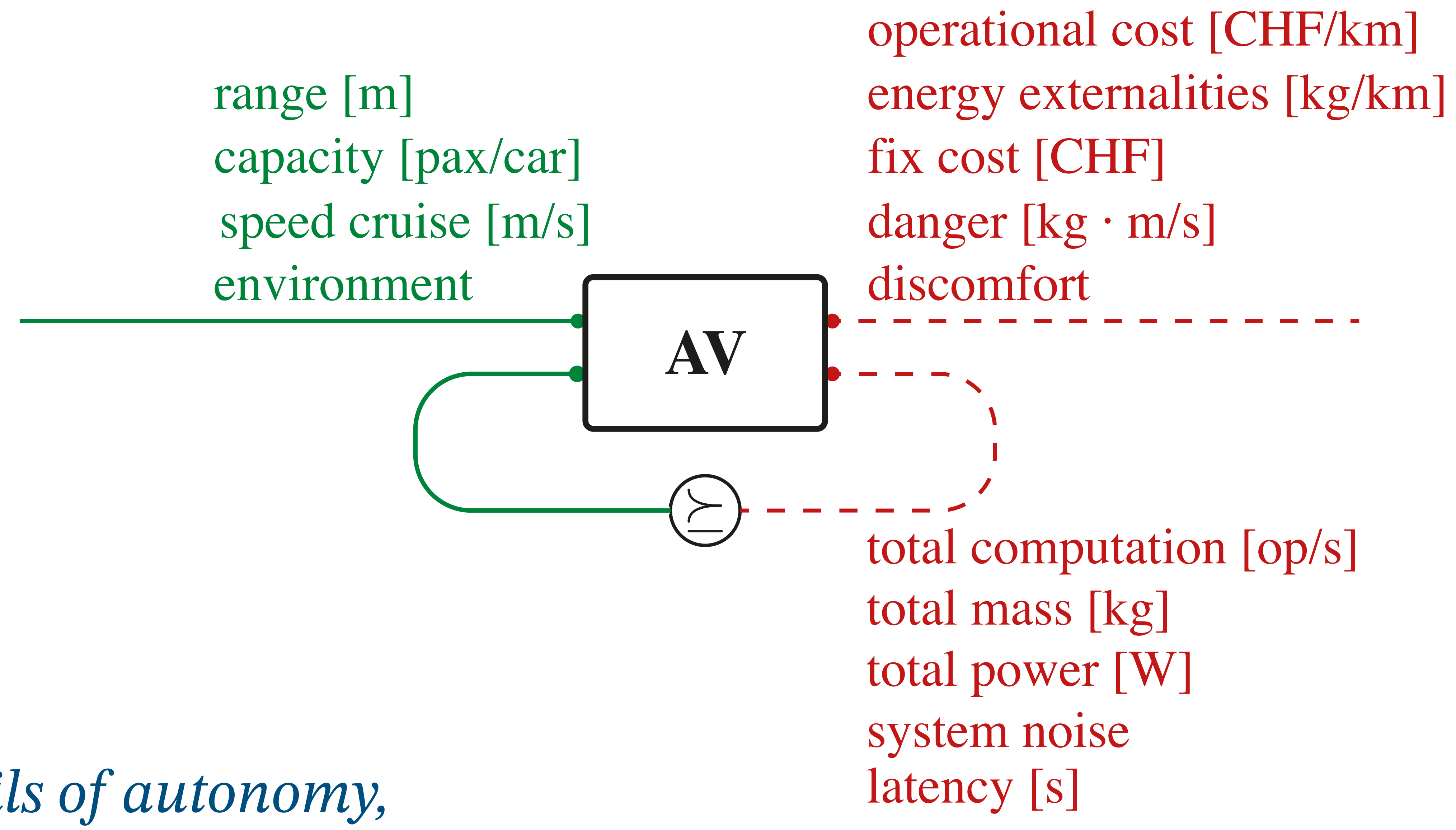
Choose query type:

- Fixed the functionality, minimize the resources.
- Fixed the resources, maximize the functionality.
- Given an implementation, evaluate functionality/resources. [UI not implemented]
- Given min functionality and max resources, determine if there is a feasible implementation. [UI not implemented]
- Given min functionality and max resources, find a feasible implementation. [UI not implemented]
- "Solve for X": find the minimal component that makes the co-design problem feasible. [UI not implemented]

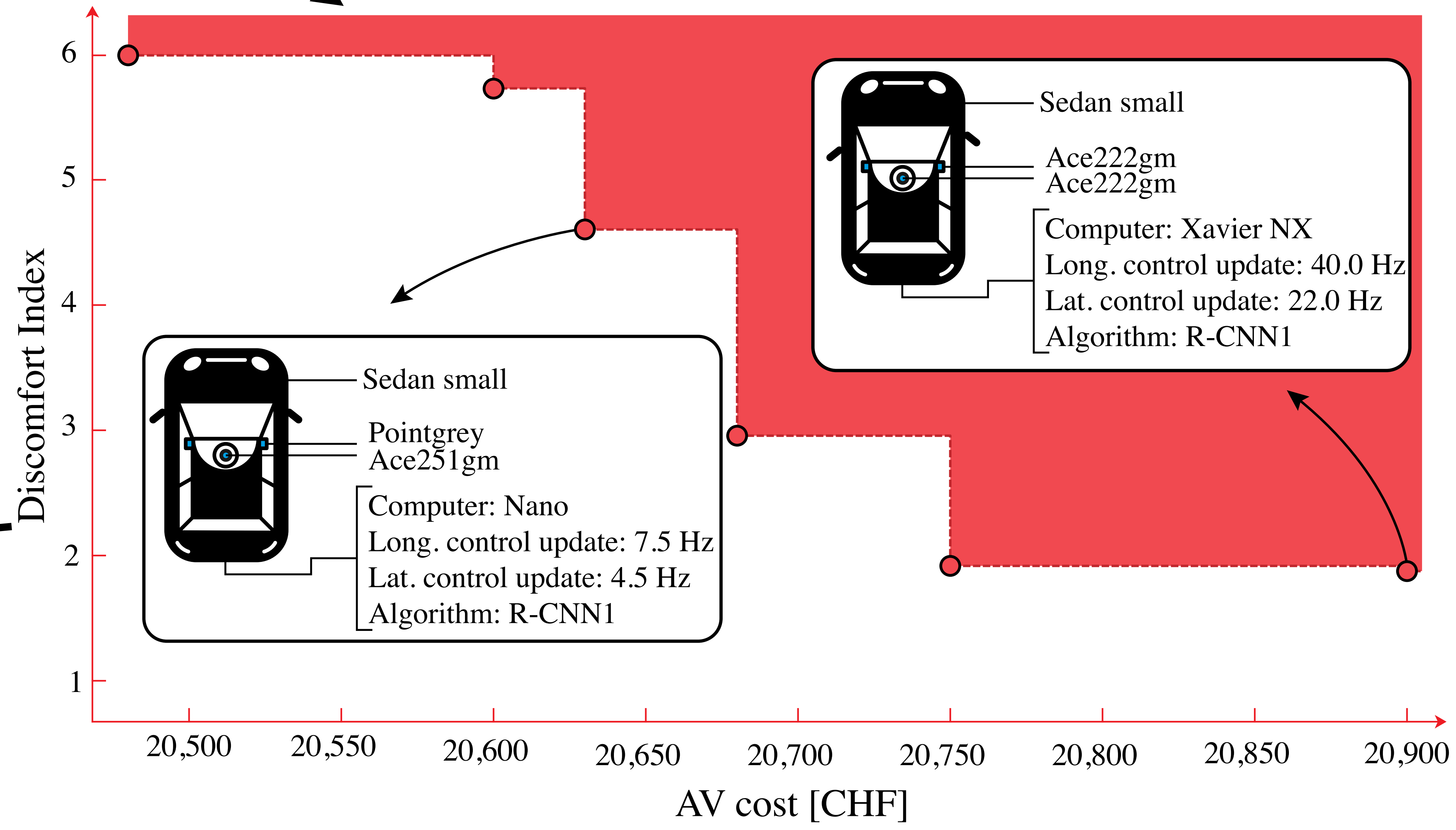
Co-design of an autonomous vehicle



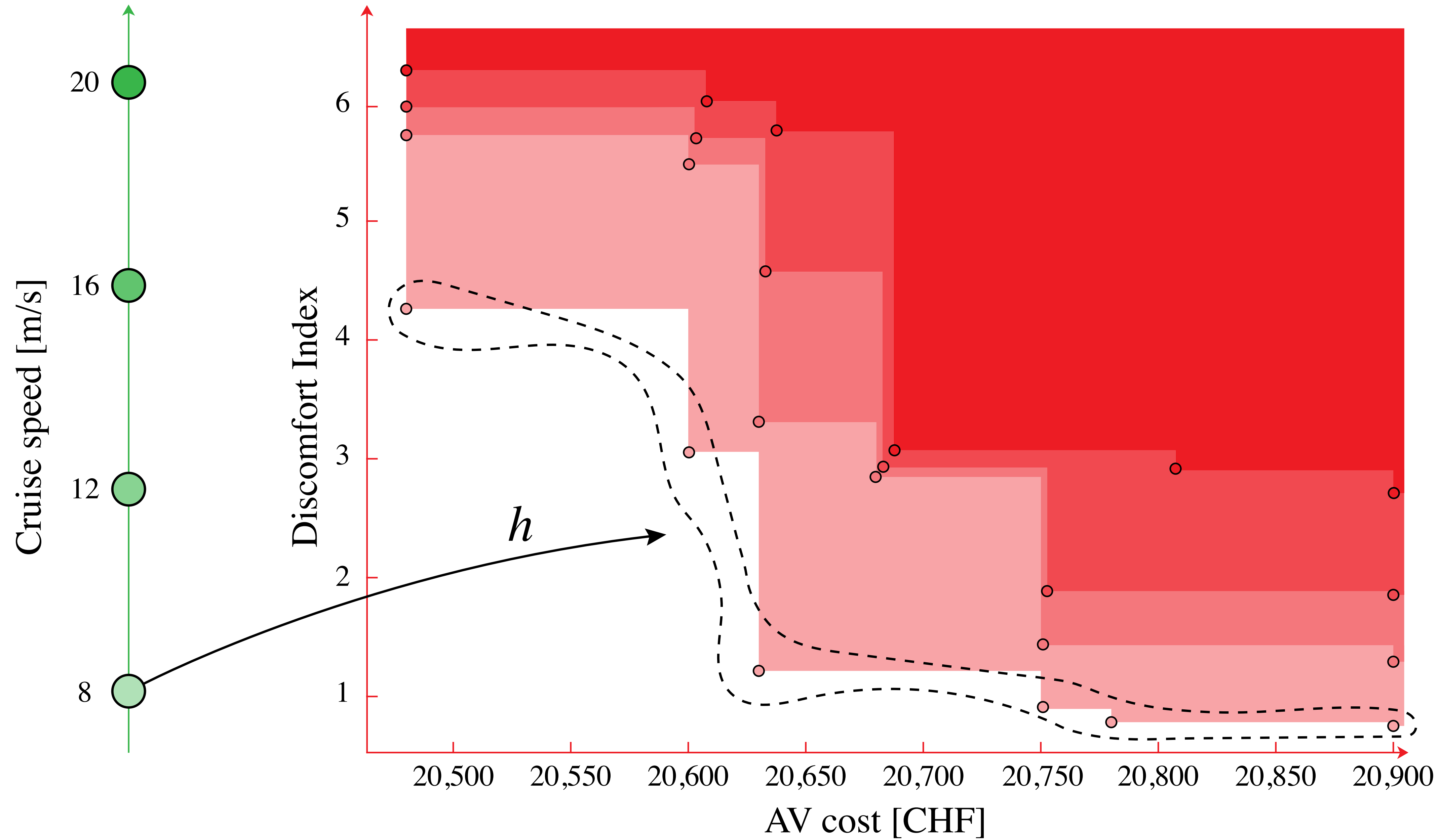
Solution of DPs



Fix **functionalities**, look at minimal **cost** and **discomfort**



*Details of autonomy, both **hardware** and **software***



***Monotonicity:** Higher achievable speeds will not require **less** resources*

Takaways

- Using co-design, it is **easy** to formalize **hierarchical models** (never possible before)
  We formalized AVs all the way from sensor selection to control and perception algorithms

- Very **intuitive** modeling approach (no acrobatics like common in optimization theory)
  *The **interpreter** allows one to easily model problems of interest*

- **Rich modeling capabilities:**
  ***Simulation**: E.g., POMDPs for brake control*
  ***Catalogues**: E.g., Sensors, vehicles, computers, algorithms, ...*
  ***Analytical**: E.g., LQG closed-form solutions, discomfort models, ...*

- **Compositionality** and **modularity** allow **interdisciplinarity**
  *We did all of this, but technically this could have been possible with different **teams***

- Co-design comes with a **formal language** and an **optimizer**
  *After easily modeling the problem, you can directly solve **queries** of your choice*

- Co-design produces **actionable information** for designers to **reason** about their problems
  *We have shown actionable information for **designers***

Outlook and references

- Showcase **compositionality** by including the co-design of specific **robot tasks** in the co-design of the entire **system**
- In the future:
  - Include the co-design of the **AV** in the co-design of the entire **mobility system**
  - Exploit the framework to synthesize **energy** and **computation-aware** design solutions

- **References:**

G. Zardini, D. Milojevic, A. Censi, E. Frazzoli, "**Co-design of embodied intelligence: a structured approach** “, *in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2021.

G. Zardini, A. Censi, and E. Frazzoli, “**Co-design of autonomous systems: From hardware selection to control synthesis**”, in *2021 20th European Control Conference (ECC), 2021*.

G. Zardini, N. Lanzetti, A. Censi, E. Frazzoli, M. Pavone, "**Co-design to enable user-friendly tools to assess the impact of future mobility solutions**", *arXiv preprint arXiv:2008.08975*, 2021.

This is a **new** topic, we are making an effort in **evangelization:**

We are writing a **book,** teaching **classes**, both at ETH and internationally, and organizing **workshops**

https://applied-compositional-thinking.engineering *https://idsc.ethz.ch/research-frazzoli/workshops/compositional-robotics*

http://gioele.science