Solving Interoperability Between Digital Twins

John Morris

PhD Candidate, PLM Applications Engineer
Dept. of Mechanical Engineering
Clemson University

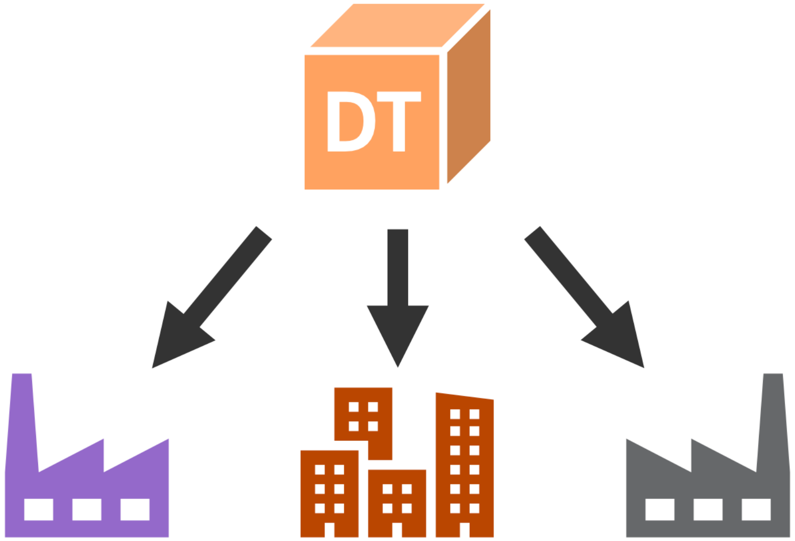
PLMC

CEDAR
Clemson Engineering Design Applications and Research

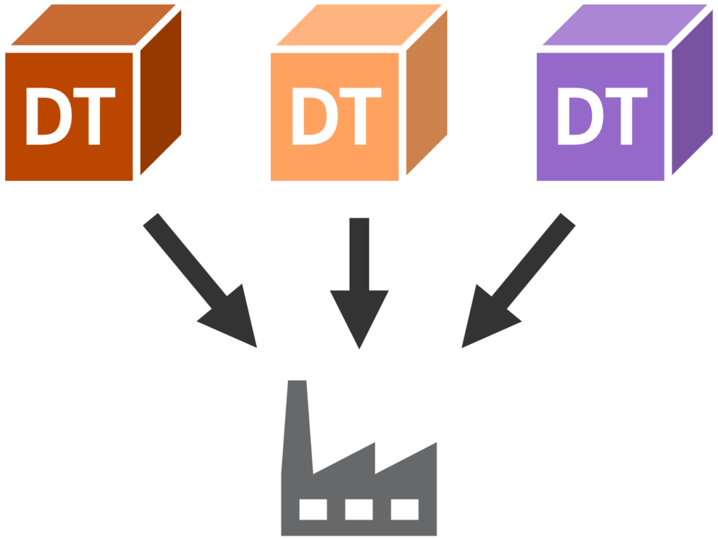
CLEMSON

Off-the-shelf Digital Twins must be interoperable*

Maintains meaning in different contexts



Maintains meaning with different DTs

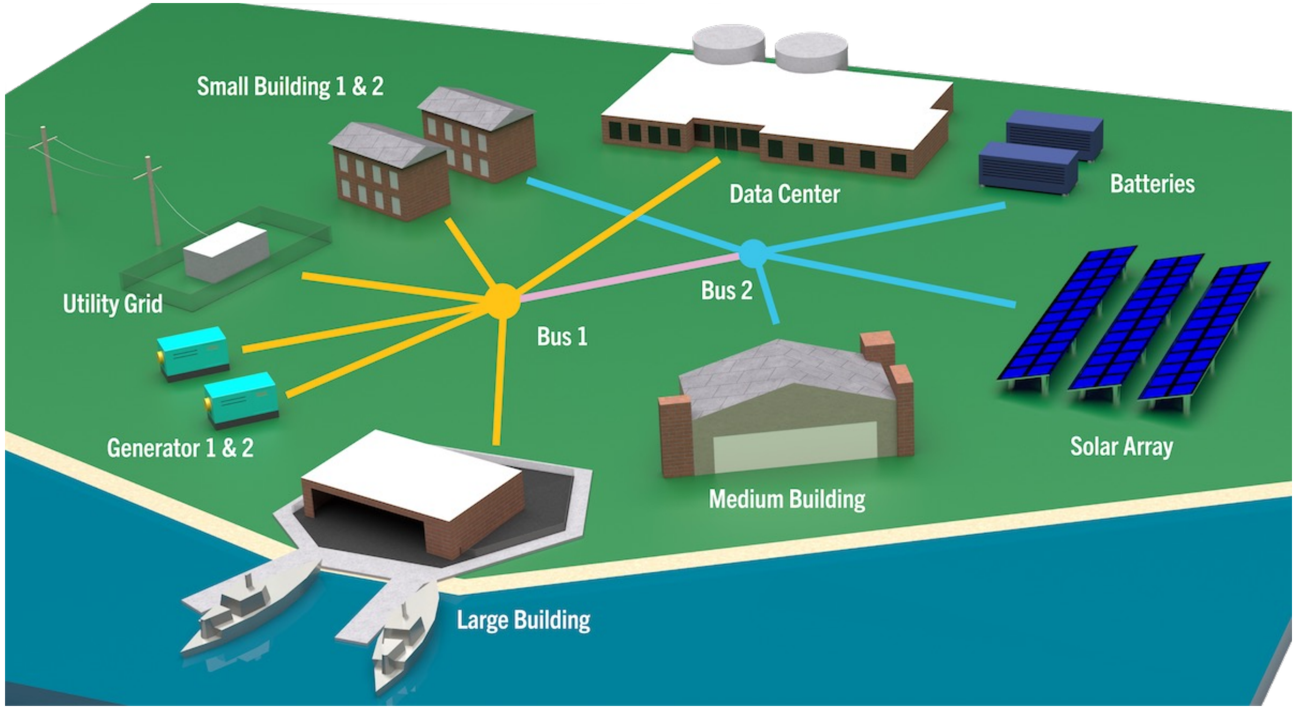


*Similar terms include composable, extensible, interactive, modular, etc.

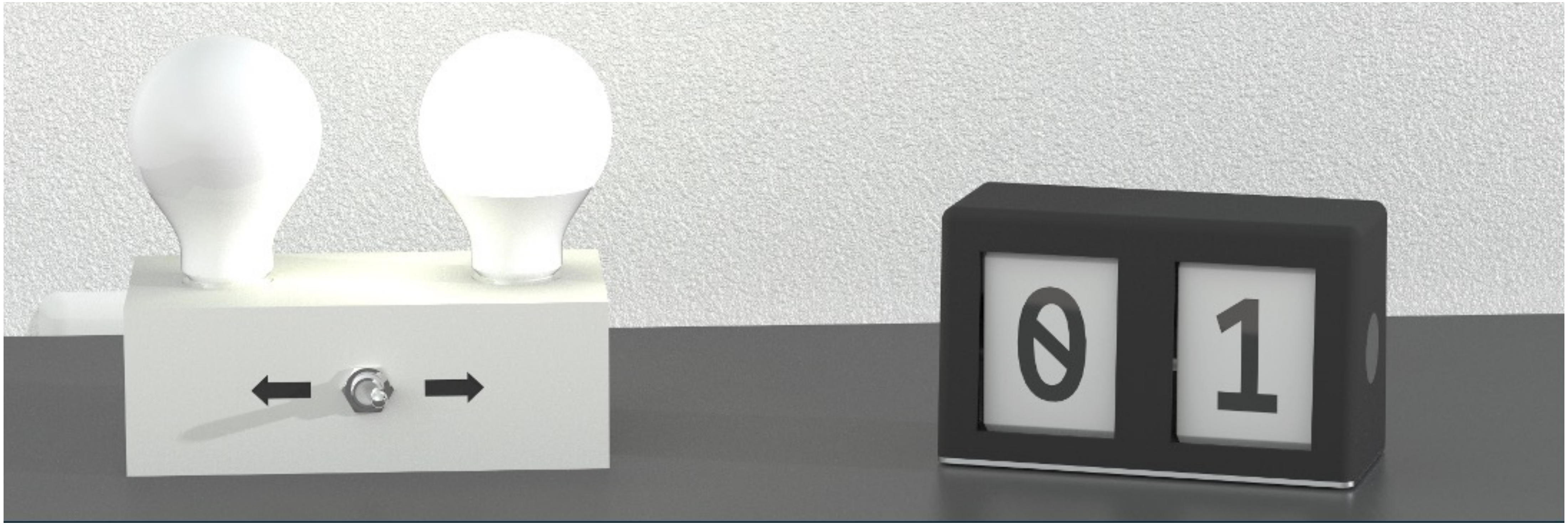
Requires multiple-front approach to interoperability

Syntactic Interoperability:
The definition of the system is truthful across all interactions

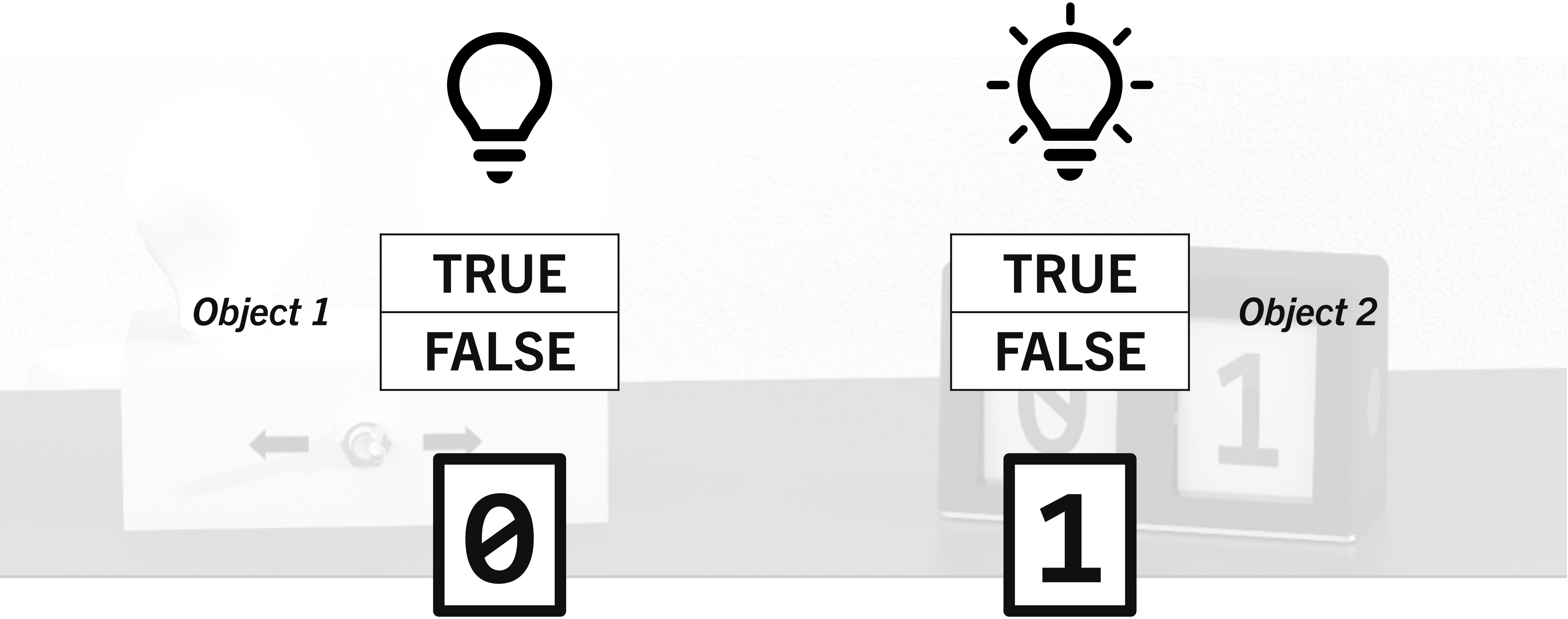
Semantic Interoperability:
The behavior of the system is truthful across all interactions



DTs are systems synchronized with another system of interest



Both systems are scoped to a similar set of variables



Object 1

TRUE
FALSE

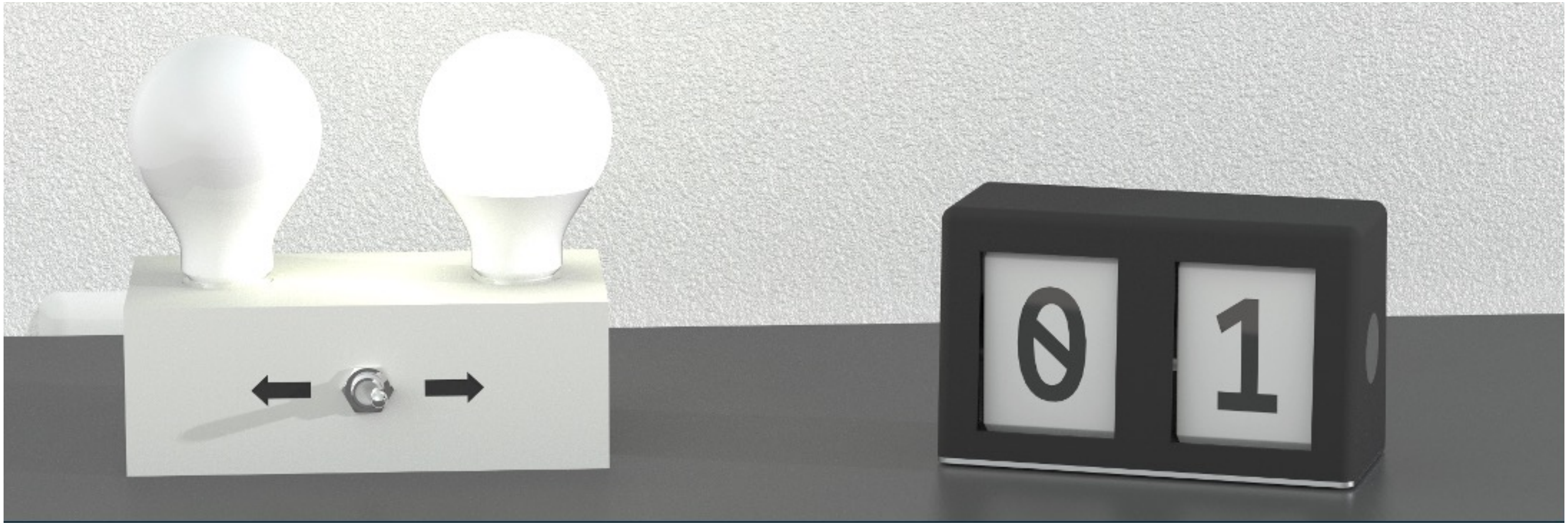
0

Object 2

TRUE
FALSE

1

DTs provide observations for the system facts



System of interest

Digital Twin

Synchronized DTs provide observations for measured system facts

System of interest

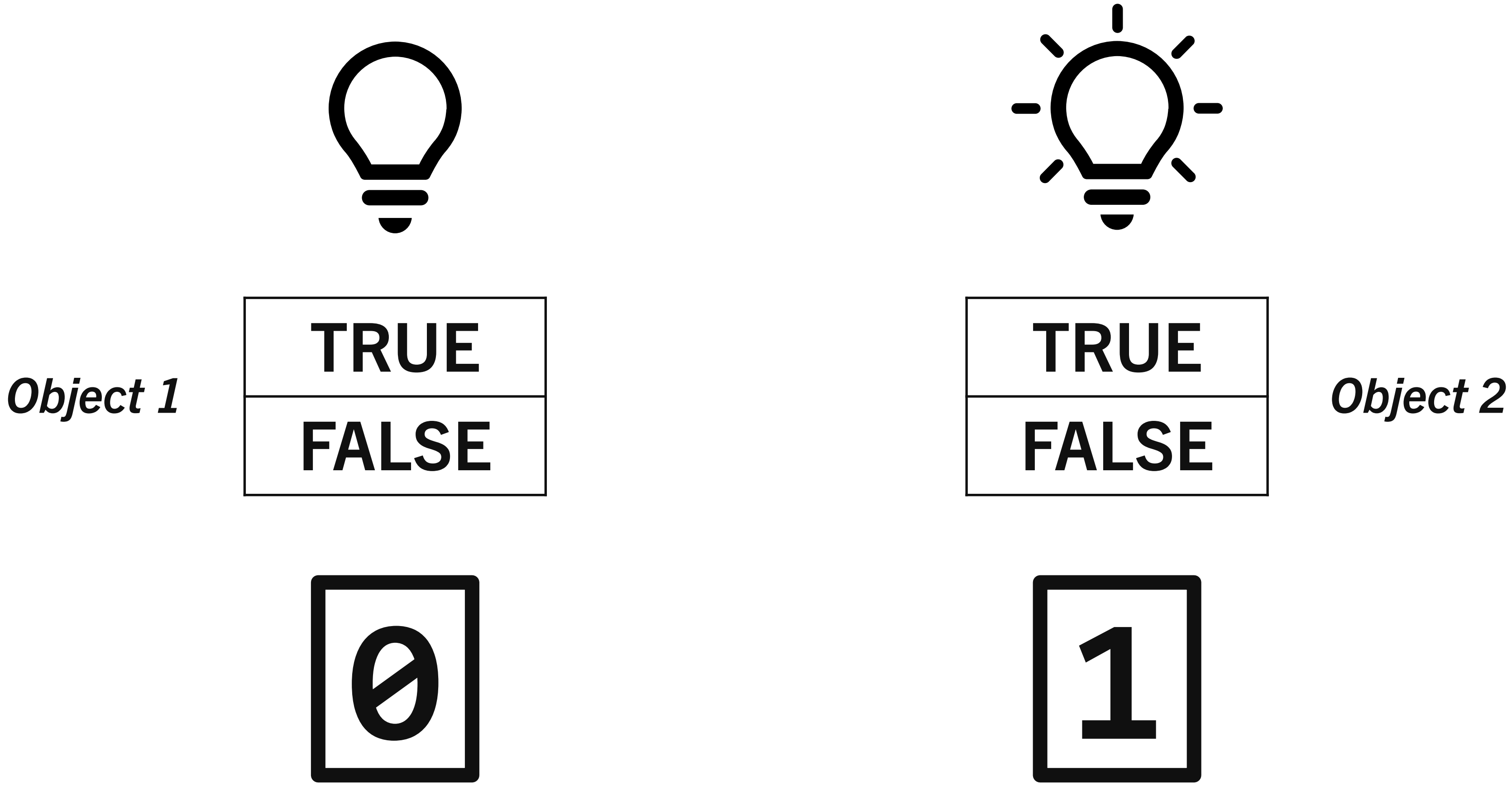
Digital Twin

How are observations provided when full synchronization is impossible?

System of interest

Digital Twin

The DT shares behavior with the system of interest



Object 1

TRUE

FALSE

0

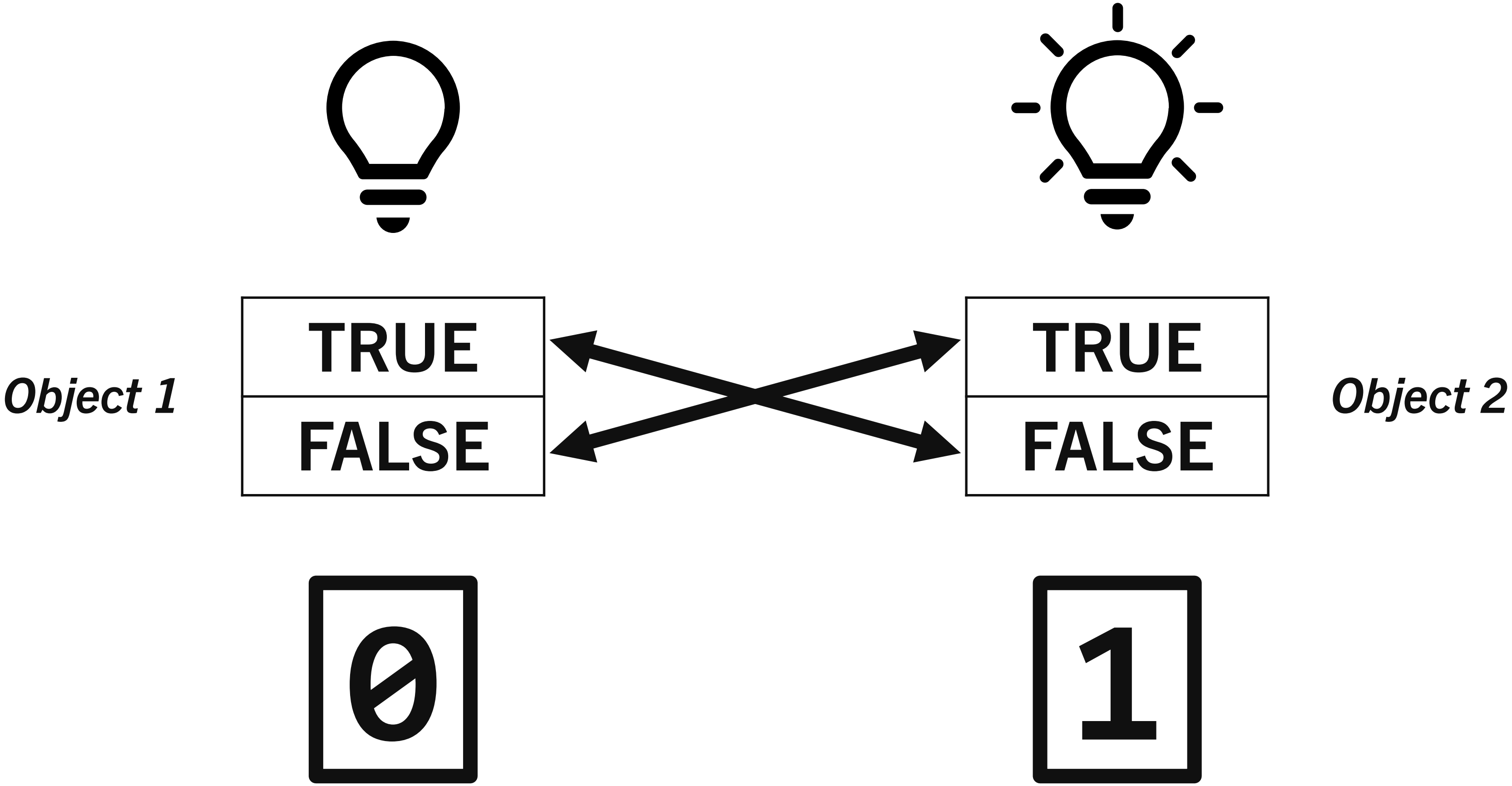
TRUE

FALSE

Object 2

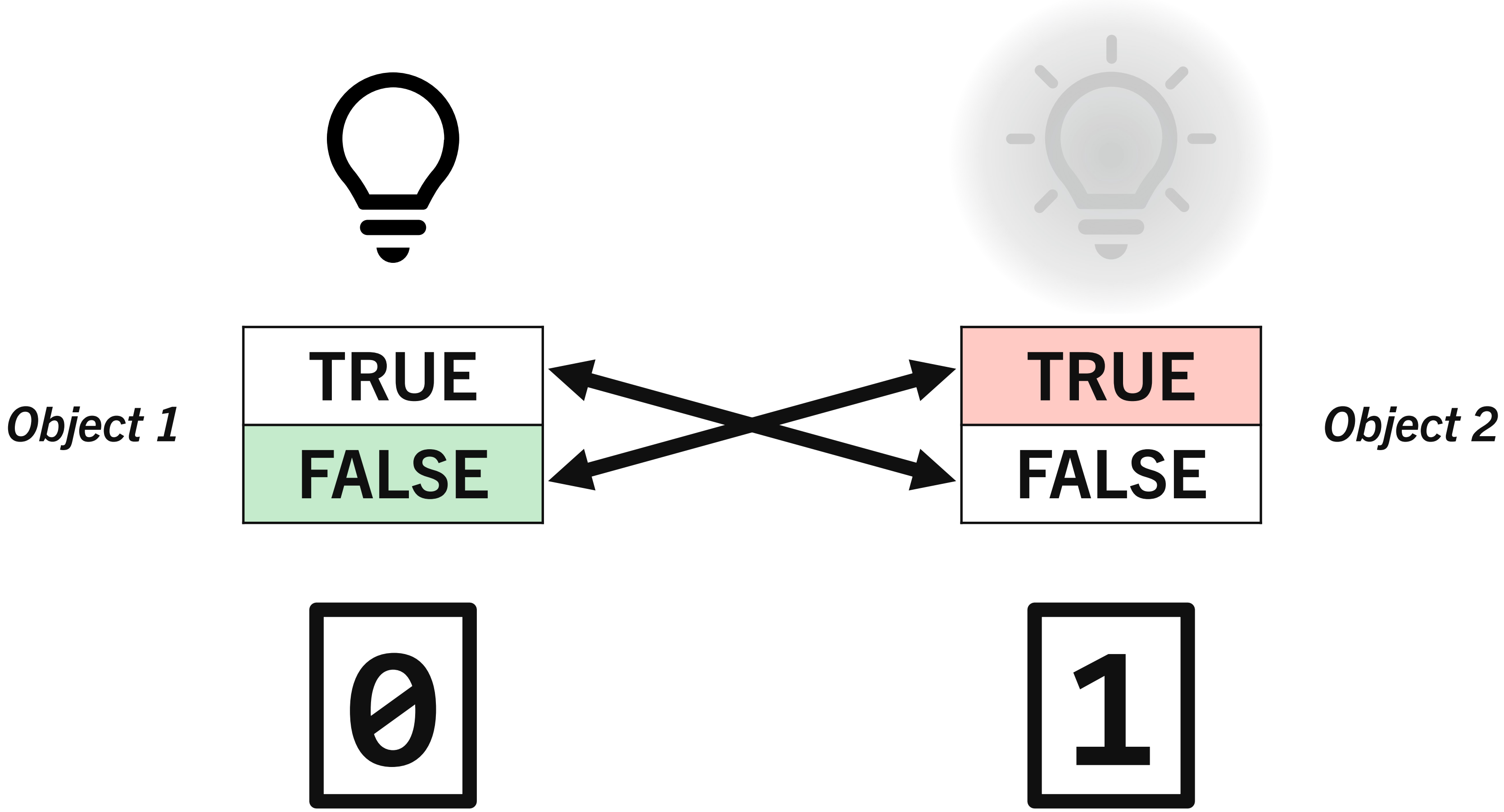
1

Behaviors are the constraints between system states



Polderman, J. W., & Willems, J. C. (1998). Dynamical Systems. In Introduction to Mathematical Systems Theory: A Behavioral Approach (Vol. 26, pp. 1–25). Springer.

Simulations are conducted by providing inputs and outputs to models



Object 1

TRUE

FALSE

0

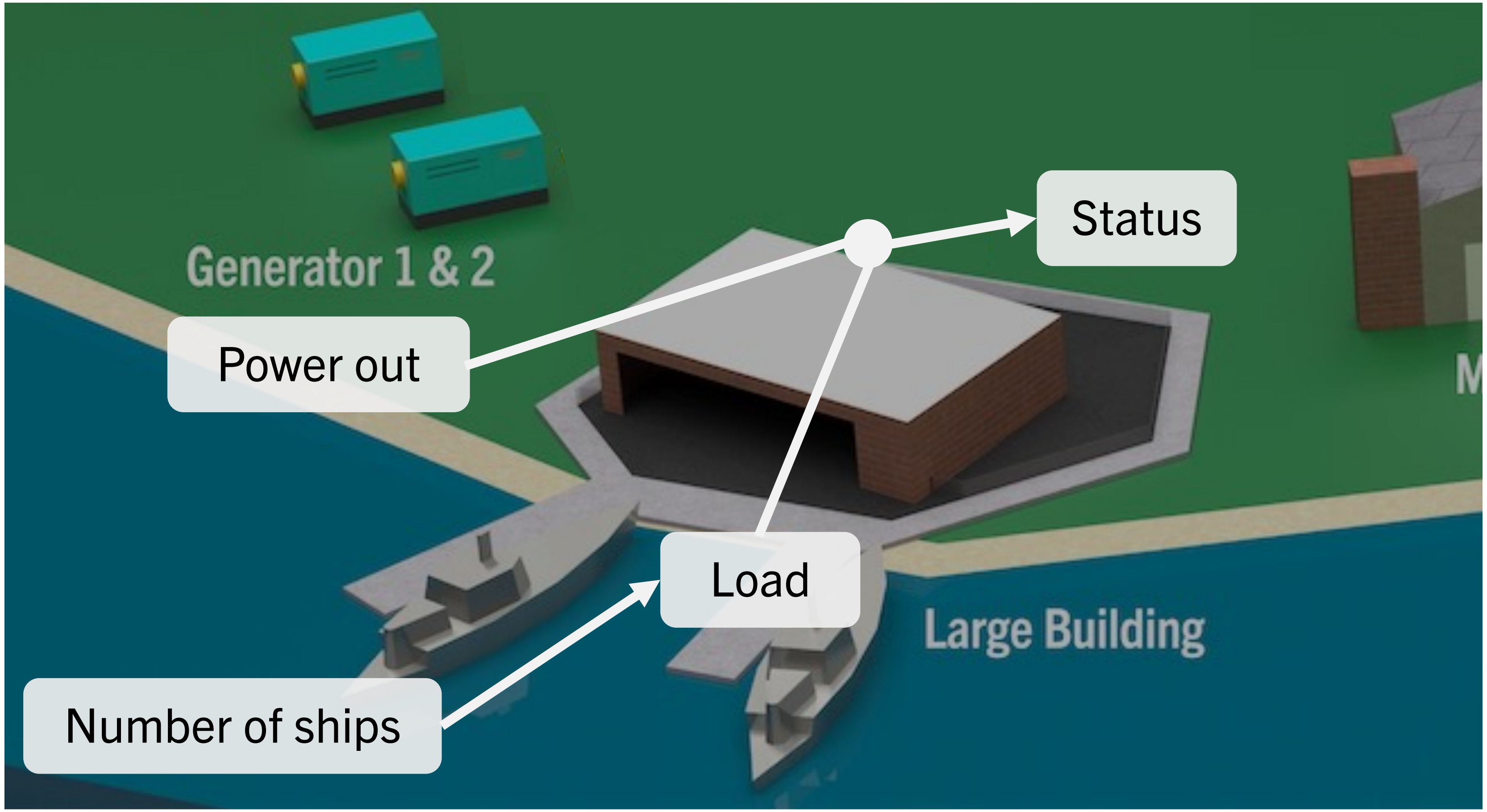
Object 2

TRUE

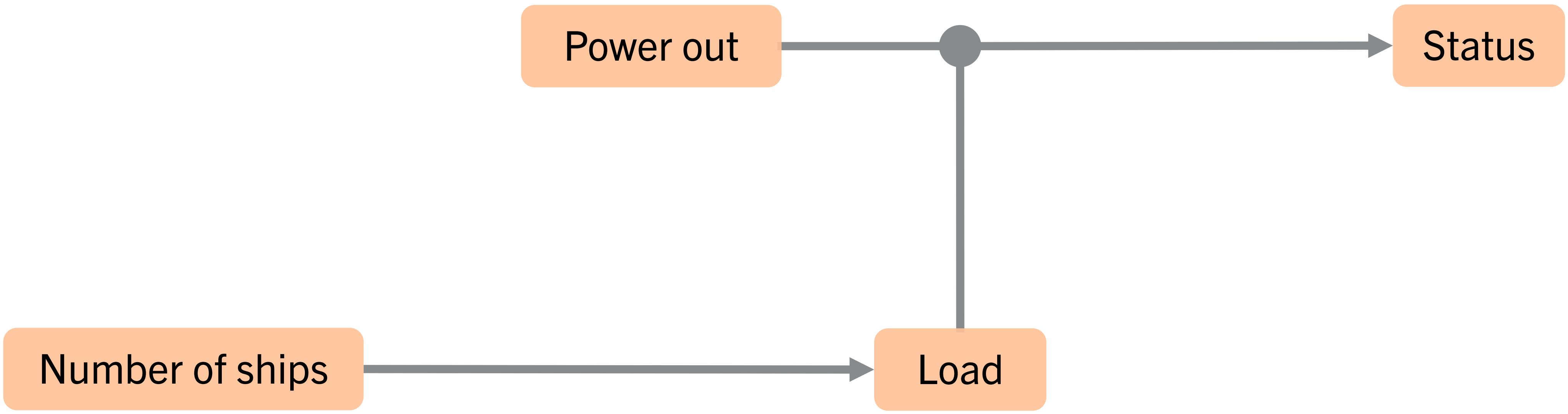
FALSE

1

Constraint hypergraphs reduce all system behavior to a single model

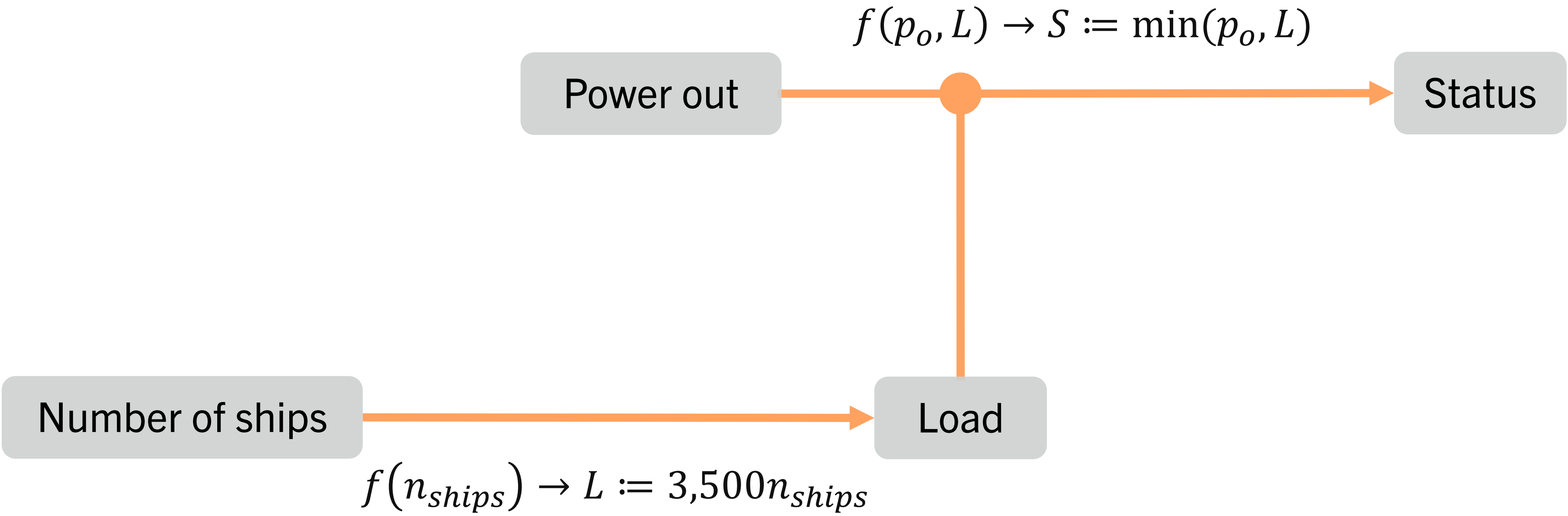


Nodes represent facts—variables—about the system



Nodes represent sets of values

Edges represents constraints on system variables

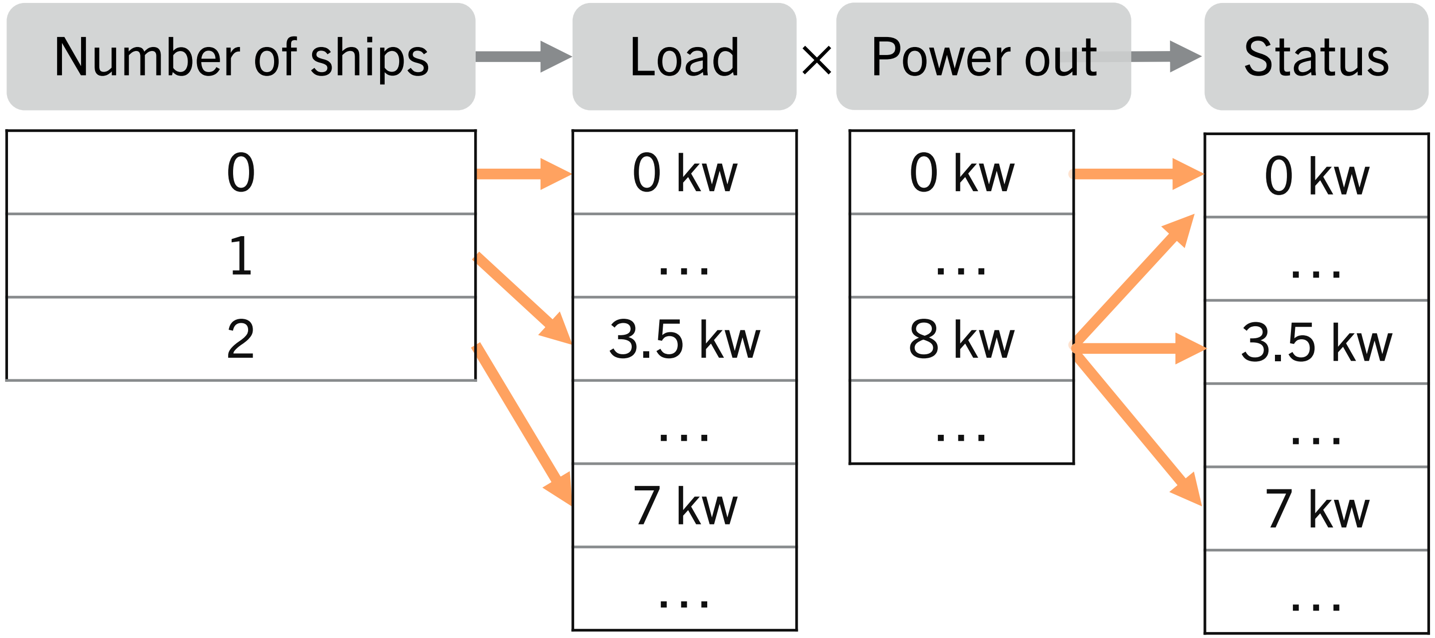

$f(p_o, L) \to S \coloneqq \min(p_o, L)$
Power out
Status
Number of ships
Load
$f(n_{ships}) \to L \coloneqq 3{,}500 n_{ships}$

Constraints are multiple-arity, mapping combinations of multiple node values to one node

Fully constrained by definition

Functions map every set of nodes to a single value that is guaranteed to exist

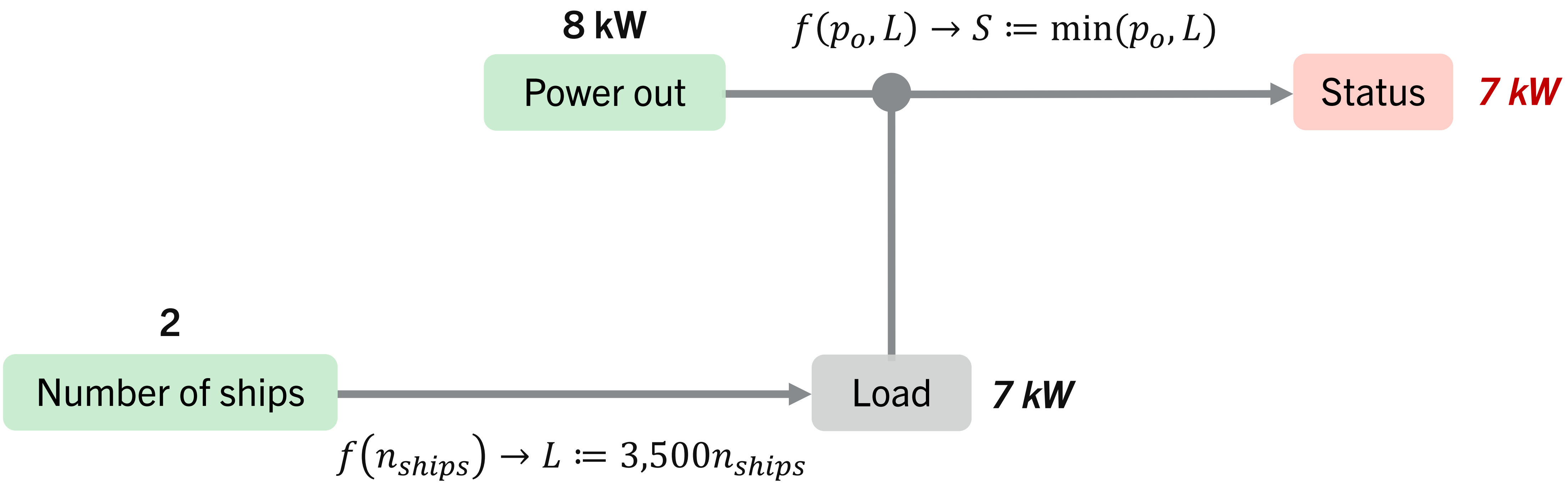
Each path describes a valid configuration of the system



Number of ships	Load	Power out	Status
0	0 kw	0 kw	0 kw
1	…	…	…
2	3.5 kw	8 kw	3.5 kw
	…	…	…
	7 kw		7 kw
	…		…

Morris, J., Mocko, G., & Wagner, J. (2025). Unified System Modeling and Simulation via Constraint Hypergraphs. Journal of Computing and Information Science in Engineering, [In Press]

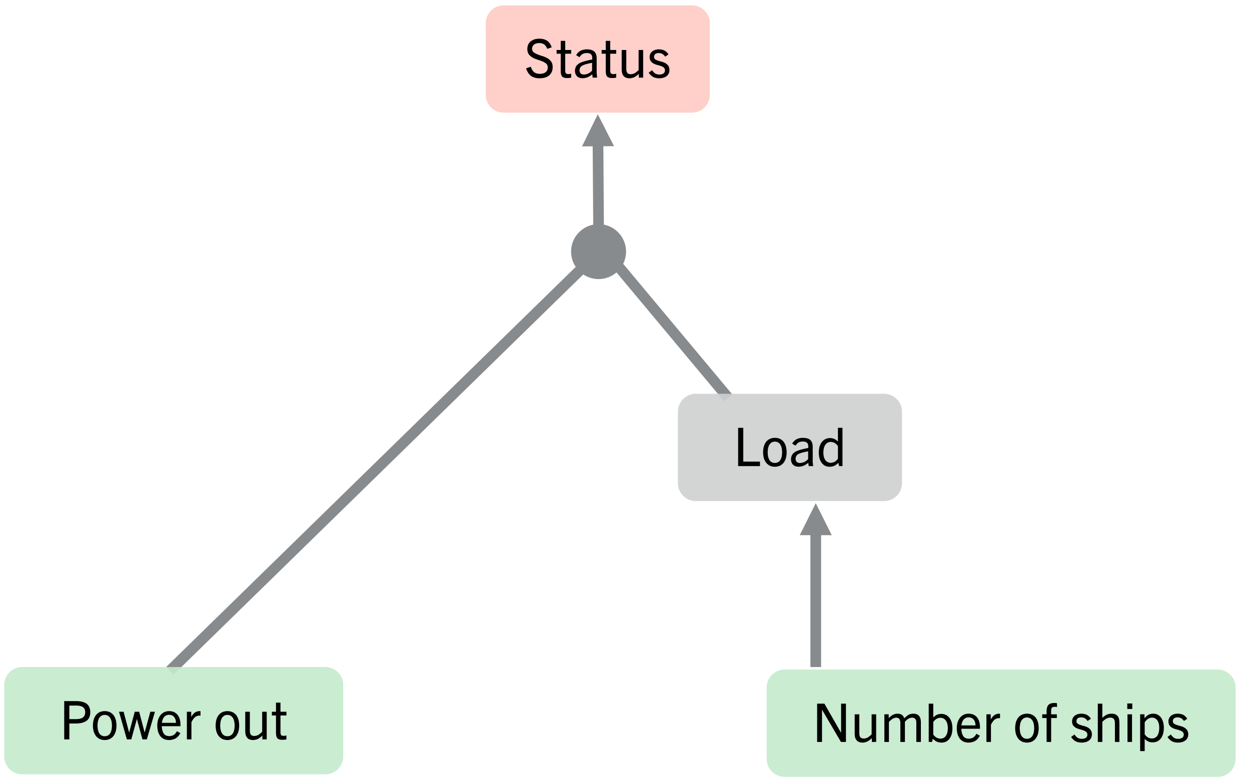
Constraint Hypergraphs engender simulation through function composition



8 kW
Power out
$f(p_o, L) \rightarrow S \coloneqq \min(p_o, L)$
Status
7 kW
2
Number of ships
Load
7 kW
$f(n_{ships}) \rightarrow L \coloneqq 3{,}500 n_{ships}$

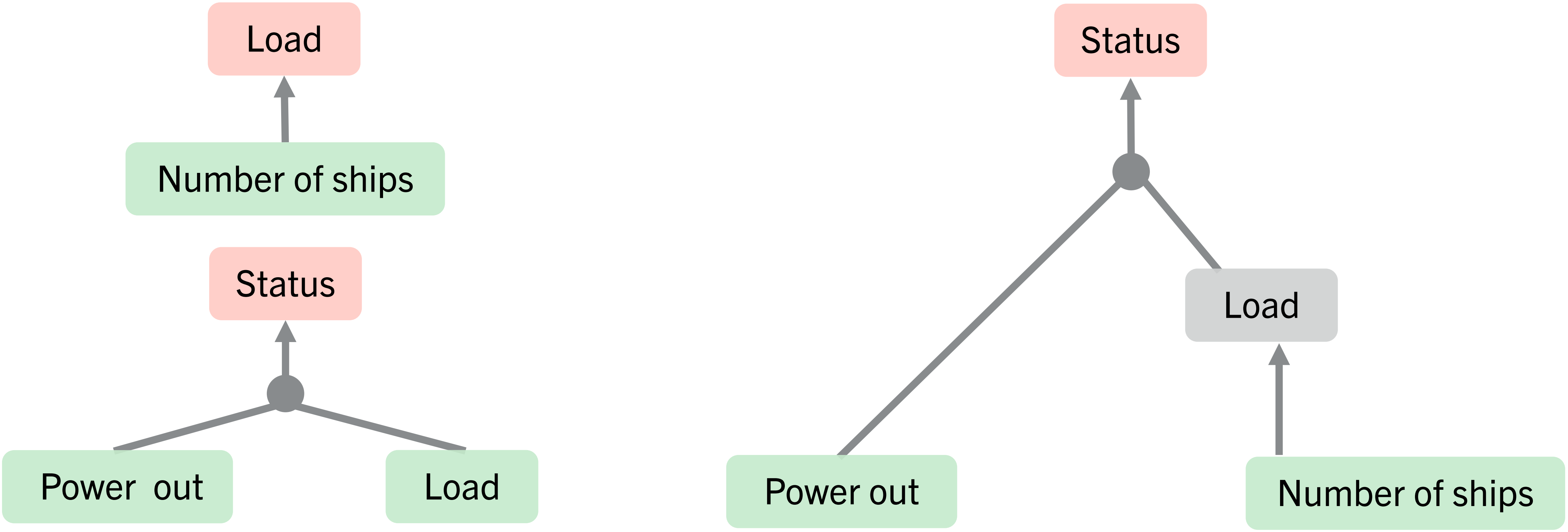
Simulation is the observation of a system’s state without real-world measurements

Simulation is represented as a path through the hypergraph



Paths in a hypergraph form *trees*

The set of all paths in the hypergraph show all possible simulations

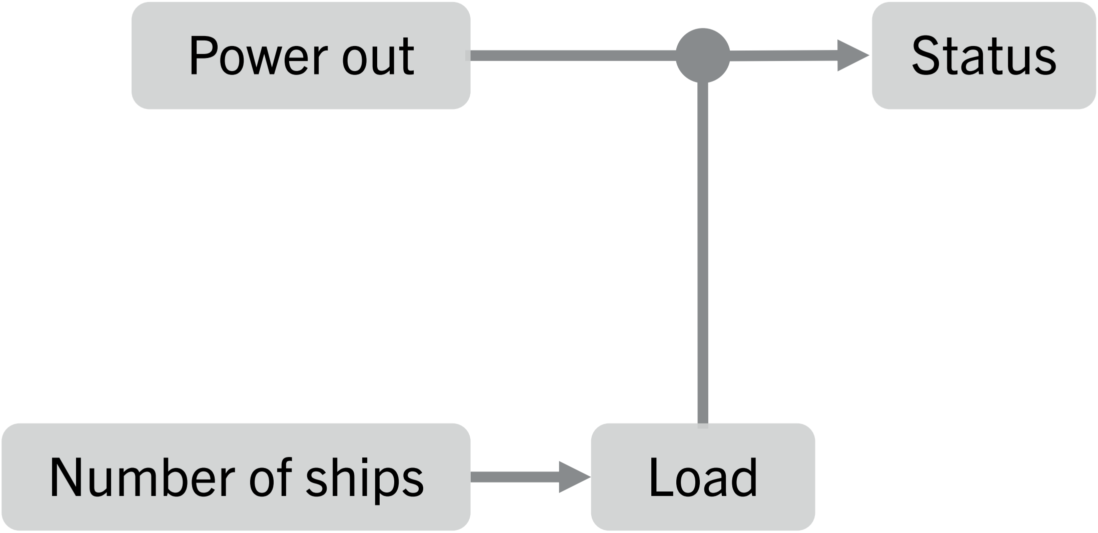


Paths in a hypergraph form *trees*

Hypergraphs expose system information to external agents

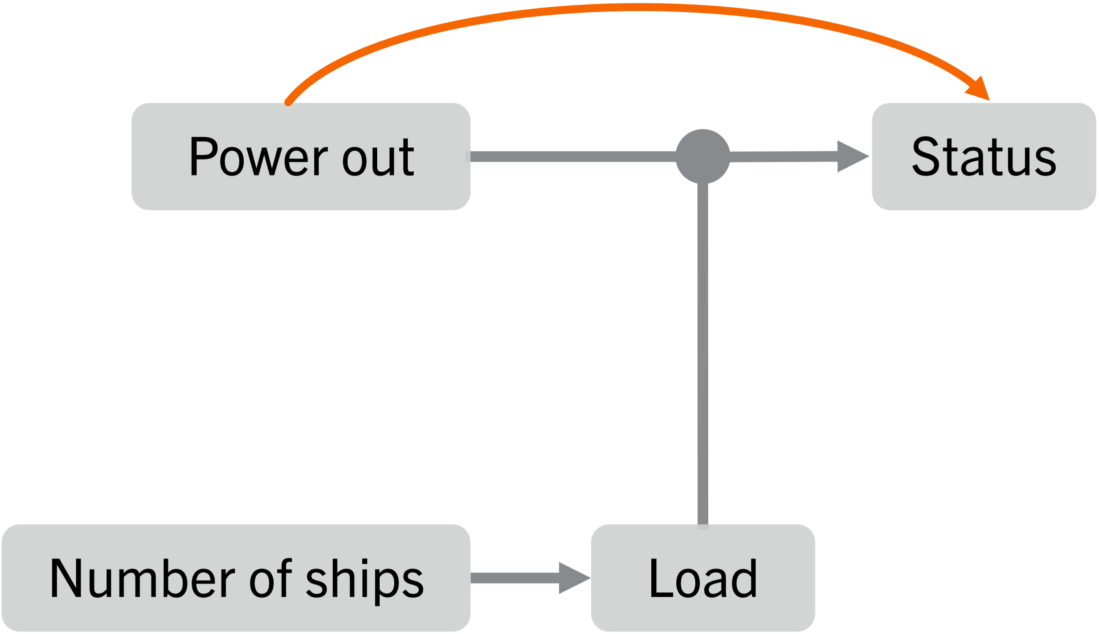
Interfacing agents—humans, computers, sensors, dashboards, etc.—can request a value for a node

If the node has an observed value, or can be simulated from a set of inputs, the hypergraph can automatically deliver the value



Independent behaviors make the hypergraphs immediately modifiable

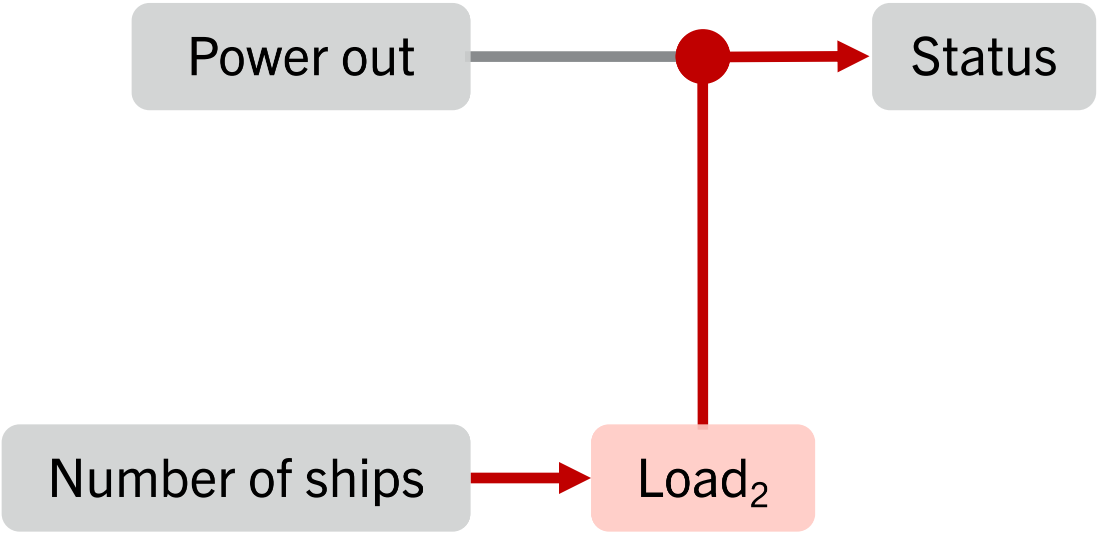
Functions can be rewritten, removed, or added without affecting the consistency of the graph *(no side effects)*



Independent behaviors make the hypergraphs immediately modifiable

Functions can be rewritten, removed, or added without affecting the consistency of the graph *(no side effects)*

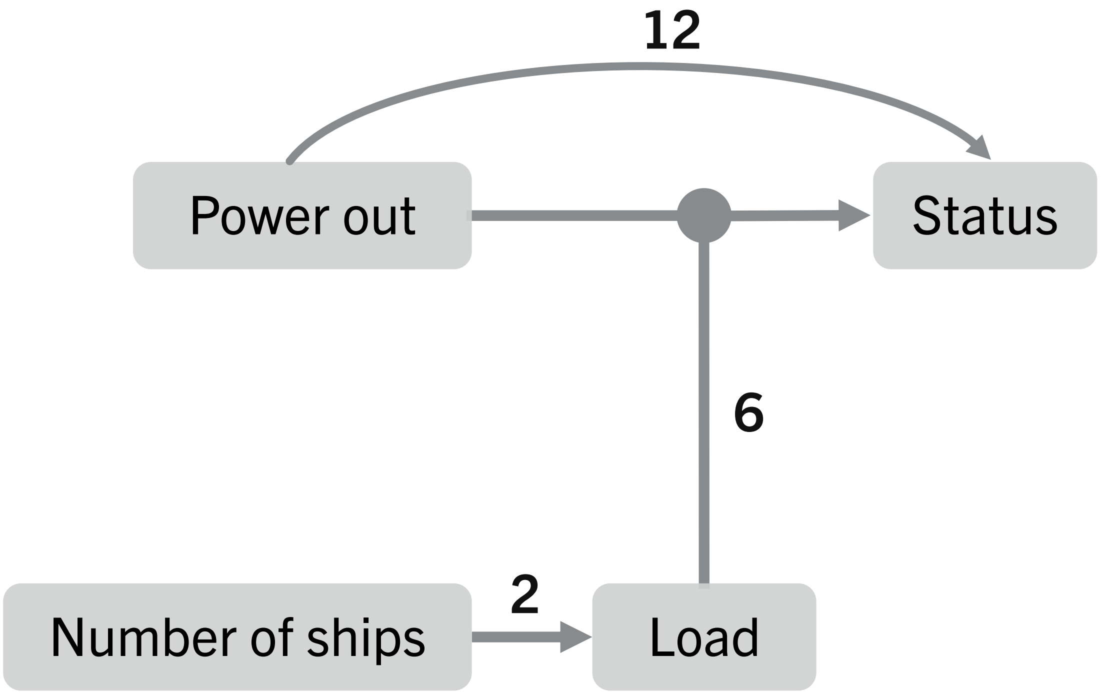
Nodes can be modified only if all connecting edges are also updated to maintain consistency *(only local side effects)*



Weights allow model characterization

Edge weights allow models to be compared (model selection)

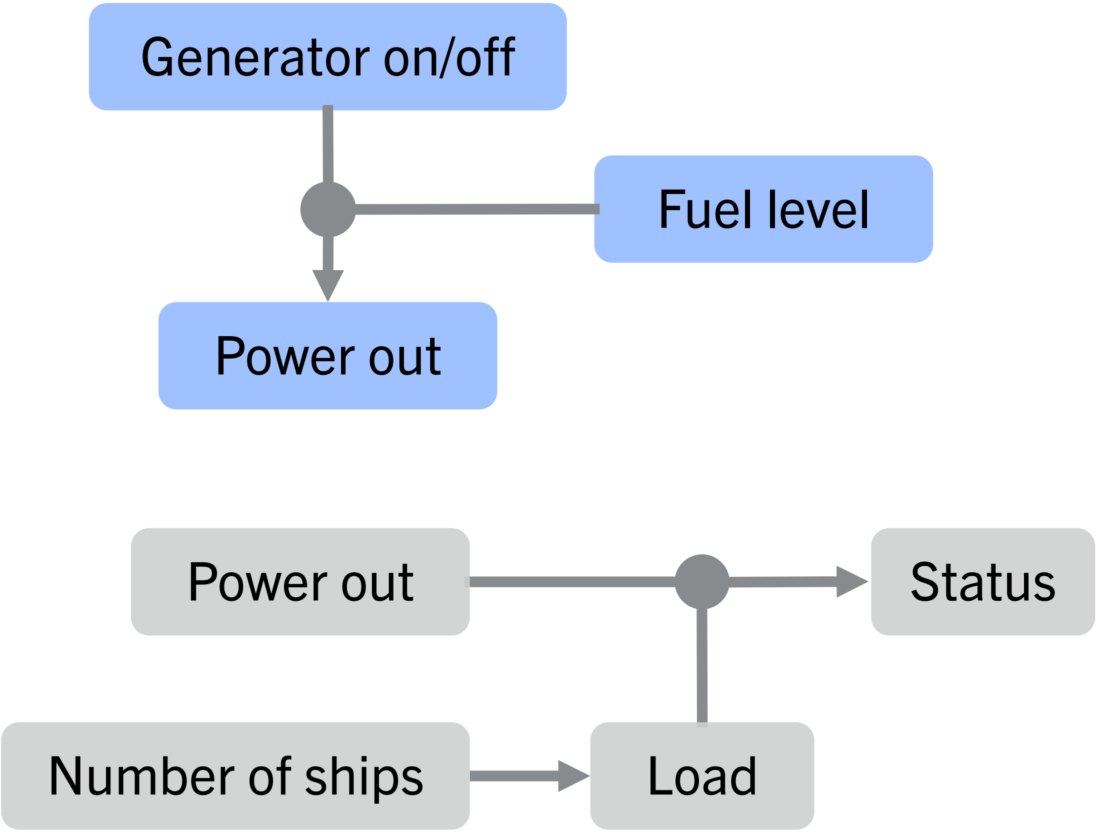
Weights can reference a model's computation time, availability, parallelization, etc.



Morris, J., Mocko, G., & Wagner, J. (2025). Unified System Modeling and Simulation via Constraint Hypergraphs. Journal of Computing and Information Science in Engineering, [In Press]

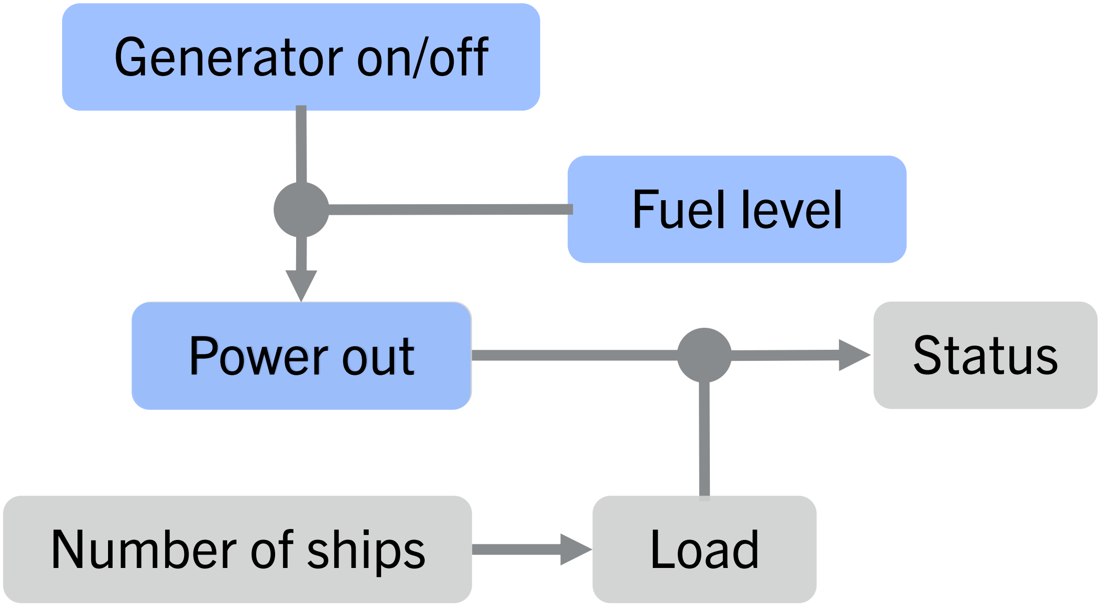
Composable with other graphs along shared nodes (union operation)

All simulation paths remain consistent

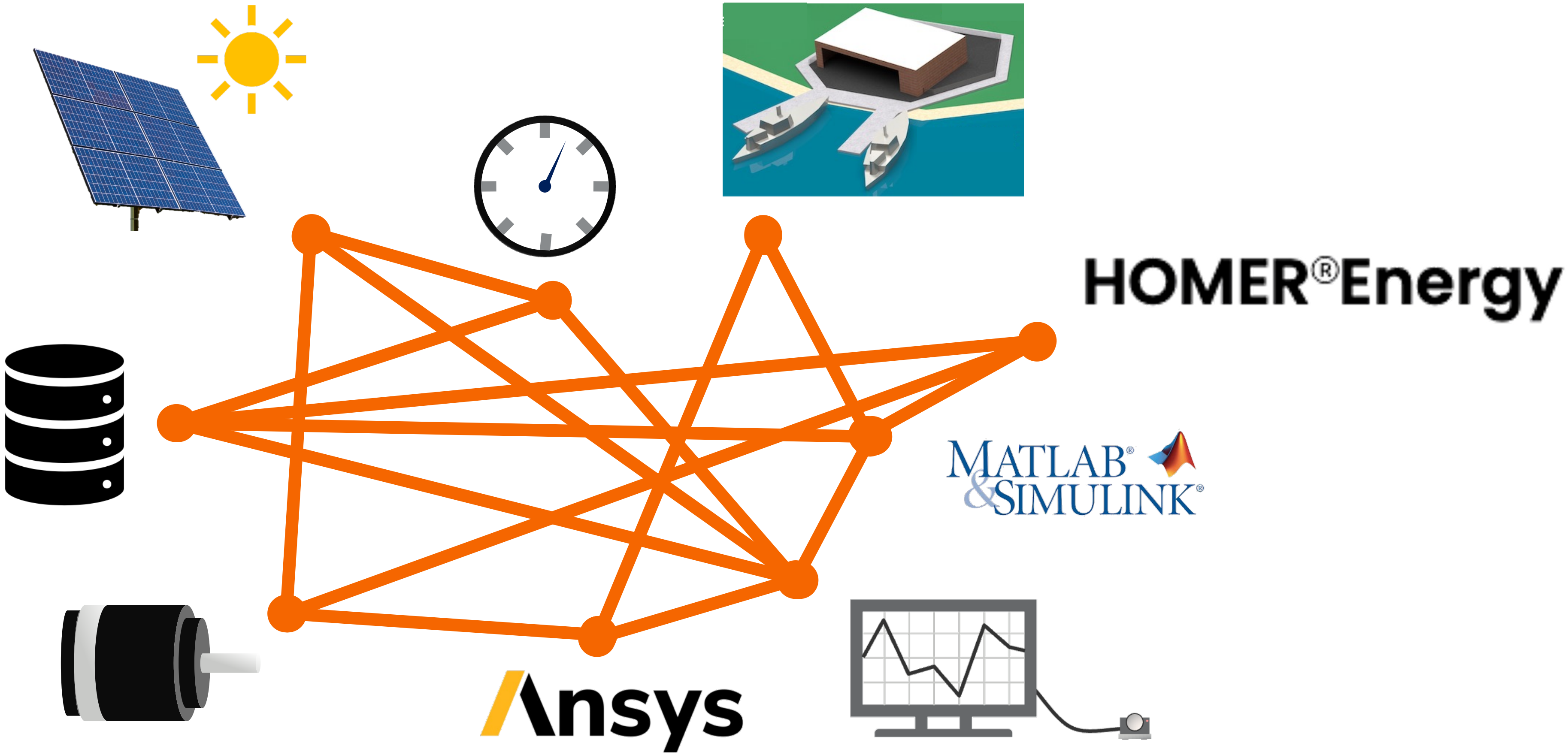


Emergent behavior is provided entirely from composition

All simulation paths remain consistent



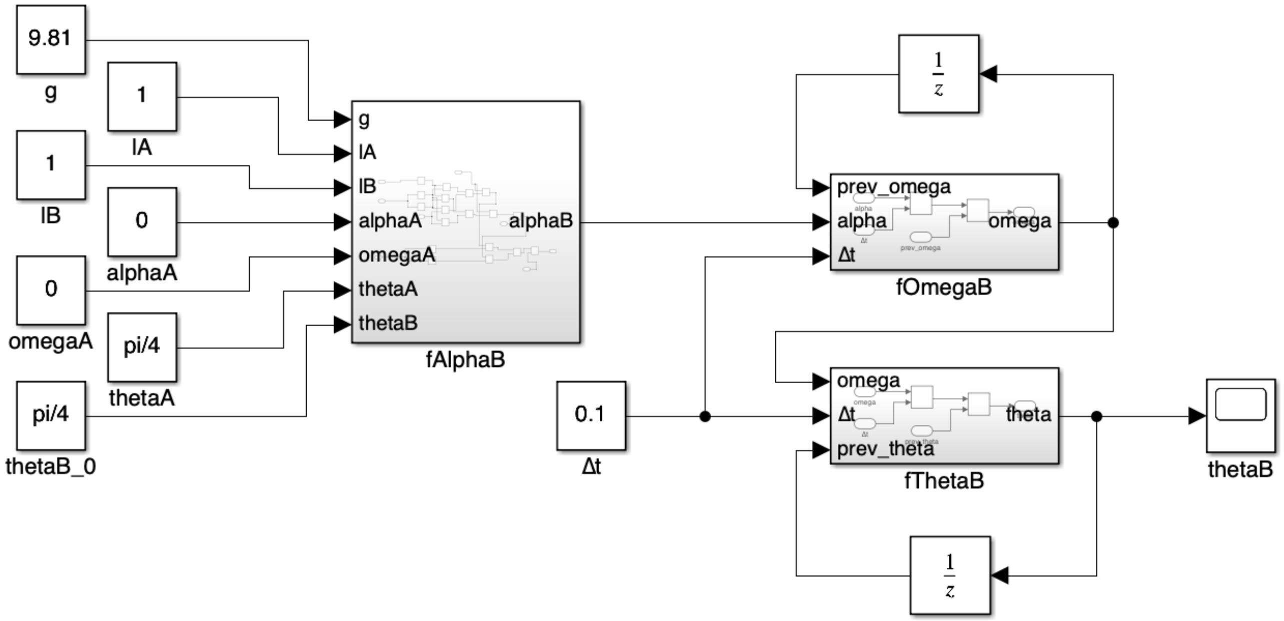
Complex systems require models to be represented and synchronized between many software applications



Imperative models provide only a single form of simulation

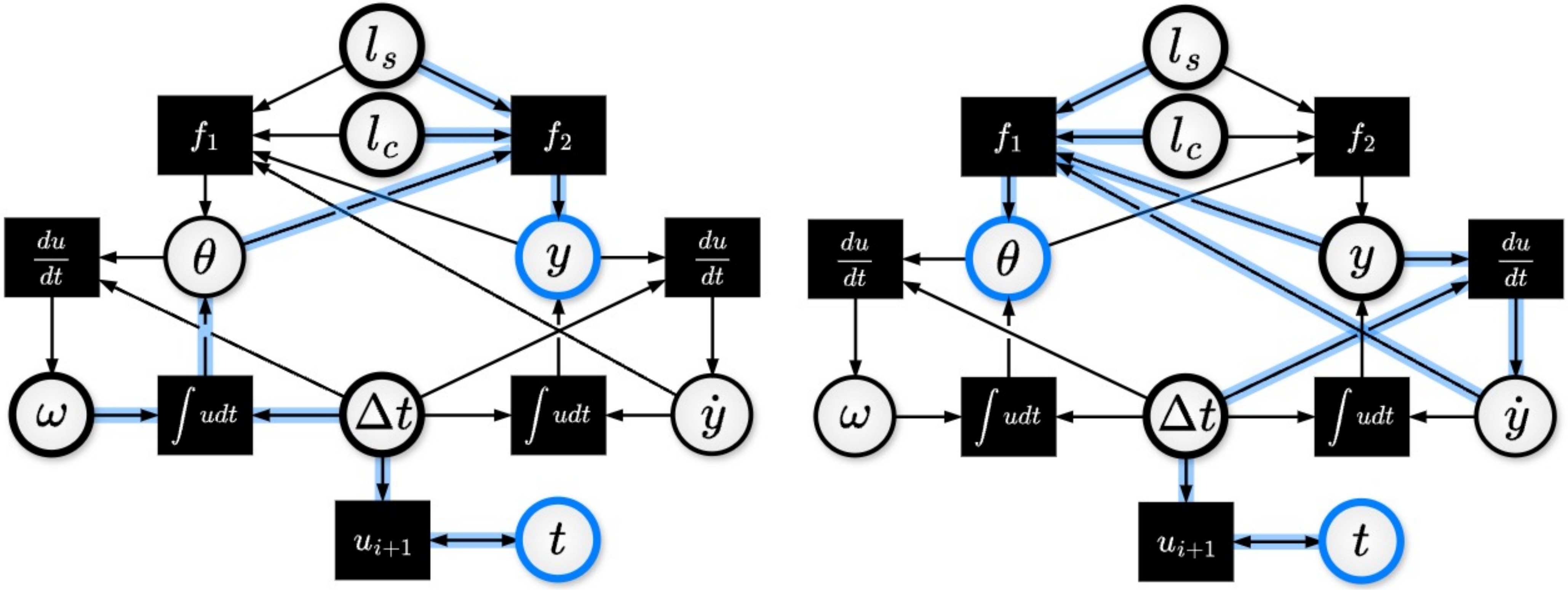
The number of imperative simulations from a system representation with n state variables is the combination of:

$$\sum_{i=1}^{n-1} n - i \binom{n}{i}$$



Morris, J., Mocko, G., Wagner, J., & Ramnath, S. (2025). Declarative Integration of CAD Software into Multi-Physics Simulation via Constraint Hypergraphs. [Submitted to the ASME IDETC-CIE 2025 conference]

Declarative models allow the modeler to focus on what the system is, rather than how it will be simulated

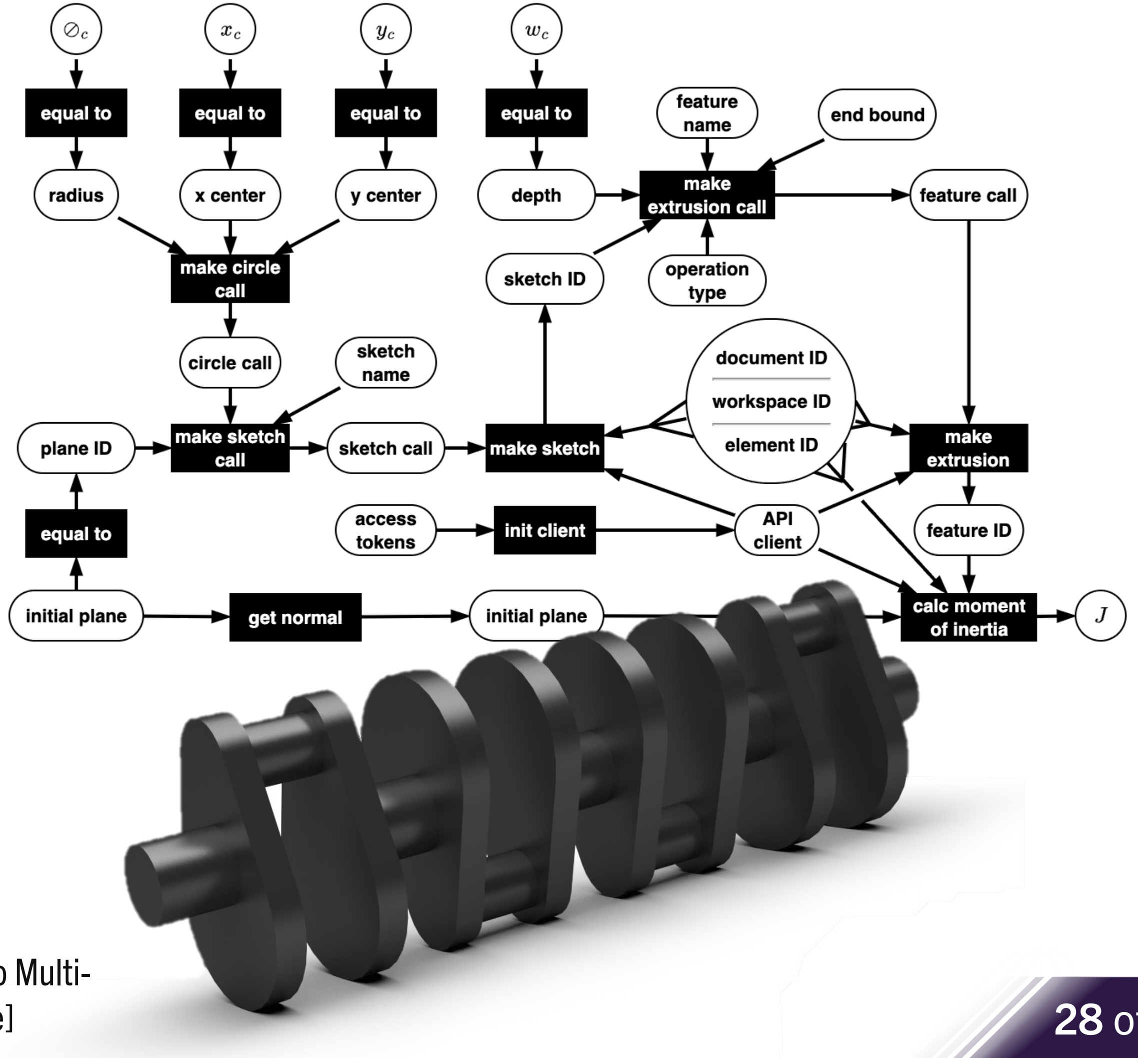


Morris, J., Mocko, G., & Wagner, J. (2025). Effects of Functional and Declarative Modeling Frameworks on System Simulation. [Manuscript Submitted for Review to the ASME Journal of Dynamic Systems, Measurement and Control].

Constraint hypergraphs tie siloed software into a single representation

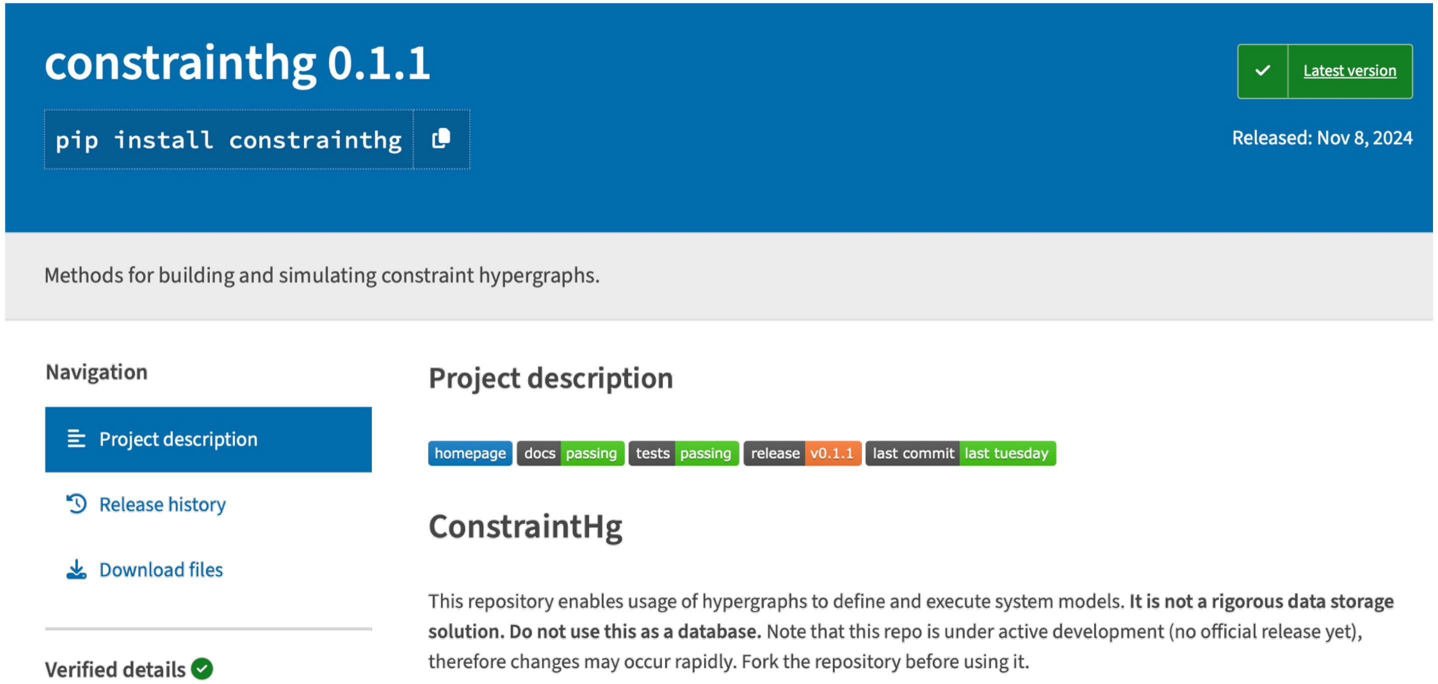
Models that must be simulated in specific software (such as FE models) can be integrated into the system by wrapping API calls as edges

Avoids imperative (hard-wired) connectors between software applications



Morris, J., Mocko, G., Wagner, J., & Ramnath, S. (2025). Declarative Integration of CAD Software into Multi-Physics Simulation via Constraint Hypergraphs. [Submitted to the ASME IDETC-CIE 2025 conference]

Solving engine implementing a Breadth-First Search approach available on the Python Package Index



constrainthg 0.1.1

```
pip install constrainthg
```

Released: Nov 8, 2024

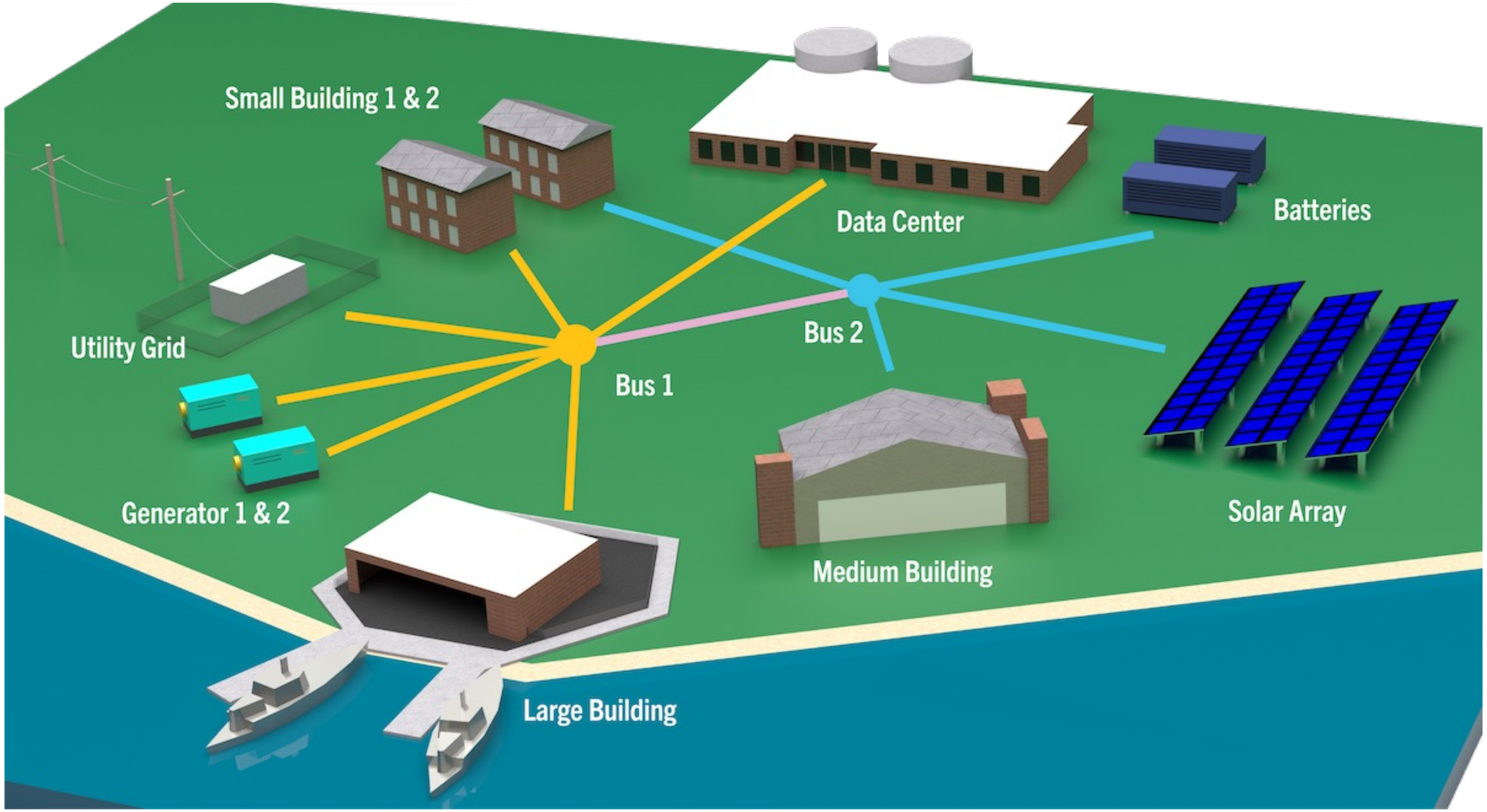
Methods for building and simulating constraint hypergraphs.

Project description

ConstraintHg

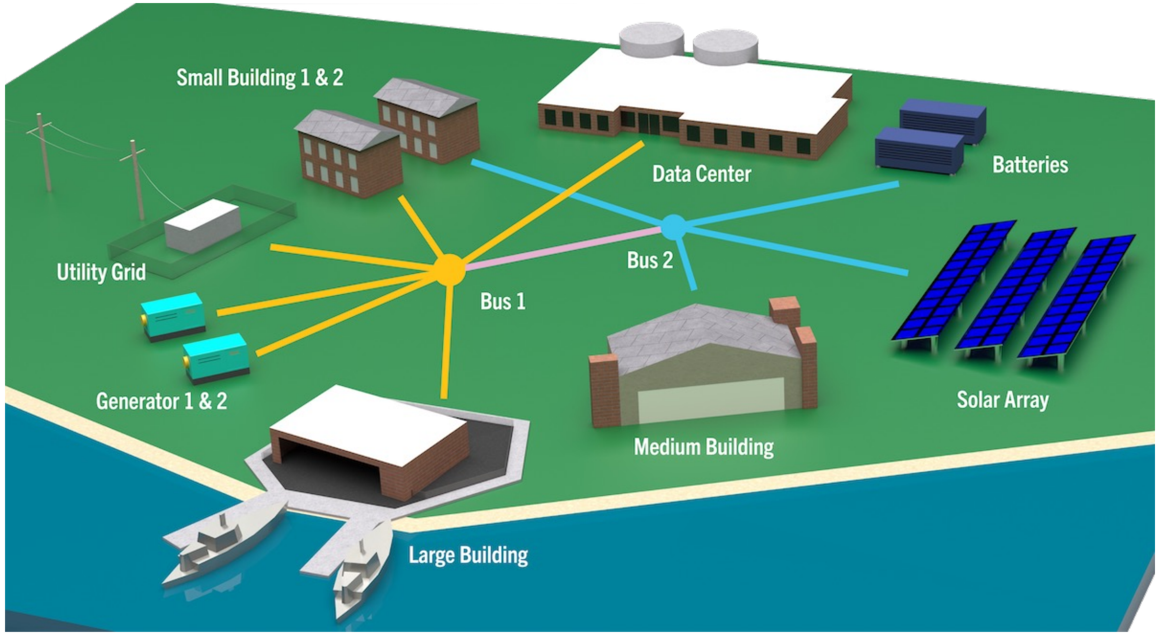
This repository enables usage of hypergraphs to define and execute system models. **It is not a rigorous data storage solution. Do not use this as a database.** Note that this repo is under active development (no official release yet), therefore changes may occur rapidly. Fork the repository before using it.

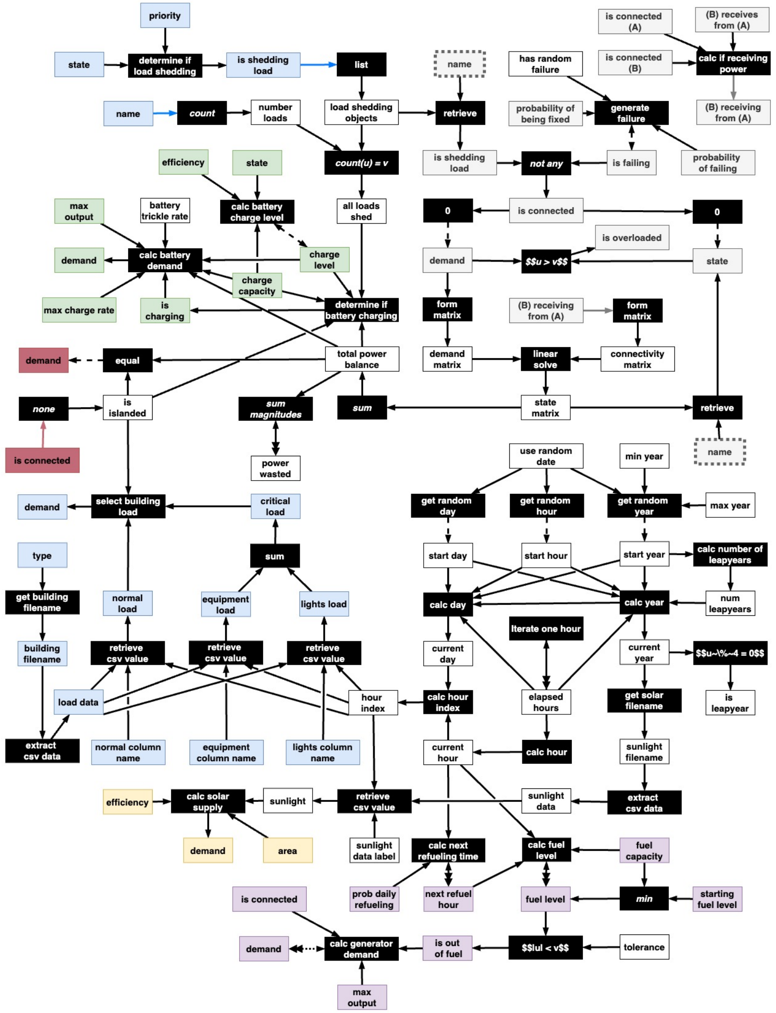
Example with a naval microgrid



Example available on GitHub by following QR code in corner

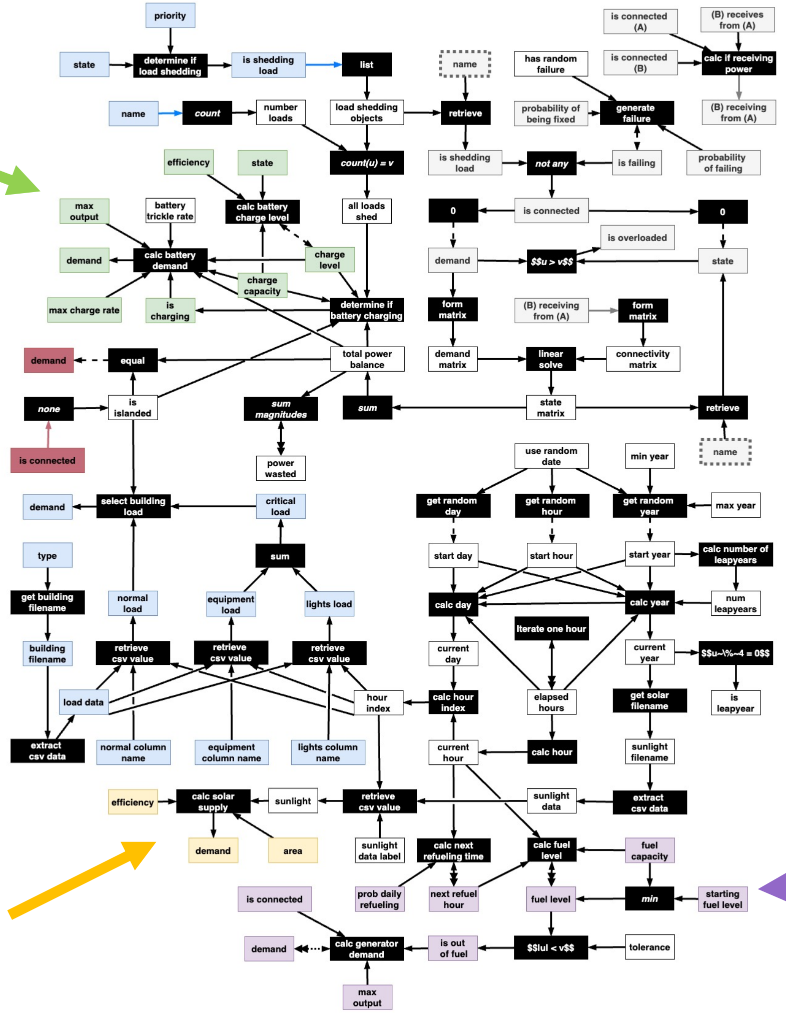
Hypergraph contains 200 nodes and 502 edges





priority
is connected (A)
(B) receives from (A)
state
determine if load shedding
is shedding load
list
name
has random failure
is connected (B)
calc if receiving power
name
count
number loads
load shedding objects
retrieve
probability of being fixed
generate failure
(B) receiving from (A)
efficiency
state
count(u) = v
is shedding load
not any
is failing
probability of failing
max output
battery trickle rate
calc battery charge level
all loads shed
0
is connected
0
is overloaded
demand
calc battery demand
charge level
demand
$$u > v$$
state
charge capacity
max charge rate
is charging
determine if battery charging
form matrix
(B) receiving from (A)
form matrix
demand
equal
total power balance
demand matrix
linear solve
connectivity matrix
none
is islanded
sum magnitudes
sum
state matrix
retrieve
is connected
power wasted
use random date
min year
name
demand
select building load
critical load
get random day
get random hour
get random year
max year
type
sum
start day
start hour
start year
calc number of leapyears
get building filename
normal load
equipment load
lights load
calc day
Iterate one hour
calc year
num leapyears
building filename
retrieve csv value
retrieve csv value
retrieve csv value
current day
current year
$$u~\%~4 = 0$$
load data
hour index
calc hour index
elapsed hours
get solar filename
is leapyear
extract csv data
normal column name
equipment column name
lights column name
current hour
calc hour
sunlight filename
efficiency
calc solar supply
sunlight
retrieve csv value
sunlight data
extract csv data
demand
area
sunlight data label
calc next refueling time
calc fuel level
fuel capacity
is connected
prob daily refueling
next refuel hour
fuel level
min
starting fuel level
demand
calc generator demand
is out of fuel
$$|u| < v$$
tolerance
max output

Regions of the hypergraph represent different *things* in the system



Batteries

Connectivity

Utility Grid

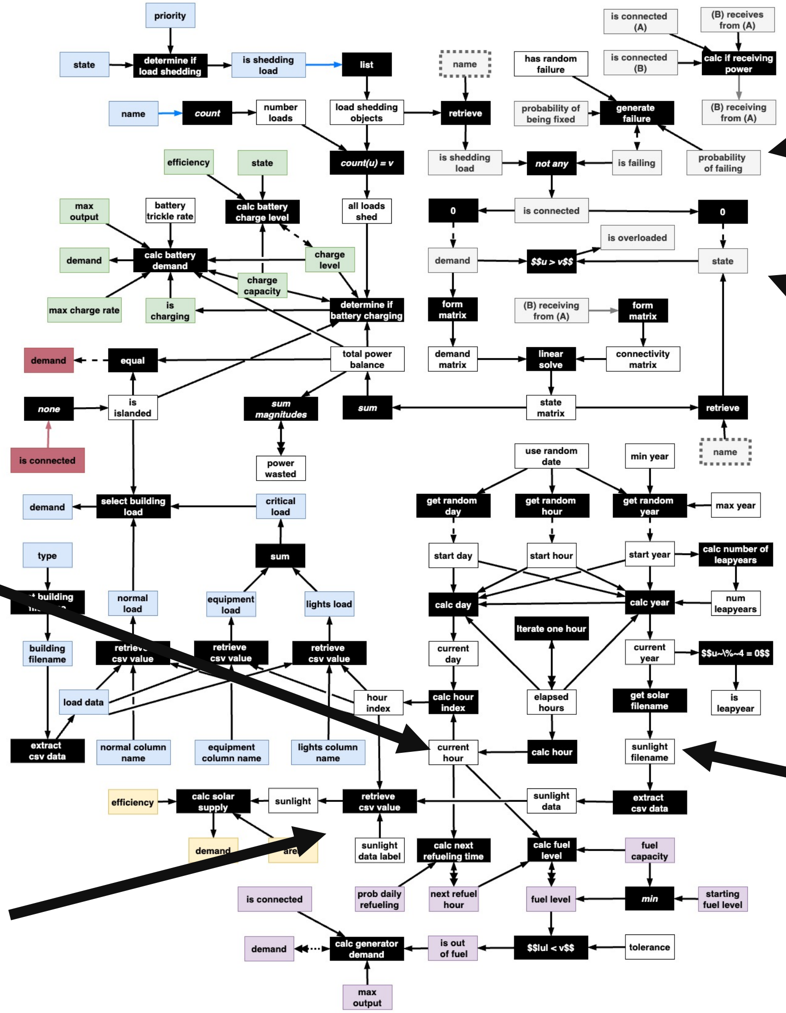
Time/Calendar

Loads/Buildings

Diesel Generators

Photovoltaic Cells

Includes all kinds of simulation and modeling



Stochastic models

Linear equation solvers

Dynamic simulation

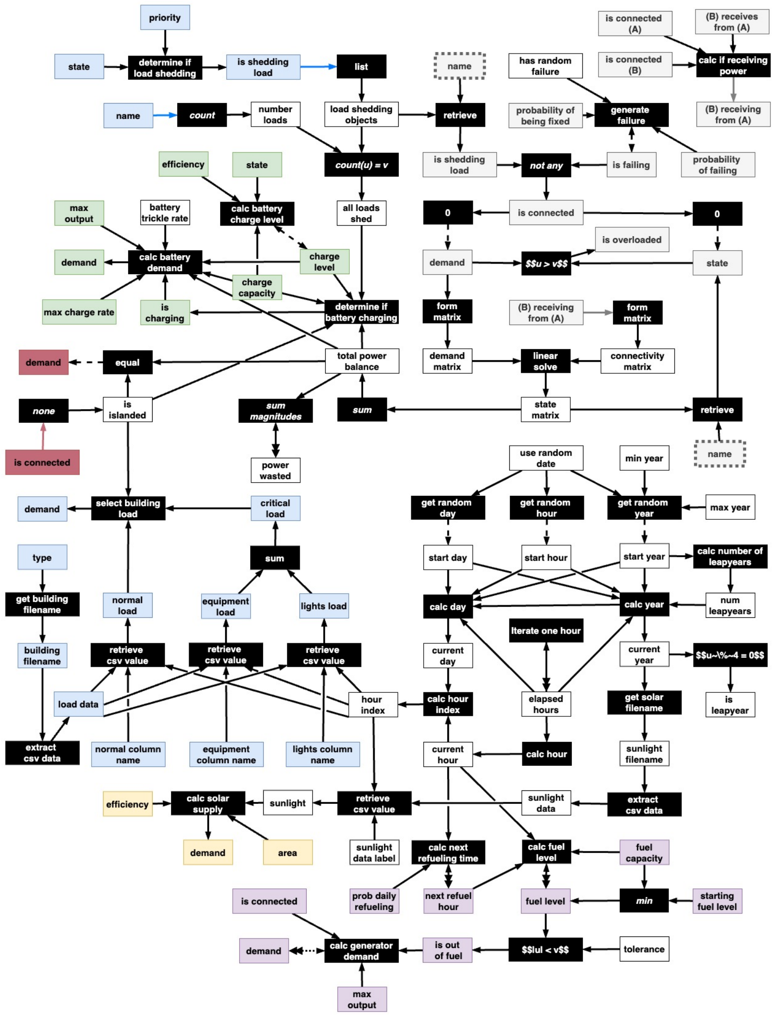
File input/output

Database queries

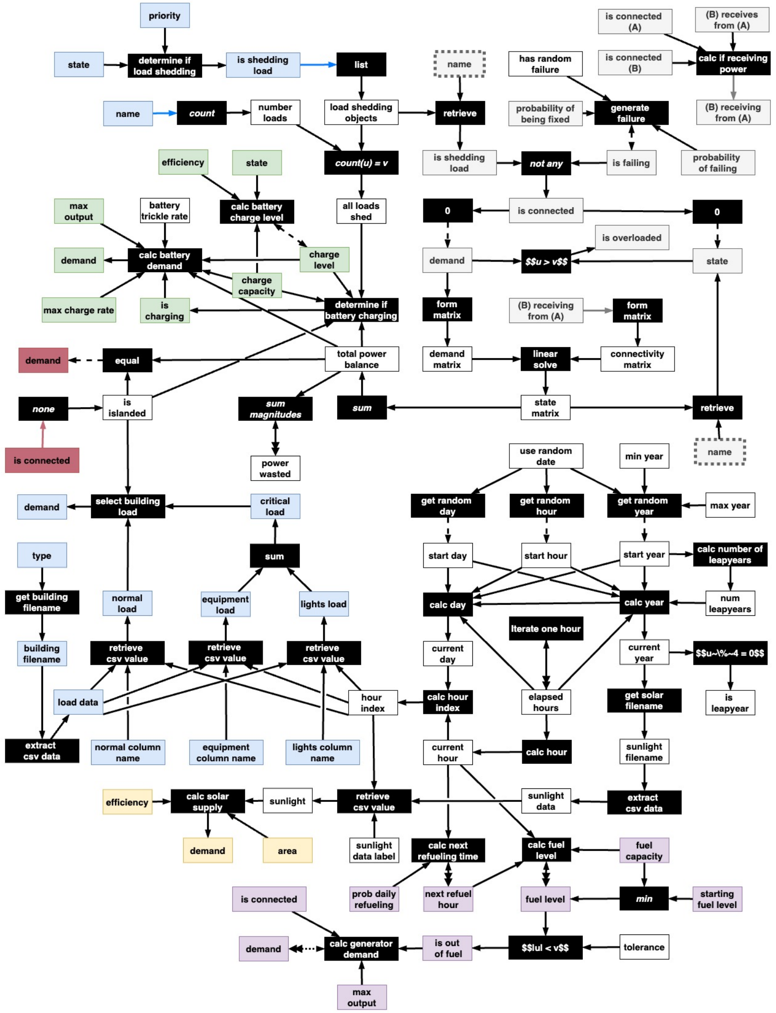
Complexity during searching (creating simulations)

Complexity of the system if fully captured by the constraint hypergraph

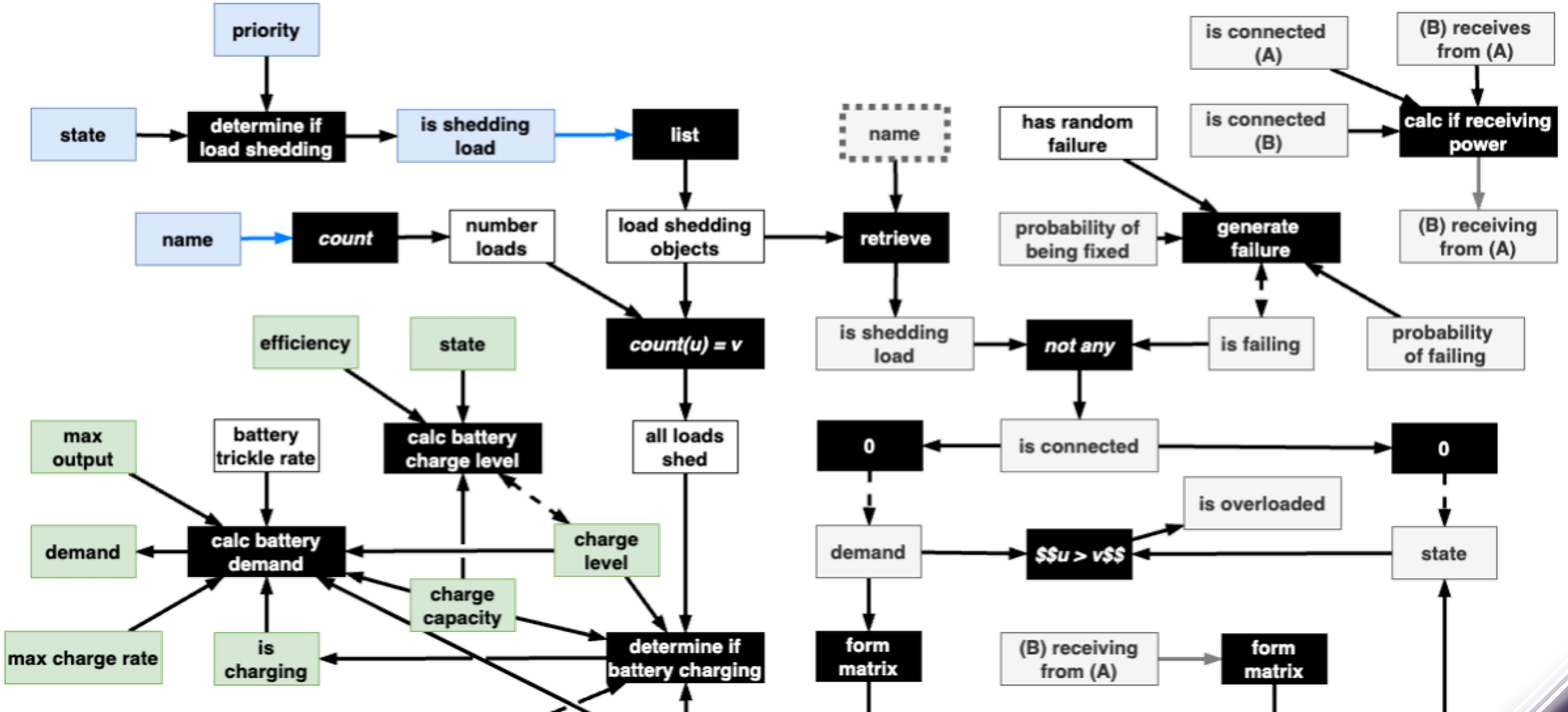
Complexity is mostly an issue when searching, as most pathfinding requires performing the simulation



Machine readable, not human readable



Hard to read for a human means hard to modify



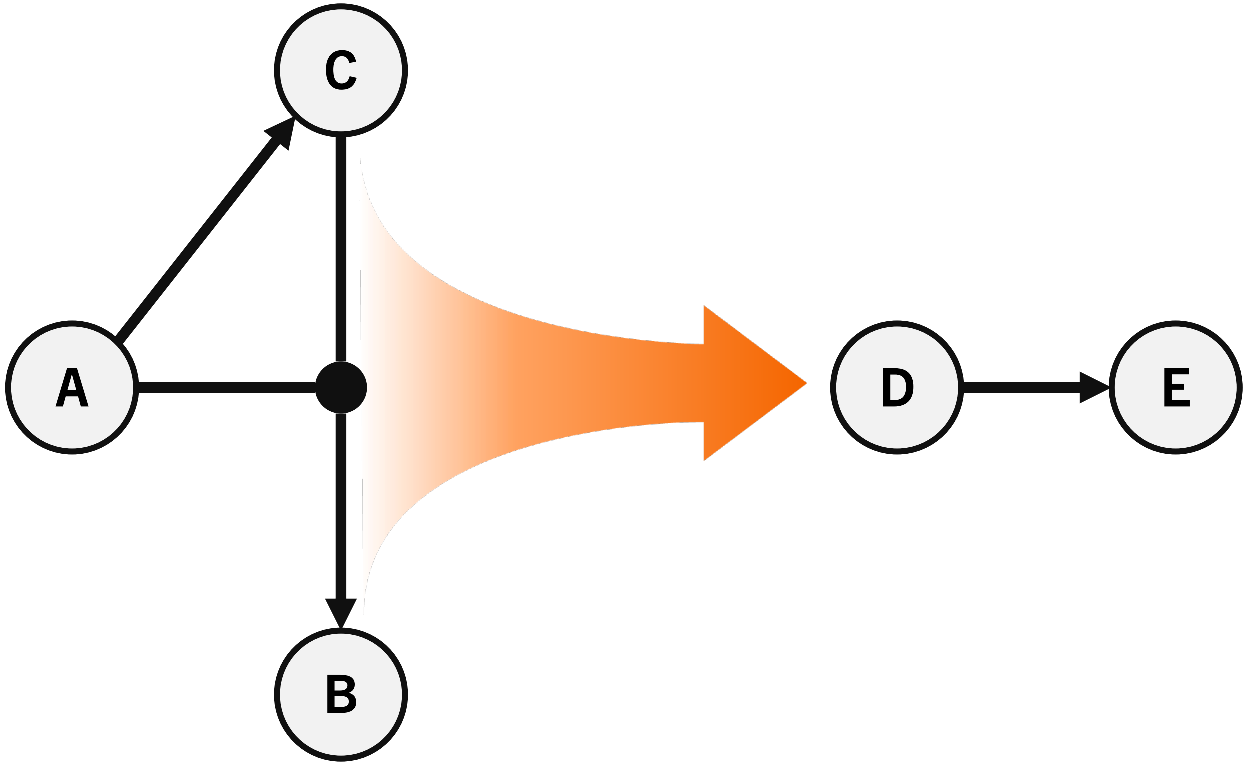
Where does this fit with existing work?

Doesn’t replace modeling

Doesn’t replace software

Doesn’t replace AI

Doesn’t replace science

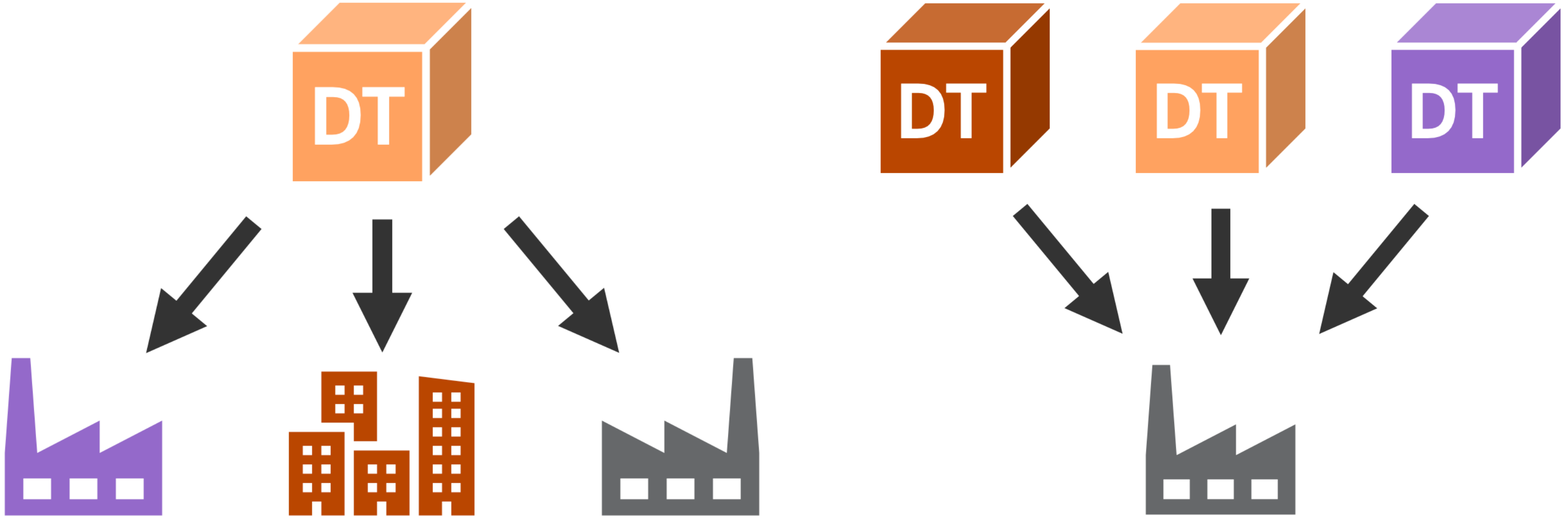


Constraint hypergraphs allow DTs to be composed, redeployed, and adapted without loss of meaning

True value is providing observations of system information—both measured and simulated

Connects all information together into a meaningful network

Wraps all possible simulations into a single model

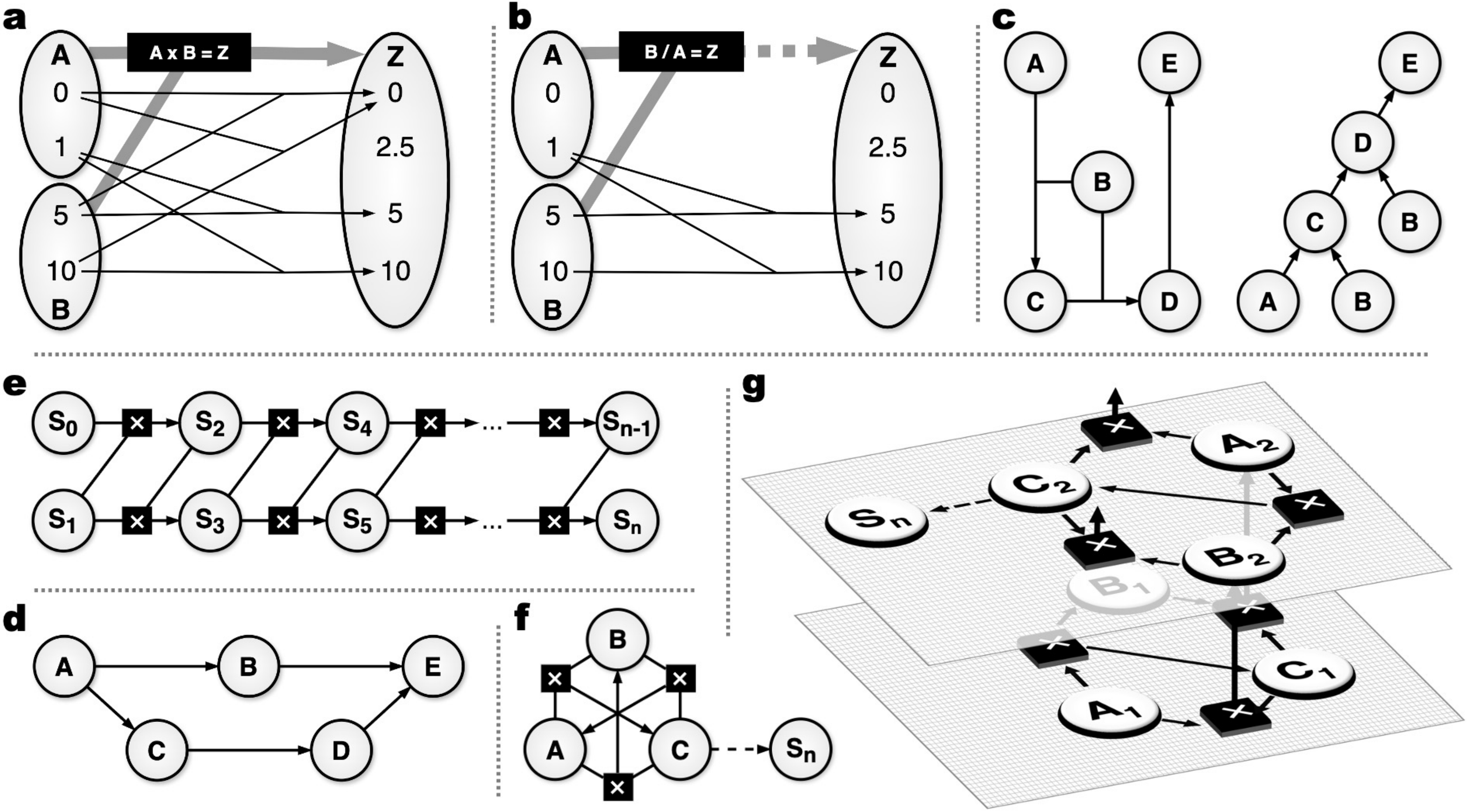


More information available at our GitHub repository

CHg

Please reach out to jhmrrs@clemson.edu

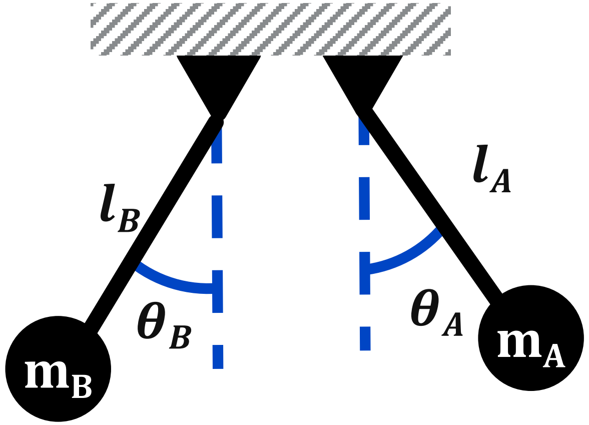
PLMC CEDAR Clemson Engineering Design Applications and Research CLEMSON


a
A
0
1
5
10
B
A x B = Z
Z
0
2.5
5
10
b
A
0
1
5
10
B
B / A = Z
Z
0
2.5
5
10
c
A
B
C
D
E
E
D
C
B
A
B
e
S0
S1
S2
S3
S4
S5
Sn-1
Sn
×
...
d
A
B
C
D
E
f
B
A
C
Sn
×
g
A2
C2
B2
Sn
B1
C1
A1
×

Emergent behavior arises out of composed functions

$$\ddot{\theta}_A = -\frac{g}{l_A}\sin\theta_A$$

$$\ddot{\theta}_B = -\frac{g}{l_B}\sin\theta_B$$

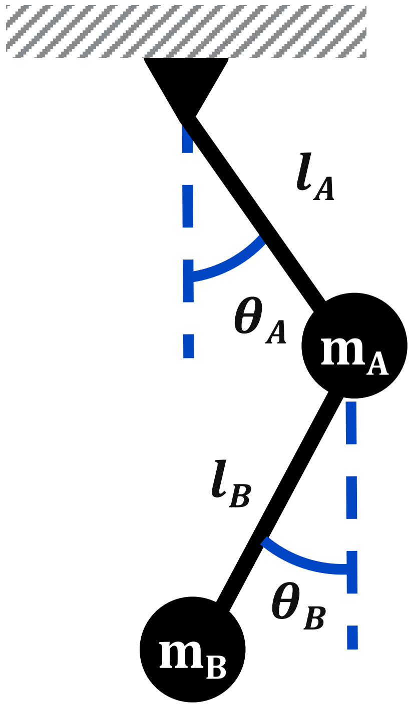


System is fully defined

When the models do not compose, the emergent behavior is not provided

$$\ddot{\theta}_A = -\frac{g}{l_A}\sin\theta_A$$

$$\ddot{\theta}_B = -\frac{g}{l_B}\sin\theta_B$$

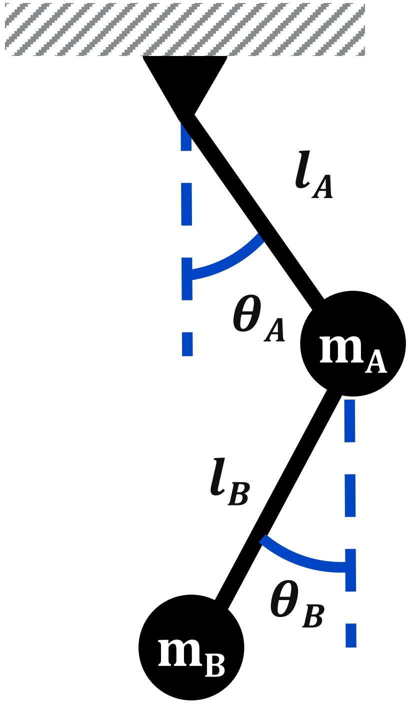


System cannot be represented as desired with the given nodes

Composition must be provided by showing the relationships between nodes

$$\ddot{\theta}_A = -\frac{g}{l_A}\sin\theta_A$$

$$\ddot{\theta}_B = -\frac{1}{l_B}(\ddot{x}_B\cos\theta_B + \sin\theta_B(\ddot{y}_B + g))$$



New model shows how the bobs can be joined

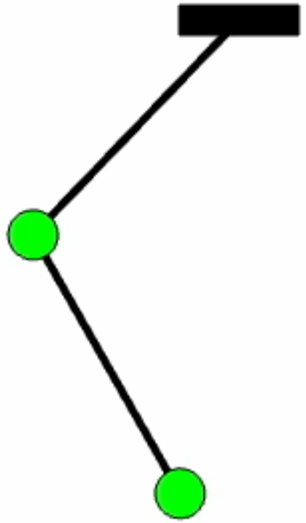
When composition is given, the emergent behavior can be fully realized

$$\ddot{\theta}_B = -\frac{1}{l_B}(\ddot{x}_B \cos\theta_B + \sin\theta_B(\ddot{y}_B + g))$$

where:

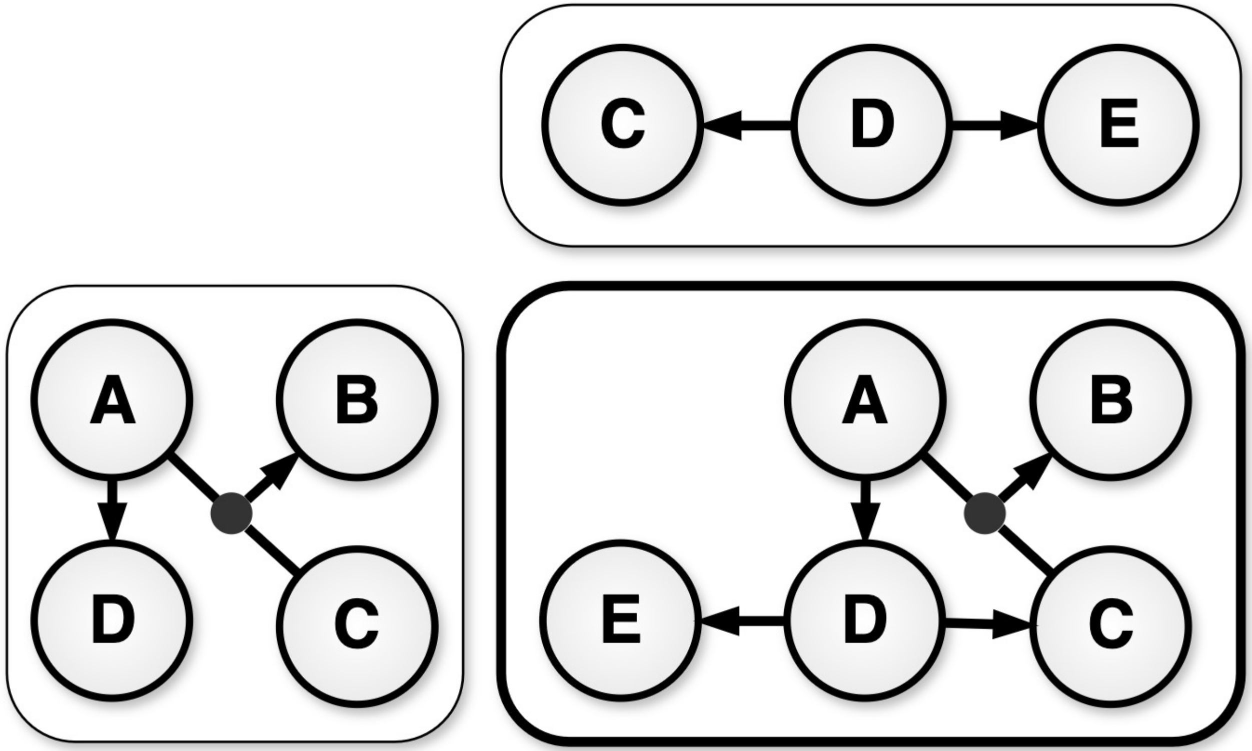
$$\ddot{x}_B = l_A(\alpha_A \cos\theta_A - \omega_A^2 \sin\theta_A)$$

$$\ddot{y}_B = l_A(\alpha_A \sin\theta_A + \omega_A^2 \cos\theta_A)$$



Solving for $\ddot{x}_B$ and $\ddot{y}_B$ using terms from the graph for A gives you the emergent behavior

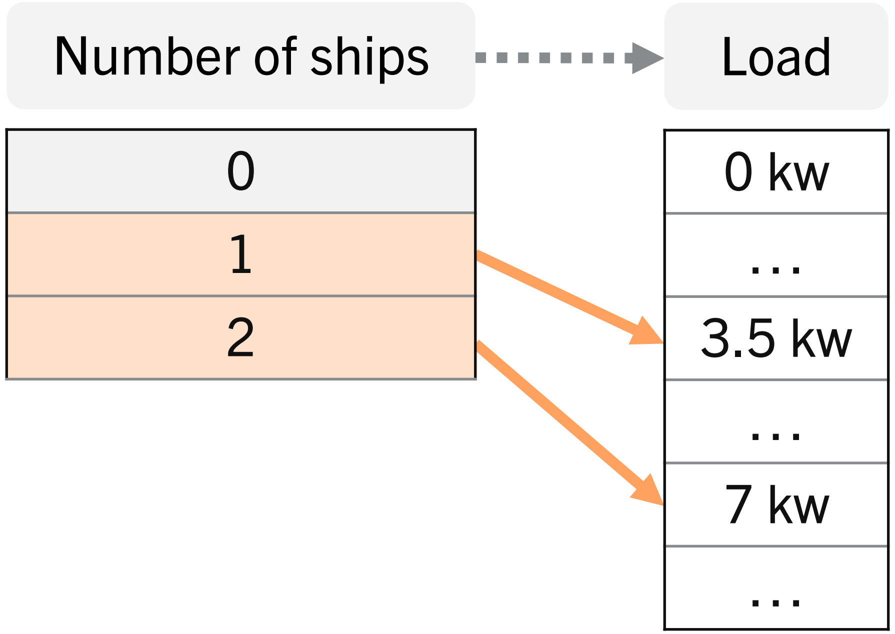
Emergent behaviors only emerge if modeled



Model only shows what you know—unseen behaviors can still exist

Properties of Constraint Hypergraphs: Handle model validity frames

Models with given validity frames can be described by functions that map from a *subset* of a node's values



Number of ships

Load

0

1

2

0 kw

…

3.5 kw

…

7 kw

…

Morris, J., Mocko, G., & Wagner, J. (2025). Unified System Modeling and Simulation via Constraint Hypergraphs. Journal of Computing and Information Science in Engineering, [In Press]