Universal Systems Simulation via Constraint Hypergraphs with Applications to Digital Twins

John Morris

6 November 2025
PhD Defense
Dept. of Mechanical Engineering | Clemson University

PLMC

CEDAR
Clemson Engineering Design Applications and Research

CLEMSON

Biographical Sketch

(2018) Married

BS: Mech Engr, BYU (2021)

(2021) Started as PLM Applications Engineer

Birth of first son (2021)

(2023) Birth of second son

MS: Mech Engr, Clemson (2023)

(2024) Collaboration with NPS

Visiting Researcher at NIST (2025)

(2025) Birth of third son

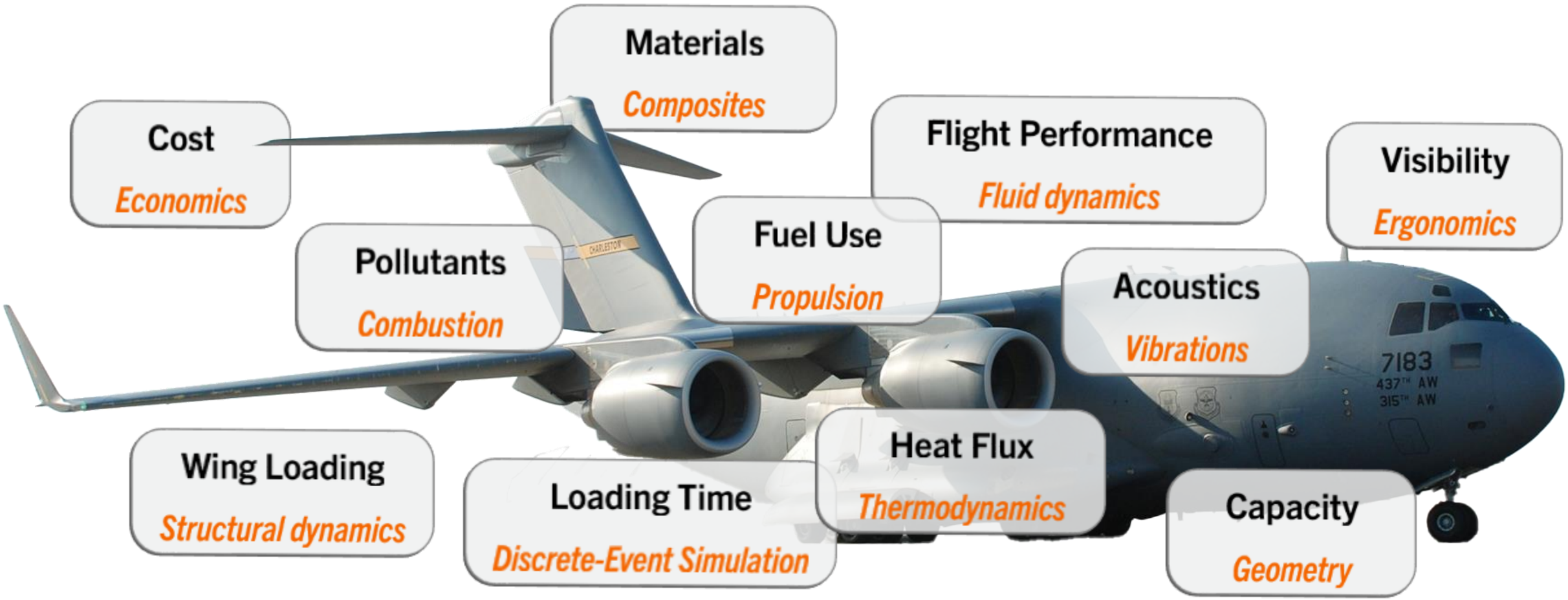
PhD: Mech Engr, Clemson (2025)

(2026) Collaboration with Alan Turing Institute

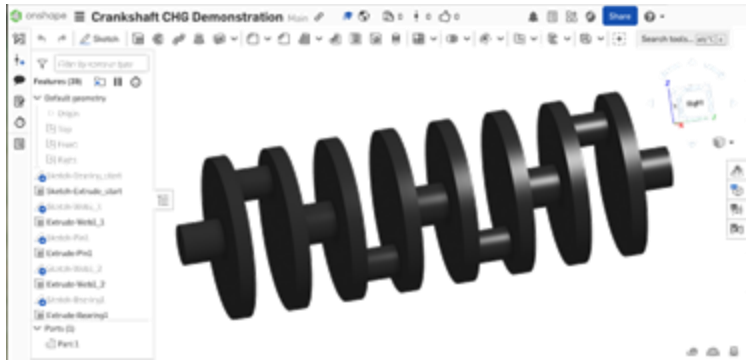
Post-doctoral fellowship with NPS (2026)

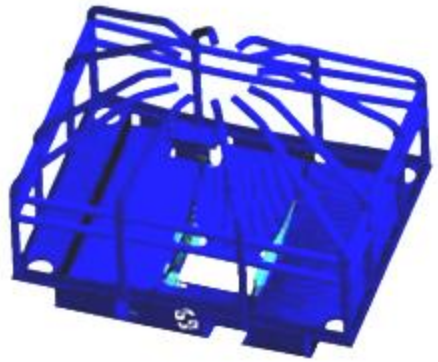
Science is the work of describing the worlds around us

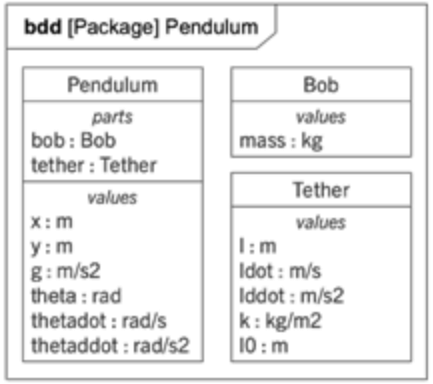
Knowing information about complex systems require models across many domains

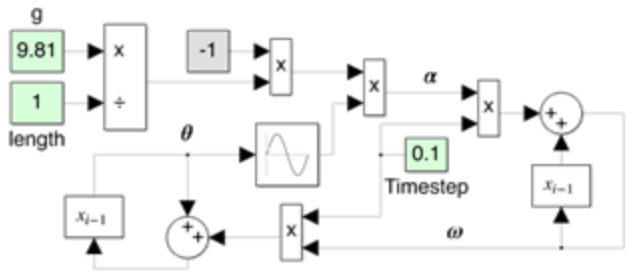


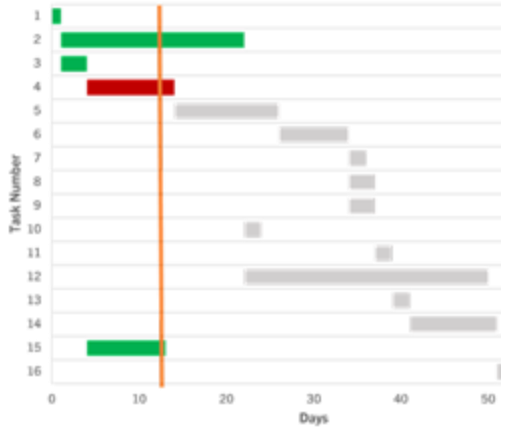
Most modeling frameworks have limited universality and descriptiveness

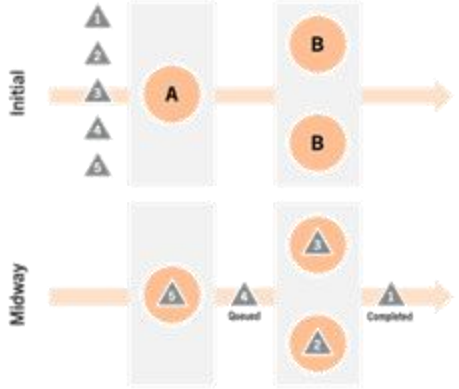


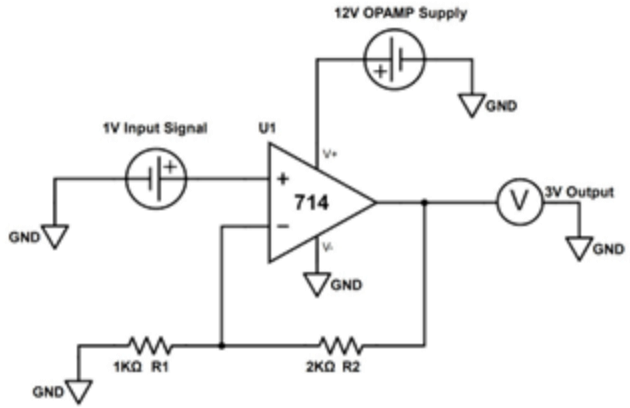


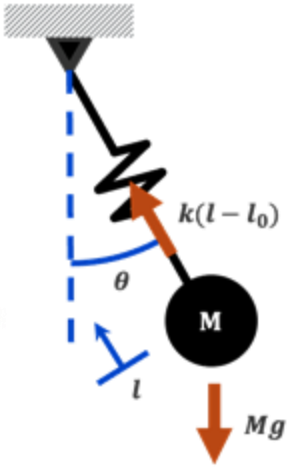






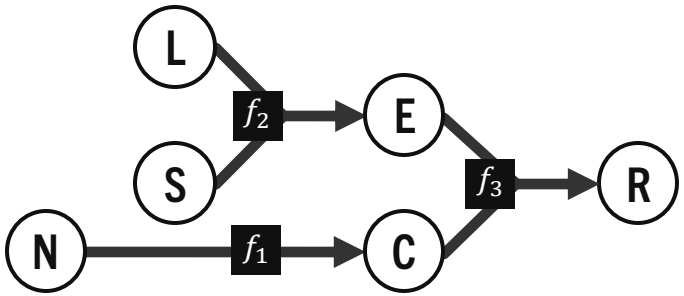


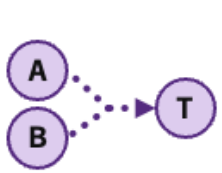


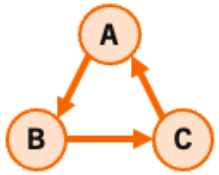


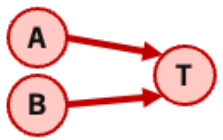
Constraint hypergraphs are a framework for universal, declarative systems modeling

CHG Definition

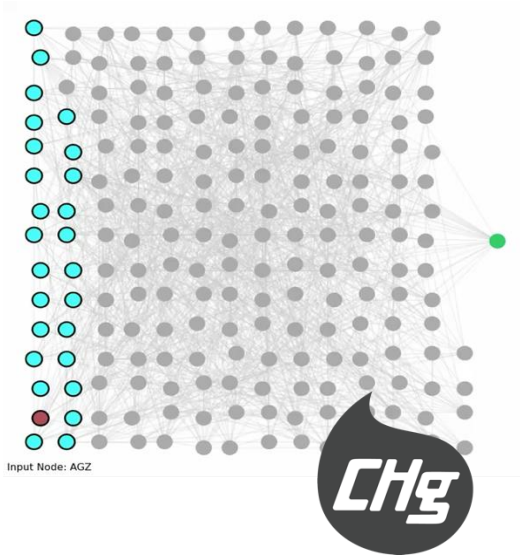




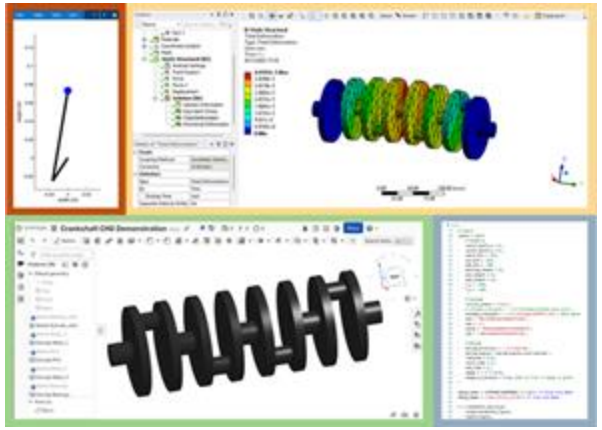


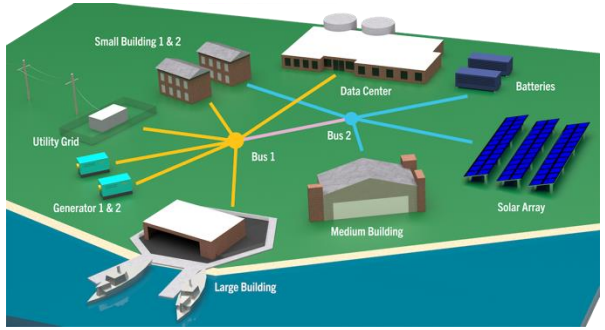


Declarative Simulation



Applications





Introduction to CHGs

Universal System Simulation via Constraint Hypergraphs

Information is what we can distinguish about a system

Economy (L/km)

Color

Fuel Capacity (L)

Name

“A system is a collection of variables” –Ross Ashby

A state is data that must be temporally consistent

Economy (L/km)
10
11
12

Color
Blue
Gray
Tan

Fuel Capacity (L)
100,000
115,000
130,000

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

We describe systems by either observations or simulations

Economy (L/km)
10
11
12

Color
Blue
Gray
Tan

Fuel Capacity (L)
100,000
115,000
130,000

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

Observations are measurements we make of reality

Economy (L/km)
10
11
12

Color
Blue
Gray
Tan

Fuel Capacity (L)
100,000
115,000
130,000

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

Simulations are predictions we make based on relationships

Economy (L/km)
10
11
12

Color
Blue
Gray
Tan

Fuel Capacity (L)
100,000
115,000
130,000

(Database lookup)

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

Model relations are algebraic functions that describe behavior

$f_{n\to e}(\text{Name}) \to \text{Economy}$

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

Economy (L/km)
10
11
12

$f_{n\to c}(\text{Name}) \to \text{Capacity}$

Fuel Capacity (L)
100,000
115,000
130,000

J. C. Willems, "The Behavioral Approach to Open and Interconnected Systems," IEEE Control Systems Magazine (2007)

Functions can have multiple arguments

$f_{n \to e}(\text{Name}) \to \text{Economy}$

$f_{n \to c}(\text{Name}) \to \text{Capacity}$

$f_{ce \to r}(\text{Capacity} \times \text{Economy}) \to \text{Range}$

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

Economy (L/km)
10
11
12

Fuel Capacity (L)
100,000
115,000
130,000

Flight Range (km)
10,000
11,000
12,000

Functions compose to form greater functions

Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

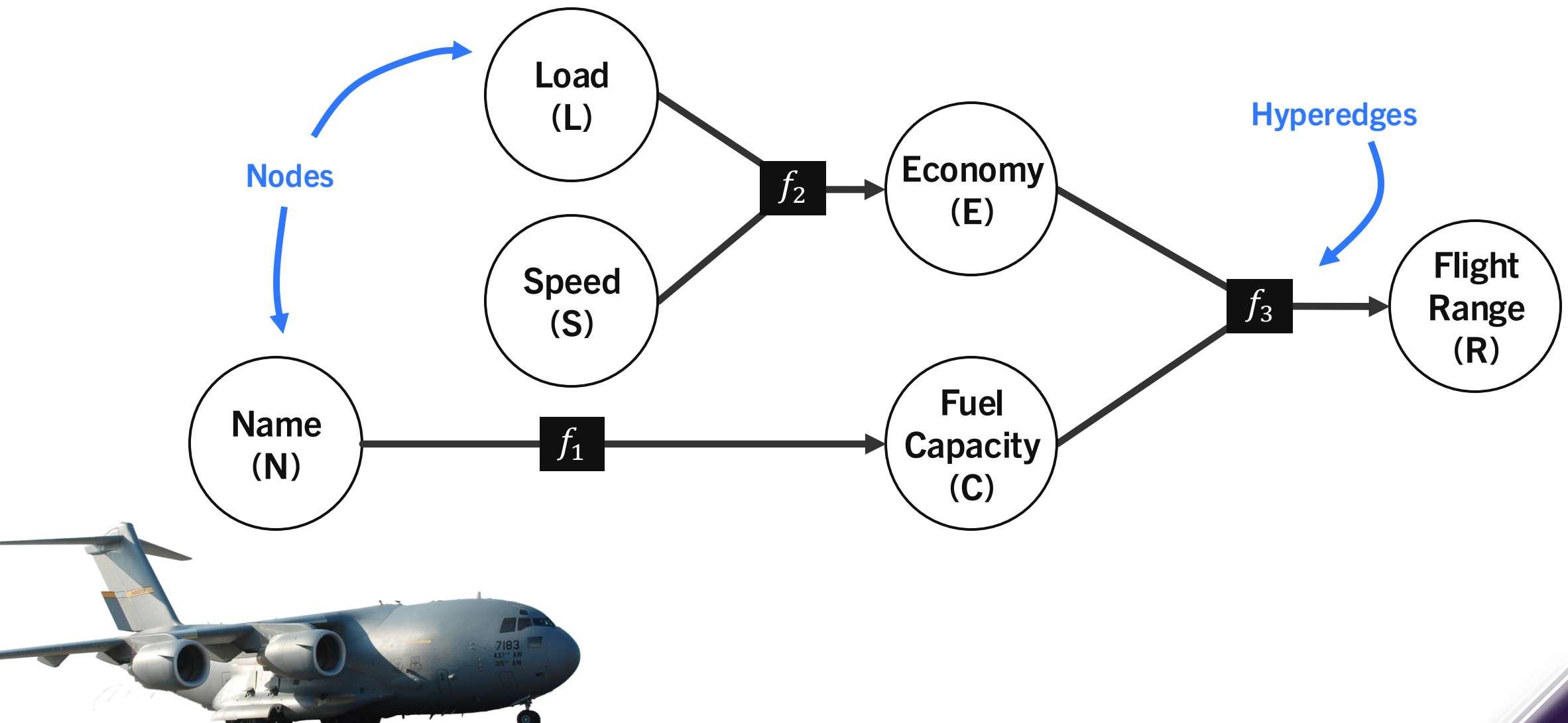
$$f_{n\to r}(\text{Name}) \to \text{Range} = f_{ce\to r}(f_{n\to e}, f_{n\to c})$$

Economy (L/km)
10
11
12

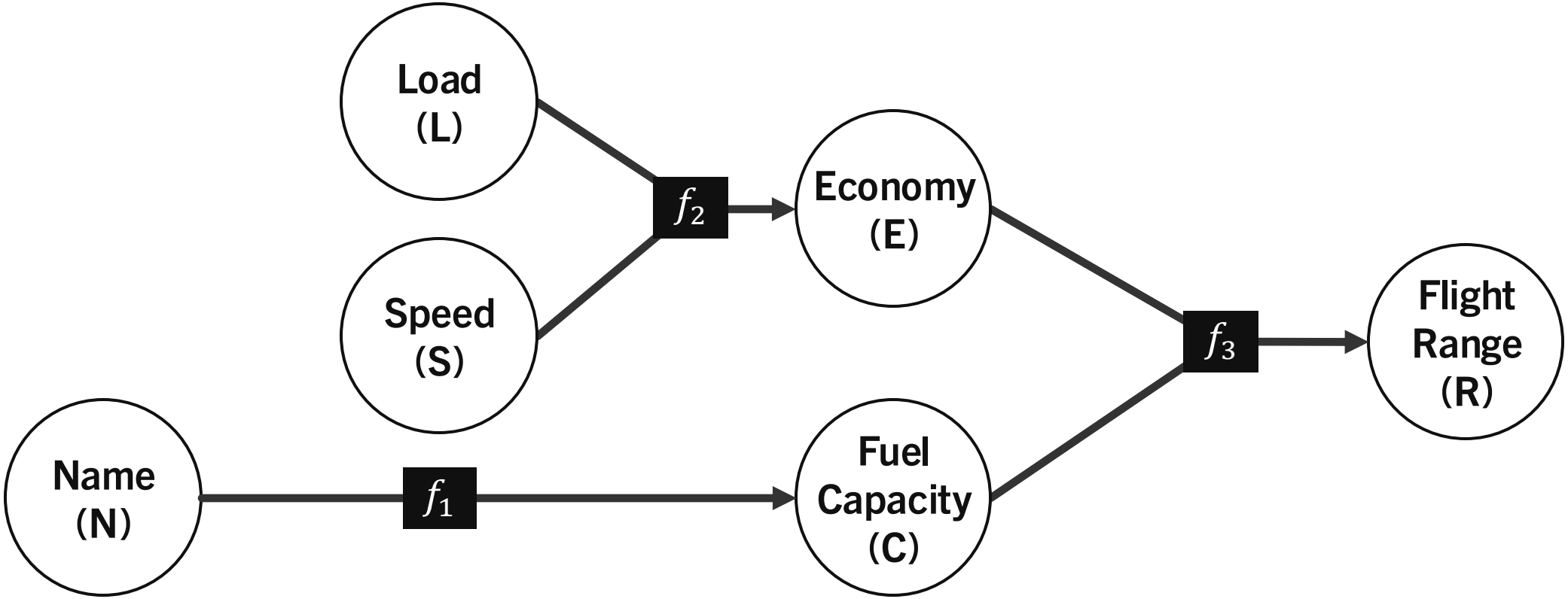
Fuel Capacity (L)
100,000
115,000
130,000

Flight Range (km)
10,000
11,000
12,000

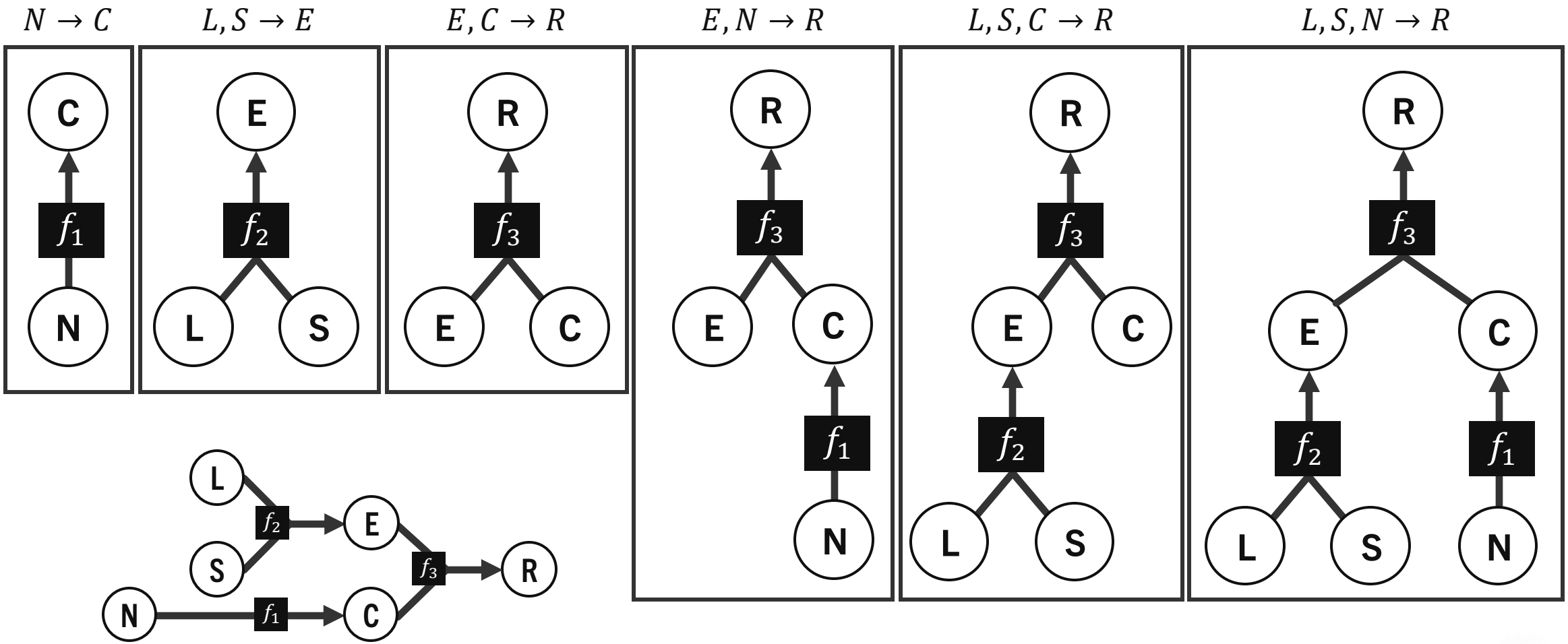
We can represent all models and variables as a constraint hypergraph (CHG)



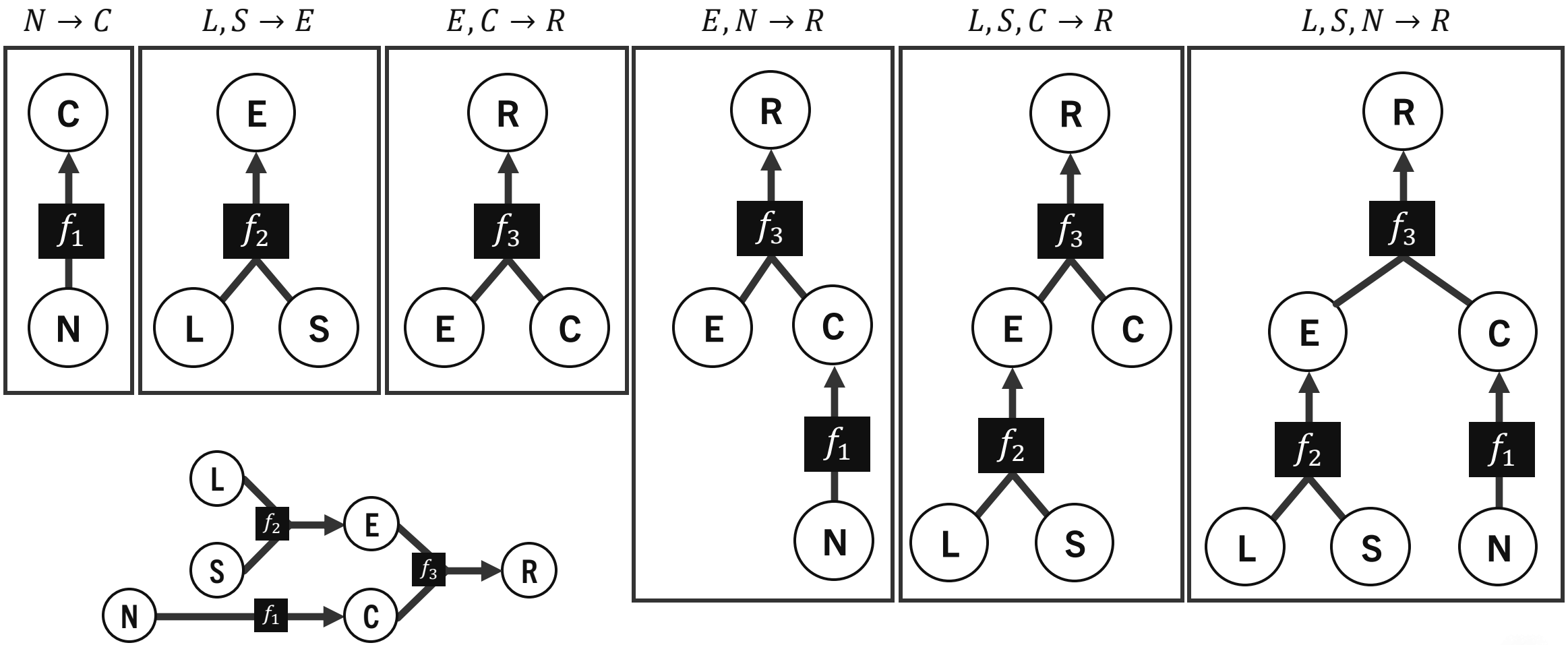
Every path in a CHG is a simulation



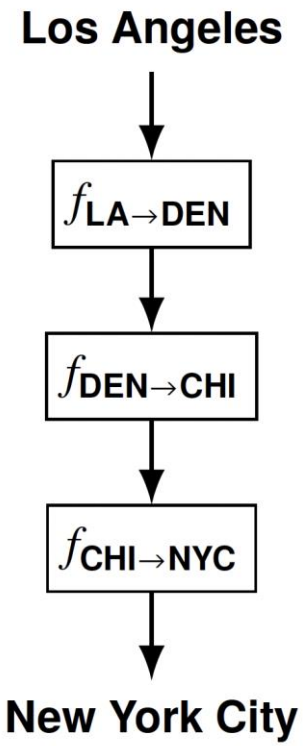
Every path in a CHG is a simulation



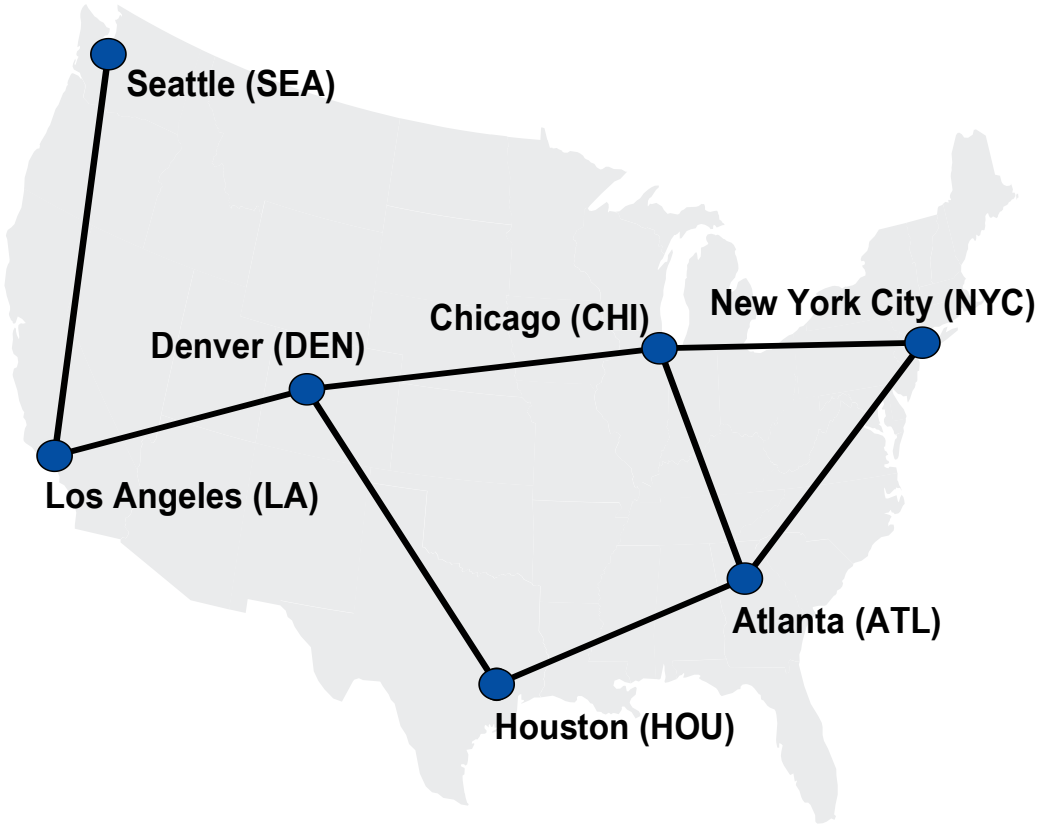
Embedded behavior (paths) enables declarative simulation



Declarative models capture the semantics of the system

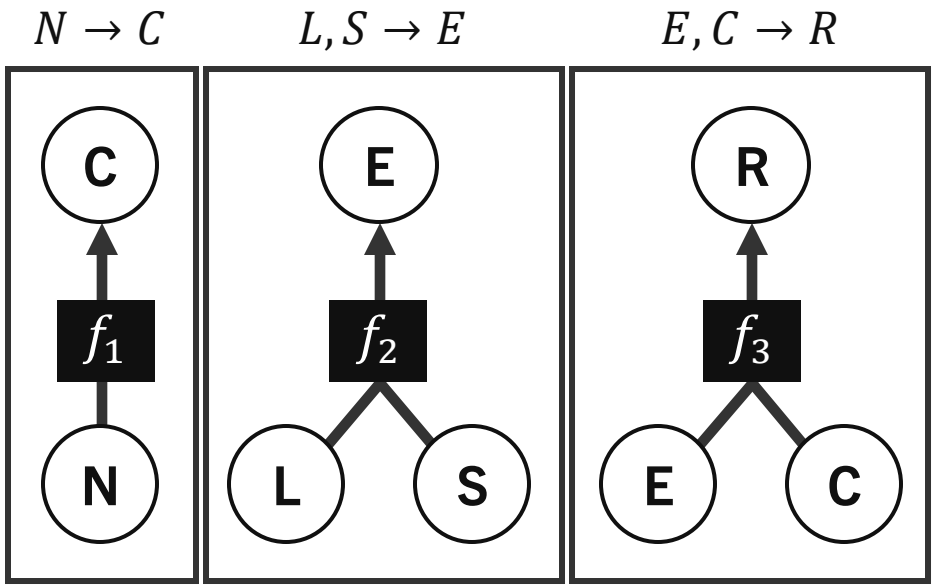


Imperative

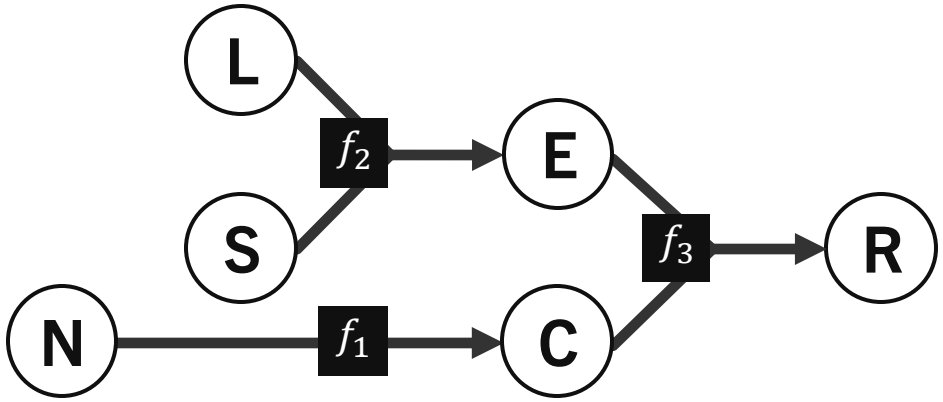


Declarative

Declarative simulation in a CHGs is pathfinding

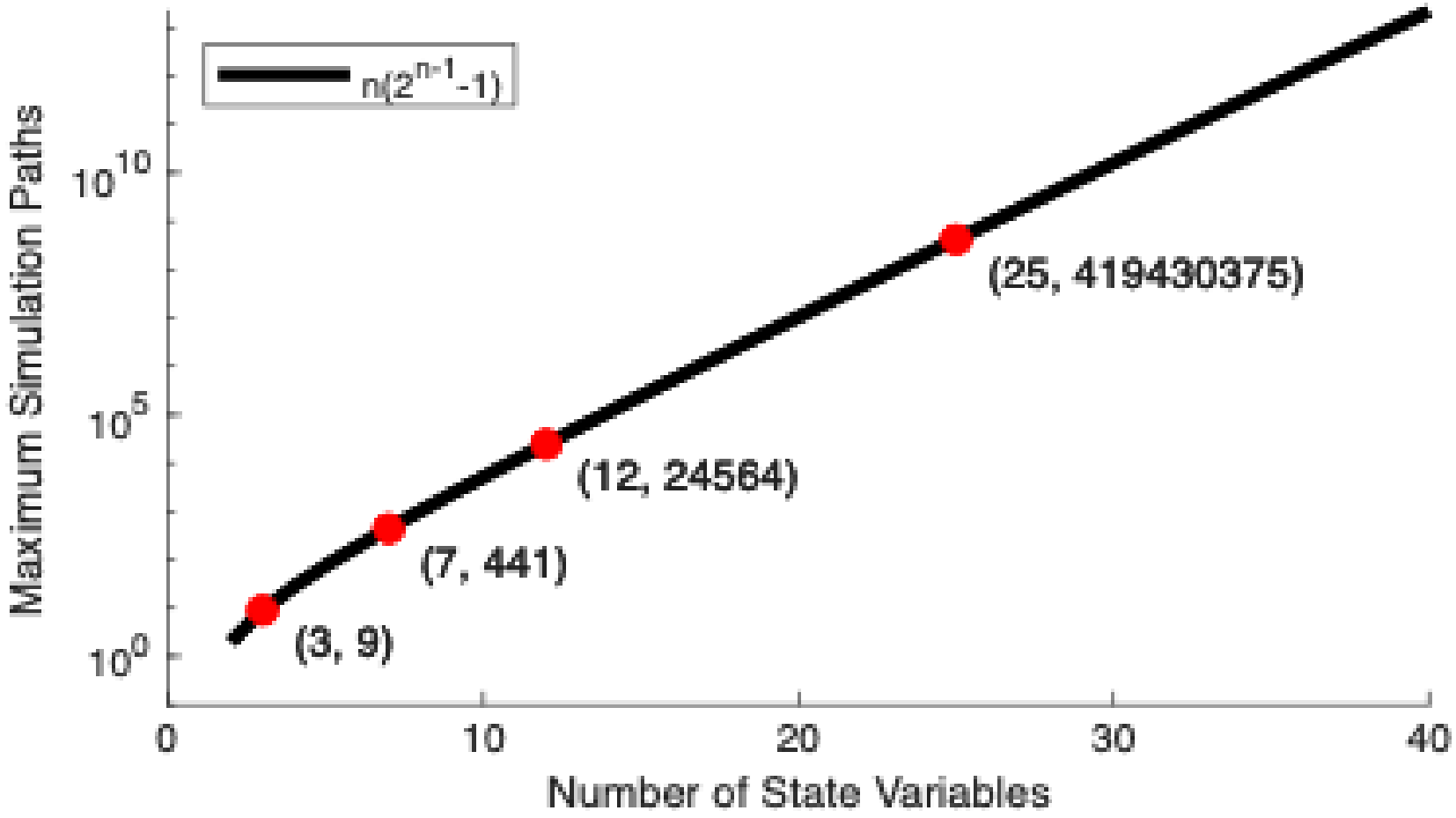


Imperative



Declarative

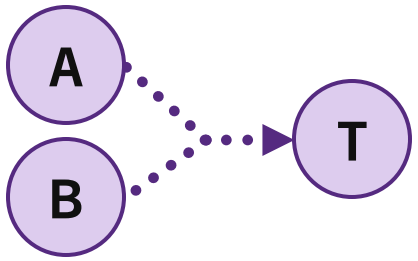
CHGs reduce complexity of expressing simulation paths



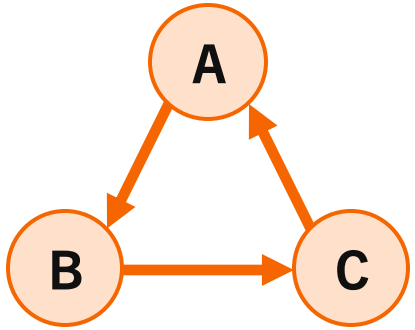
Structure of a CHG

Universal System Simulation via Constraint Hypergraphs

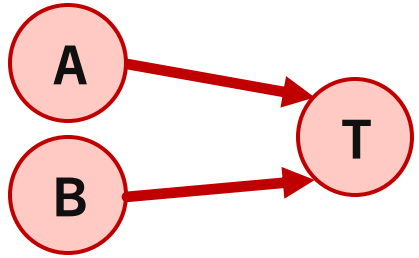
There are three structures in a CHG that a declarative agent needs to process:



Partial Edges



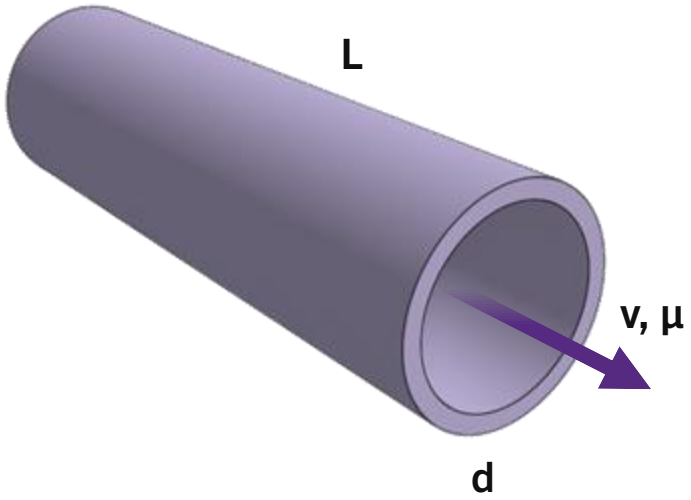
Cycles

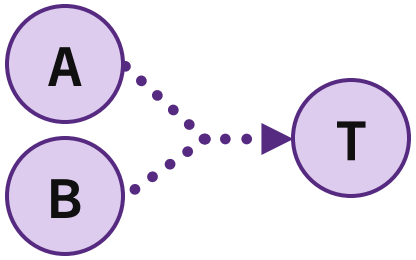


Multi-Edges

Morris, J., Mocko, G., & Wagner, J. (2025). Unified System Modeling and Simulation via Constraint Hypergraphs. *J. Comput. Inf. Sci. Eng.*

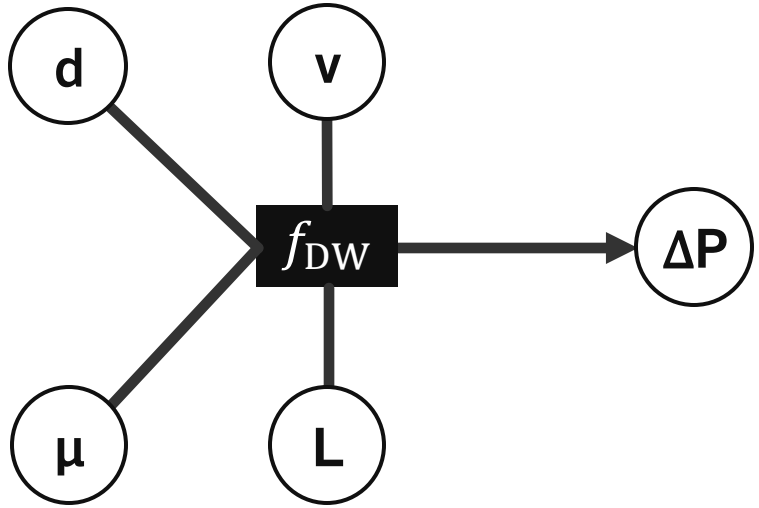
Partial edges occur when we don't have an explicit mapping for every variable

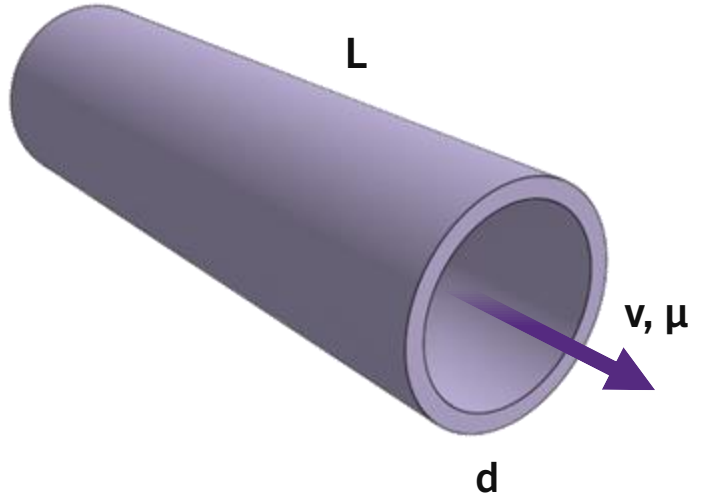


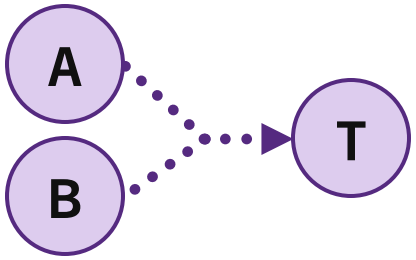


Goal is to calculate pressure loss from fluid flow in a pipe

Partial edges occur when we don't have an explicit mapping for every variable

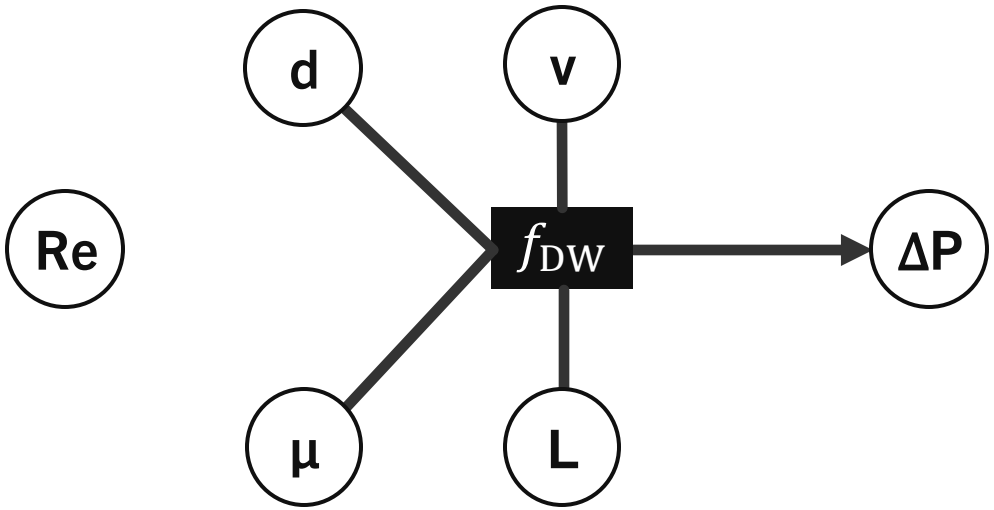


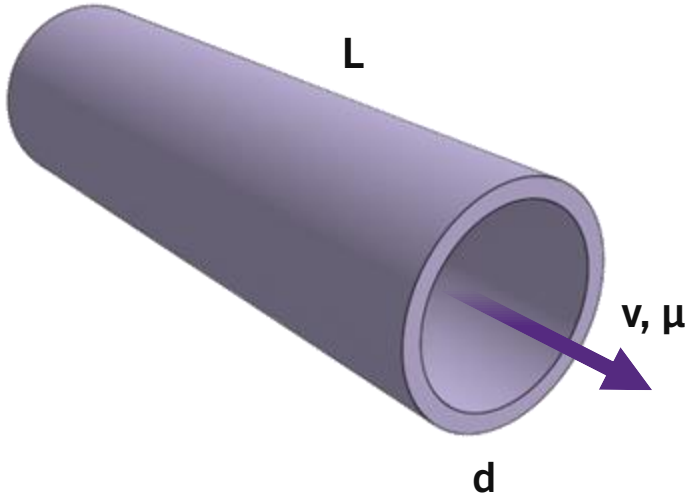


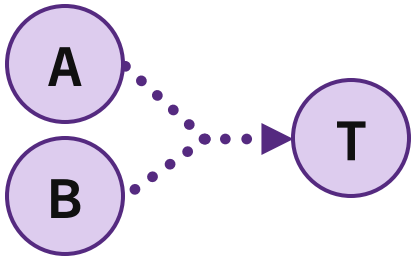


The hypergraph captures the Darcy-Weisbach relationship $\Delta P = \frac{32\mu v L}{\mathrm{d}^2}$

Partial edges occur when we don't have an explicit mapping for every variable

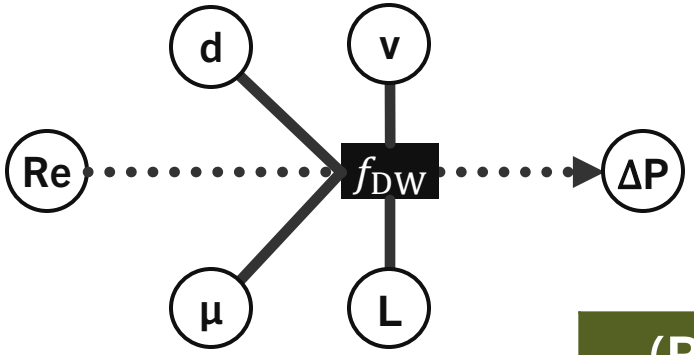


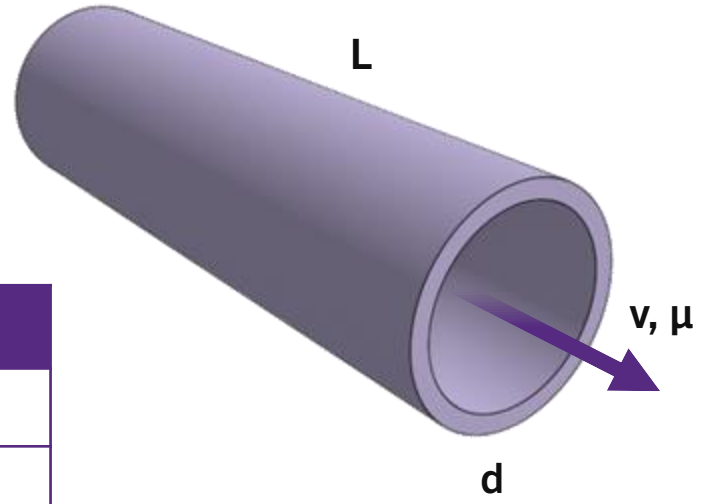




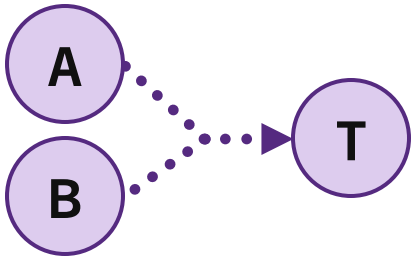
This relationship is only valid if the flow is laminar (Re < 2300)

Partial edges occur when we don't have an explicit mapping for every variable



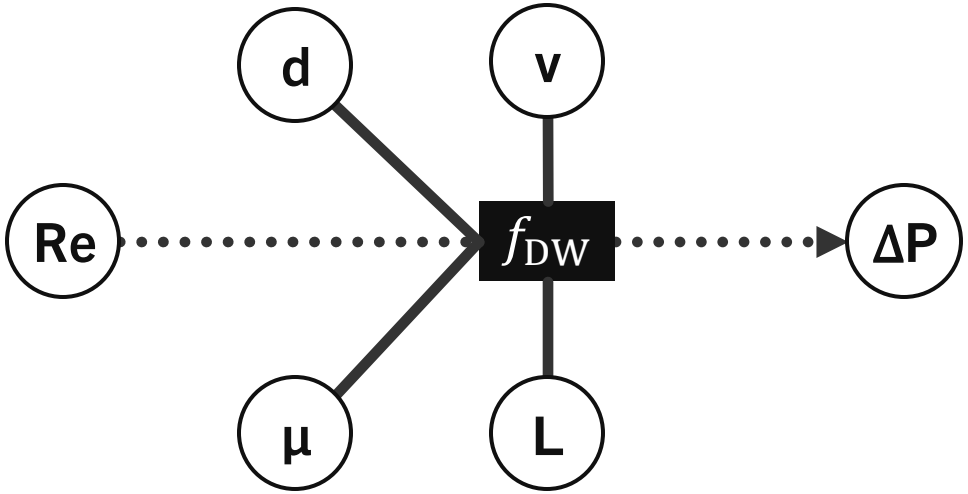


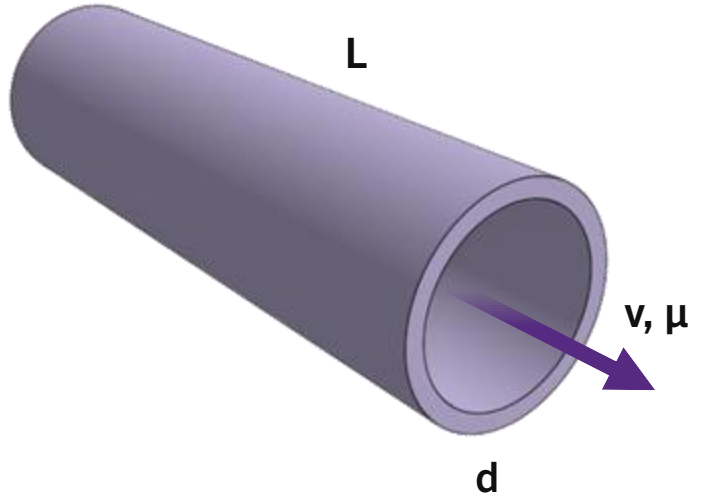
(Re, L, μ, d, v)	f_{DW}	ΔP
…		…
(1000, L, μ, d, v)		0.5
(2000, *L, μ, d, v*)		1
(3000, *L, μ, d, v*)		1.5
…		…

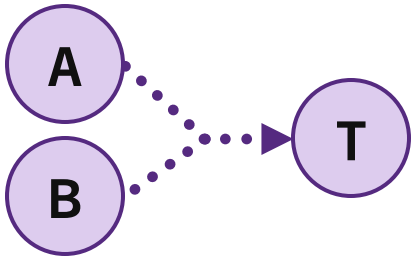


We show this by mapping a subset of the domain

Partiality means that functions don't automatically compose

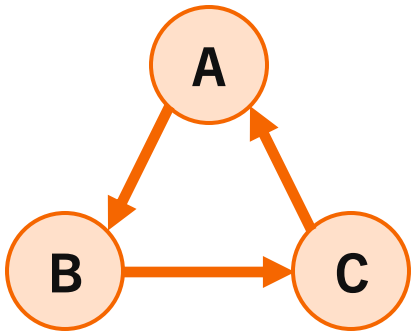






The dotted line shows that our function doesn't handle every value of Re

Cycles indicate behavioral patterns



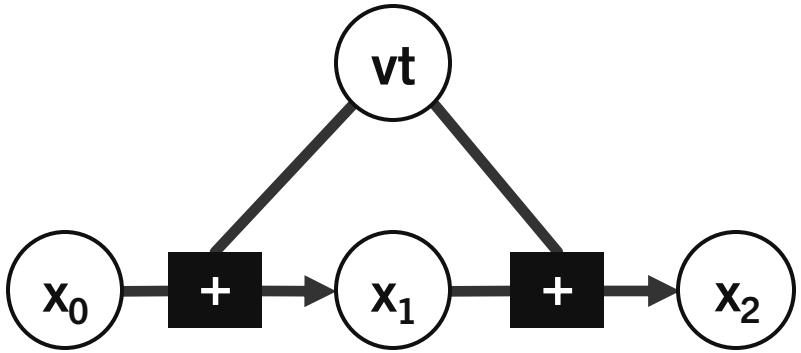
Plane moving at constant velocity

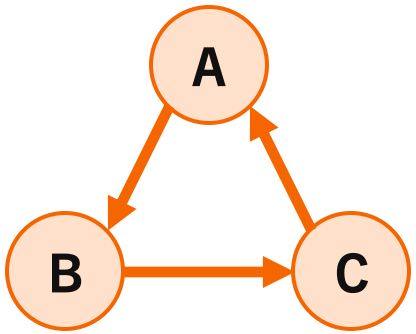
Cycles indicate behavioral patterns

v → x_0

$x_1 = x_0 + vt$ x_1

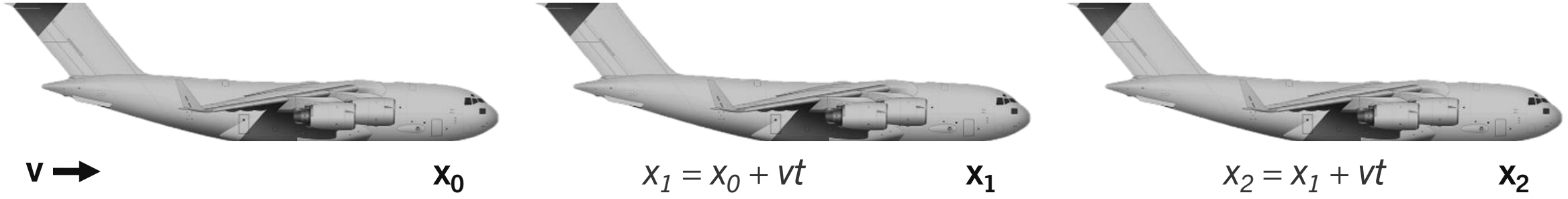
$x_2 = x_1 + vt$ x_2

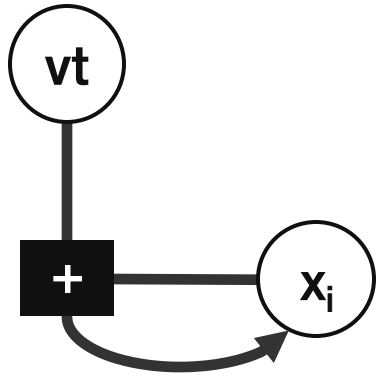


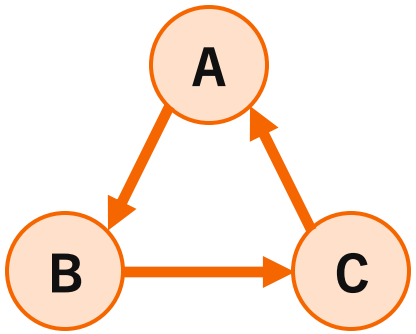


x_0, x_1, and x_2 are all different nodes

Cycles indicate behavioral patterns

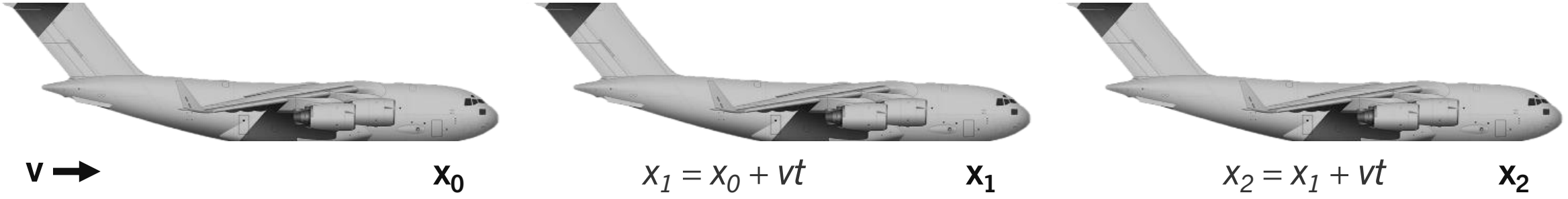


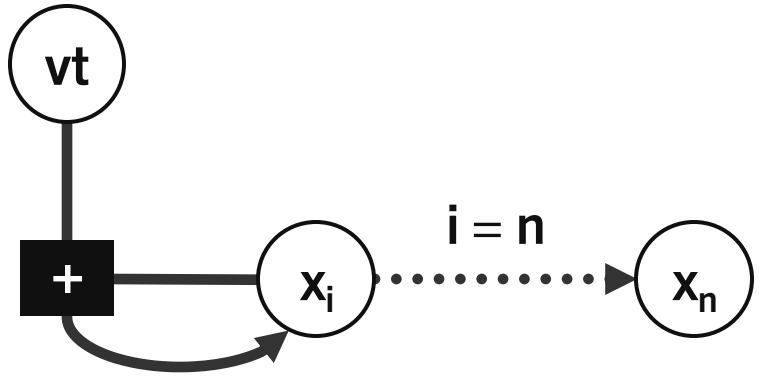


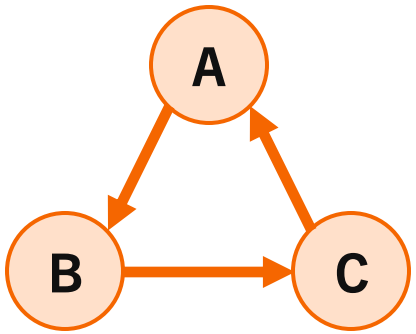


A more expressive way to show this is $x_{i+1} = x_i + vt$ (unraveling)

Unraveling requires an exit condition to be solvable



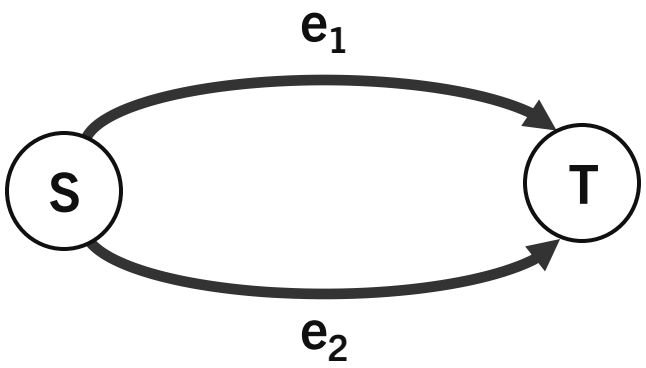




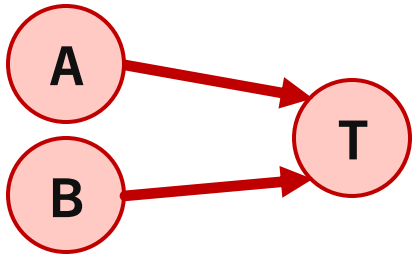
The partial edge shows which edges of x_i are mapped to x_n

Multi-edges indicate competing models

Ansys

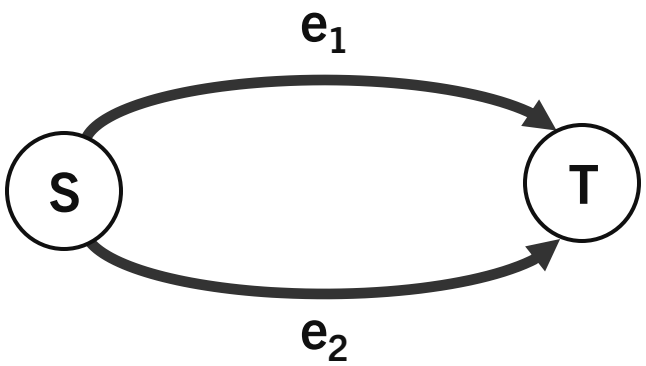


ChatGPT

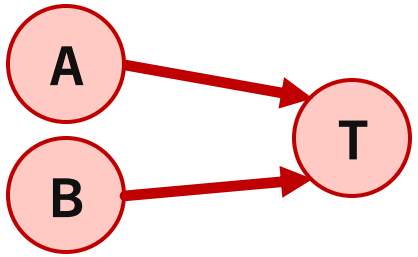


Both edges are viable (“correct”), but only one can be selected for simulation

Ansys



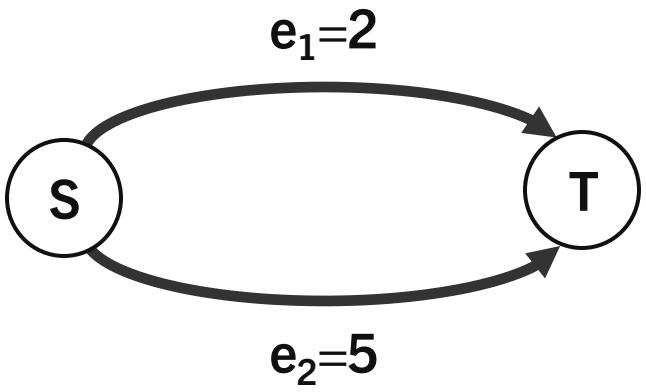
ChatGPT



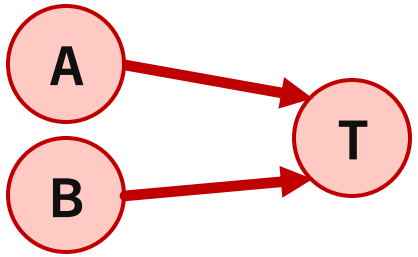
Differences might be on accuracy, computation time, software availability, etc.

Weights allow the agent to *prefer* one edge to another

Ansys



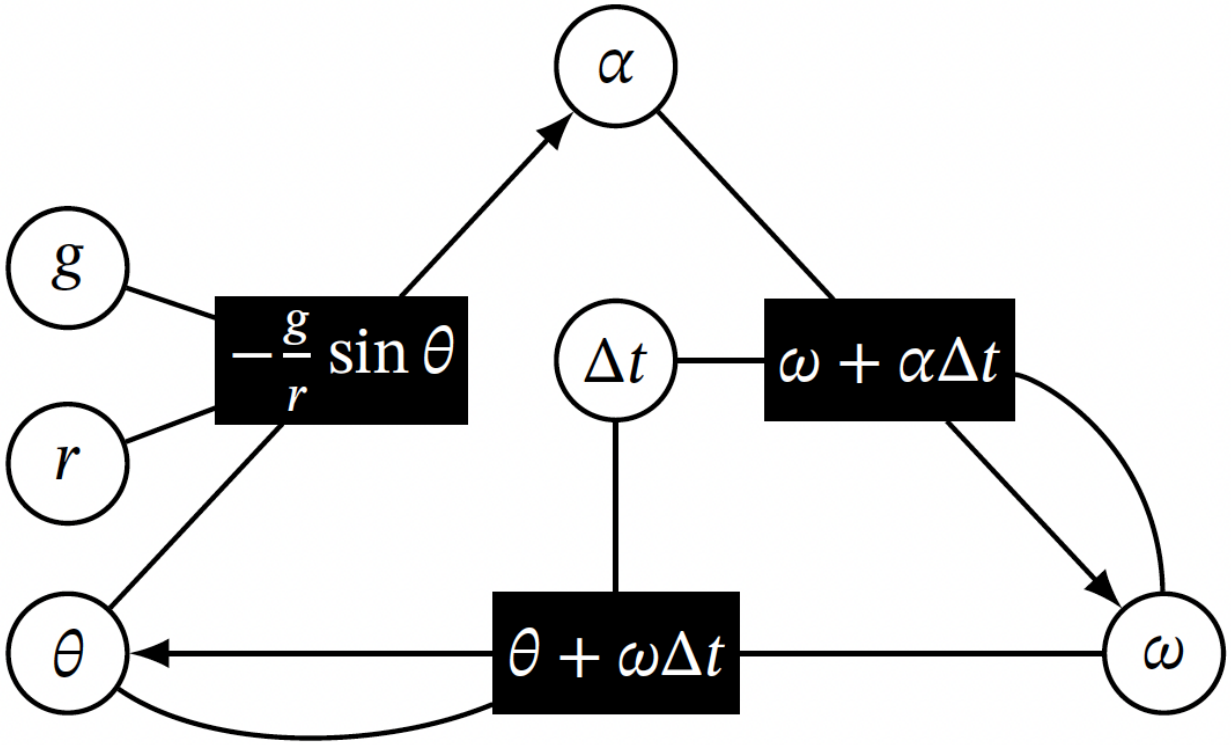
ChatGPT

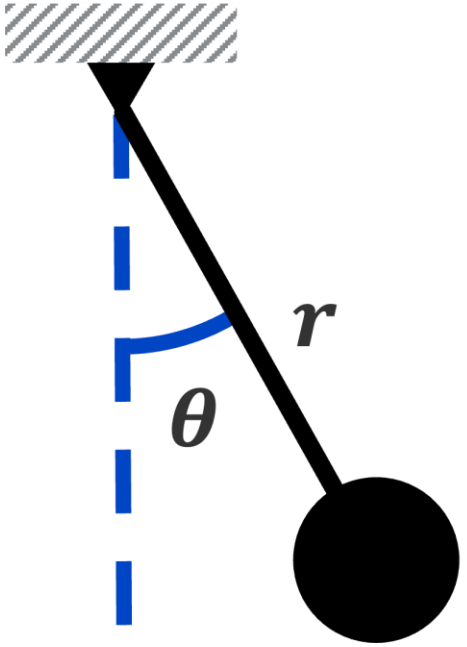


Demonstration

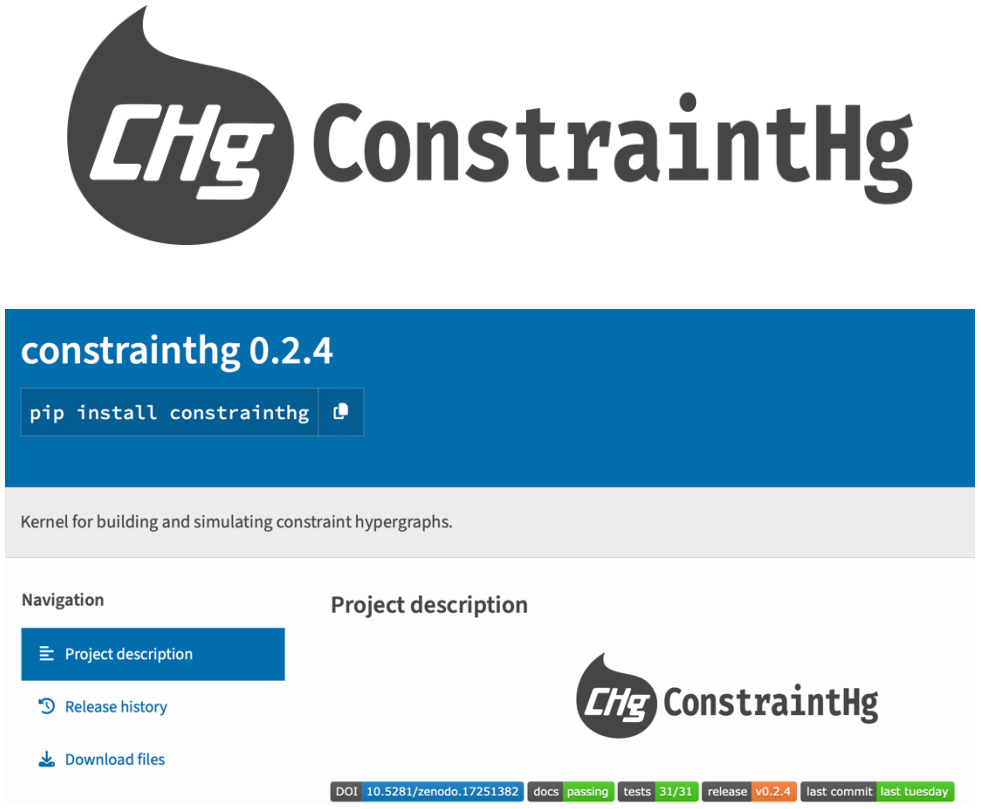
Universal System Simulation via Constraint Hypergraphs

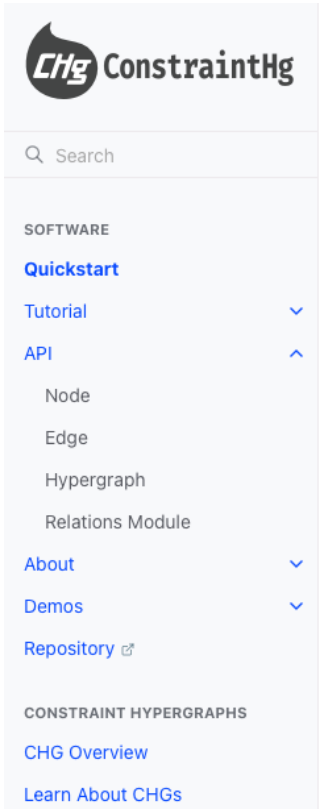
Pendulum System





Declarative simulation provided by ConstraintHg





Quickstart

Use PIP to install ConstraintHg into your Python environment:

```
pip install constrainthg
```

From there you'll want to import the library into your Python script. This is a pretty typical method to use:

```
from constrainthg.hypergraph import Node, Hypergraph
import constrainthg.relations as R
```

Simple Demo

Note that this demo is found in demos/demo_basic.py

Let's build a basic constraint hypergraph of the following equations:

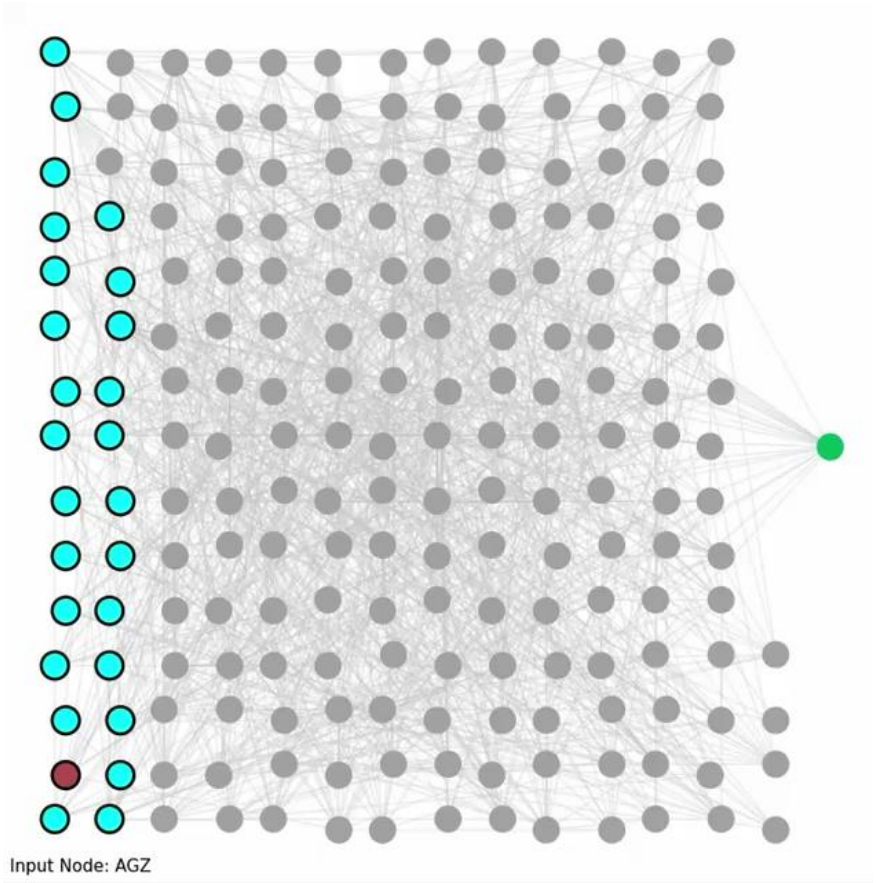
- $A + B = C$
- $A = -D$
- $B = -E$
- $D + E = F$
- $F = -C$

latest

ConstraintHg Functionality

Provides methods for:

- Representing and visualizing CHGs
- Forming cycles, partial edges, and edge weights
- Pathfinding for declarative simulation
  - Extensive logging tools
- Merging CHGs



Process of making a CHG

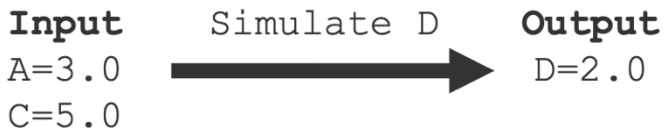
1. Identify system facts (nodes)
Parameters, application variables, API tokens, etc.

2. Form relations (edges) between nodes
Relations show how one node is determined by a set of other nodes

3. Pass to CHG solver
Solver parses the CHG

ConstraintHg

4. Request simulation
Solver simulates requested output by finding the shortest path mapping it to a set a known inputs



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

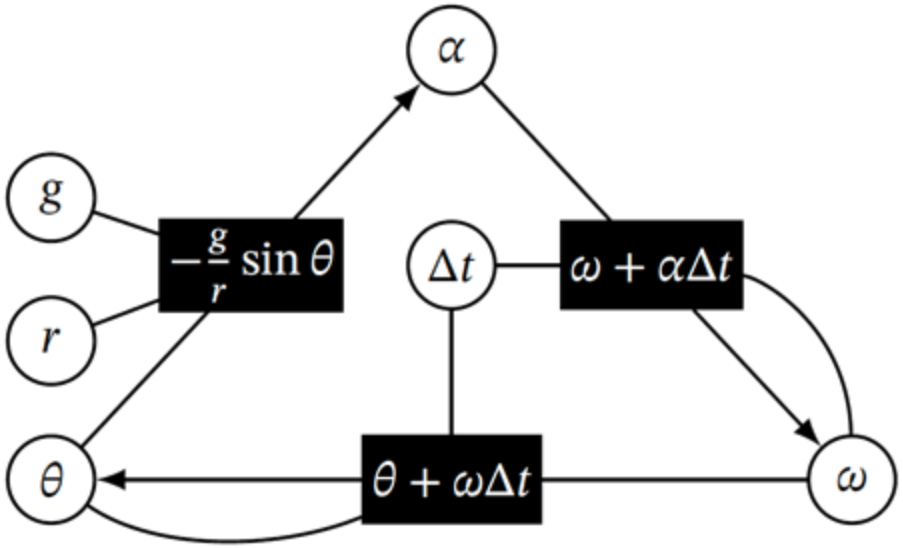
Declarative simulation requires only start & end points

```
hg.solve(
    target='theta’,
    inputs=(
        theta=0.785,
        omega=0.0,
        g=9.81,
        r=0.25,
        delta_t=0.02
    ),
    min_index=2,
)
```

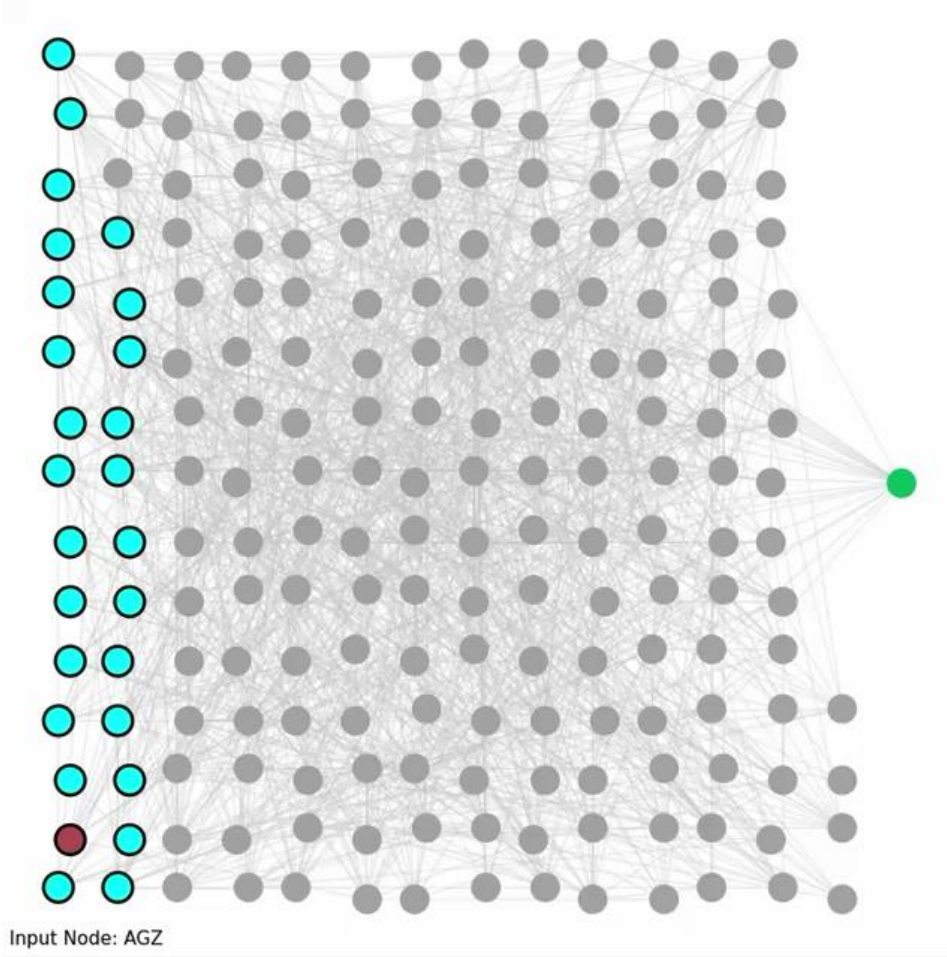
Simulation call

```
└──theta(2)=0.7739
    ├──omega(2)=-0.5547
    │   ├──alpha(2)=-27.74
    │   │   ├──g=9.81
    │   │   ├──theta=0.785
    │   │   └──r=0.25
    │   ├──delta_t=0.02
    │   └──omega=0
    ├──delta_t=0.02
    └──theta=0.785
```

Discovered, unraveled simulation



Limitation: slow searchability (cannot be presolved)

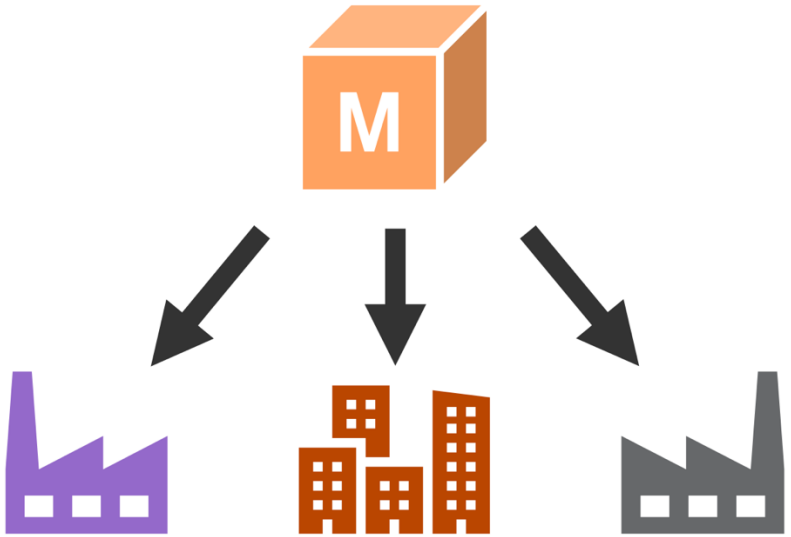


Applications of CHGs

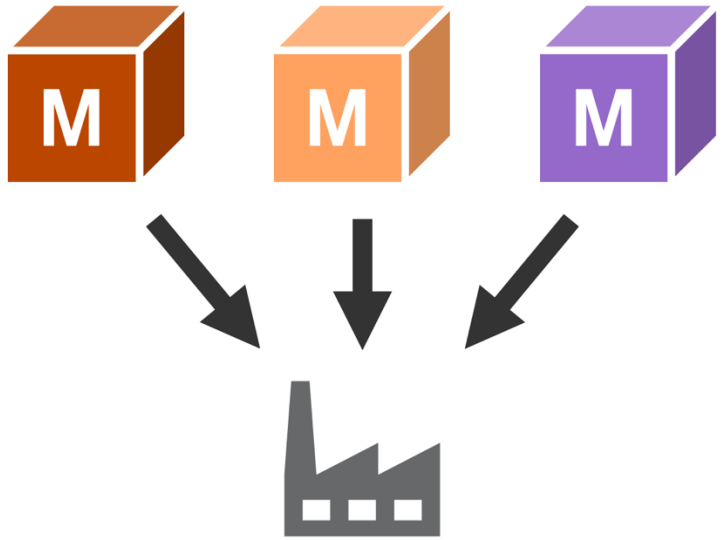
Universal System Simulation via Constraint Hypergraphs

CHGs provide platforms for model-based engineering and digital twins

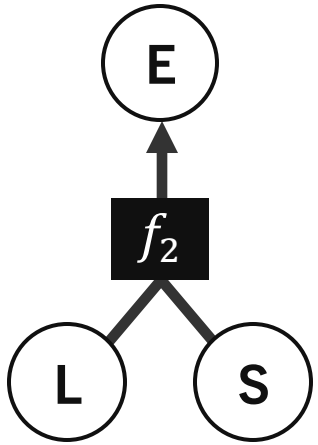
Maintains information in different contexts



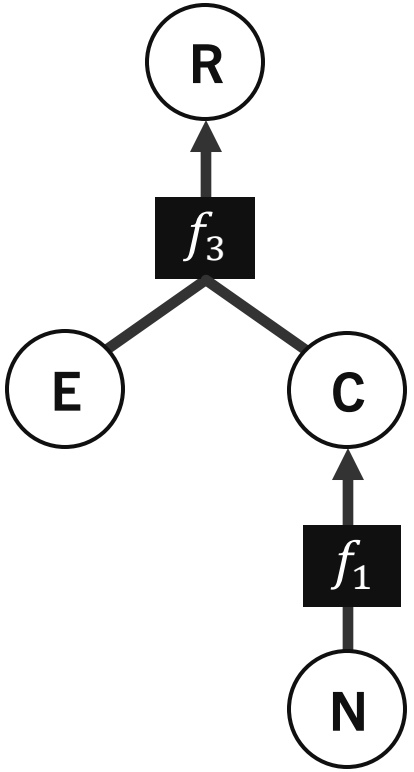
Maintains information with different models



New behaviors can be expressed as new paths in the union of two CHGs

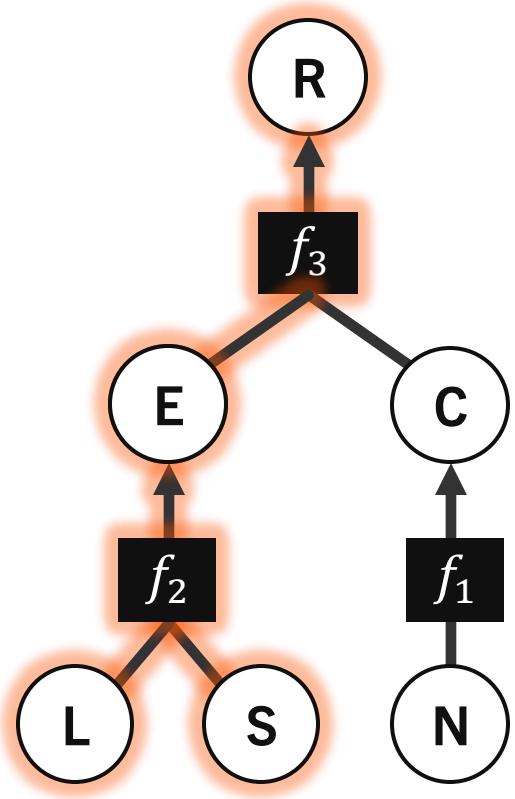


Graph 1



Graph 2

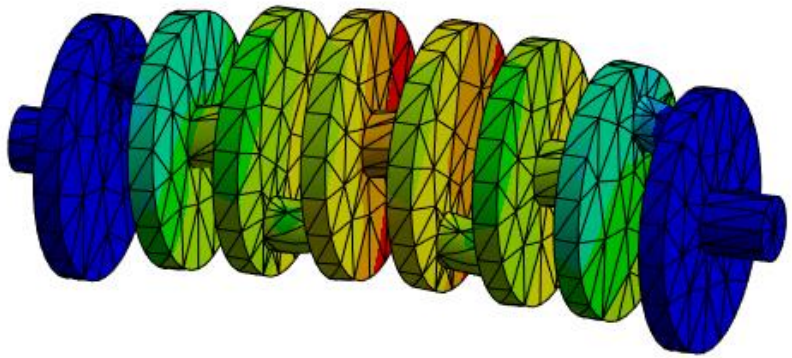
New behaviors can be expressed as new paths in the union of two CHGs



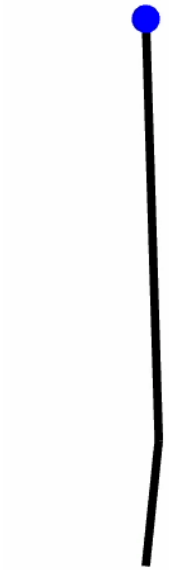
Graph 1 ∪ 2

New behaviors include simulation paths that go across calculating software

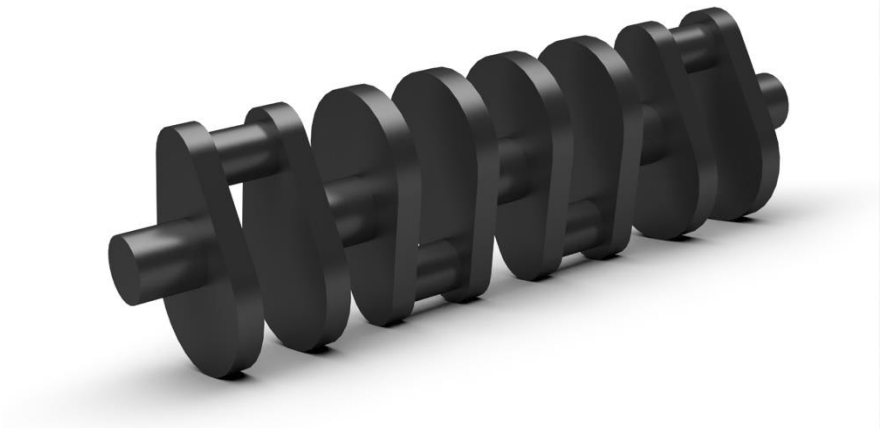
Material Mechanics



Kinematic Analysis



Mass Properties

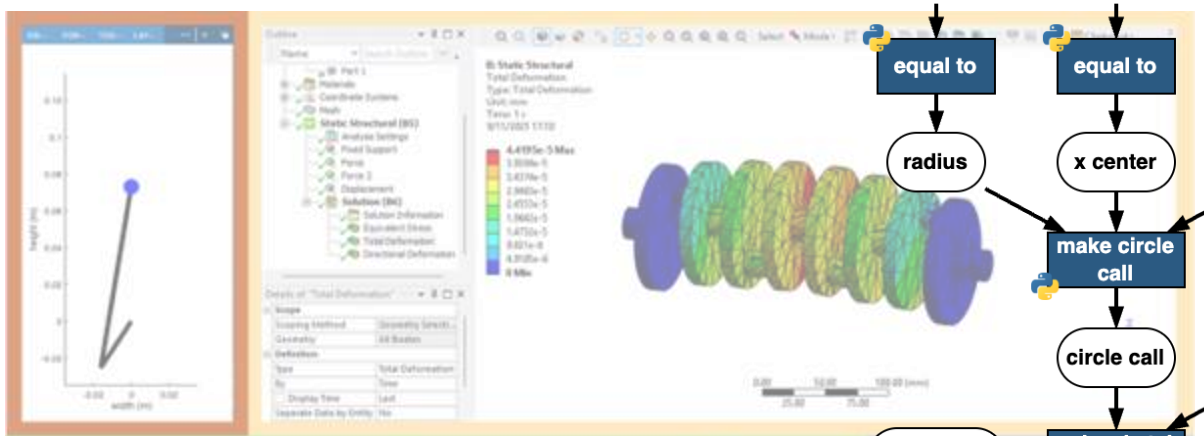


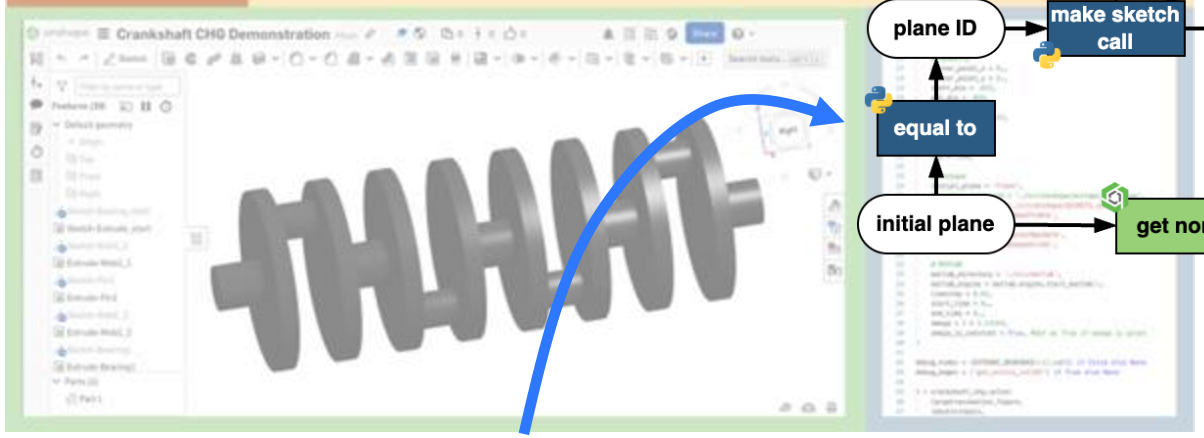
Ansys

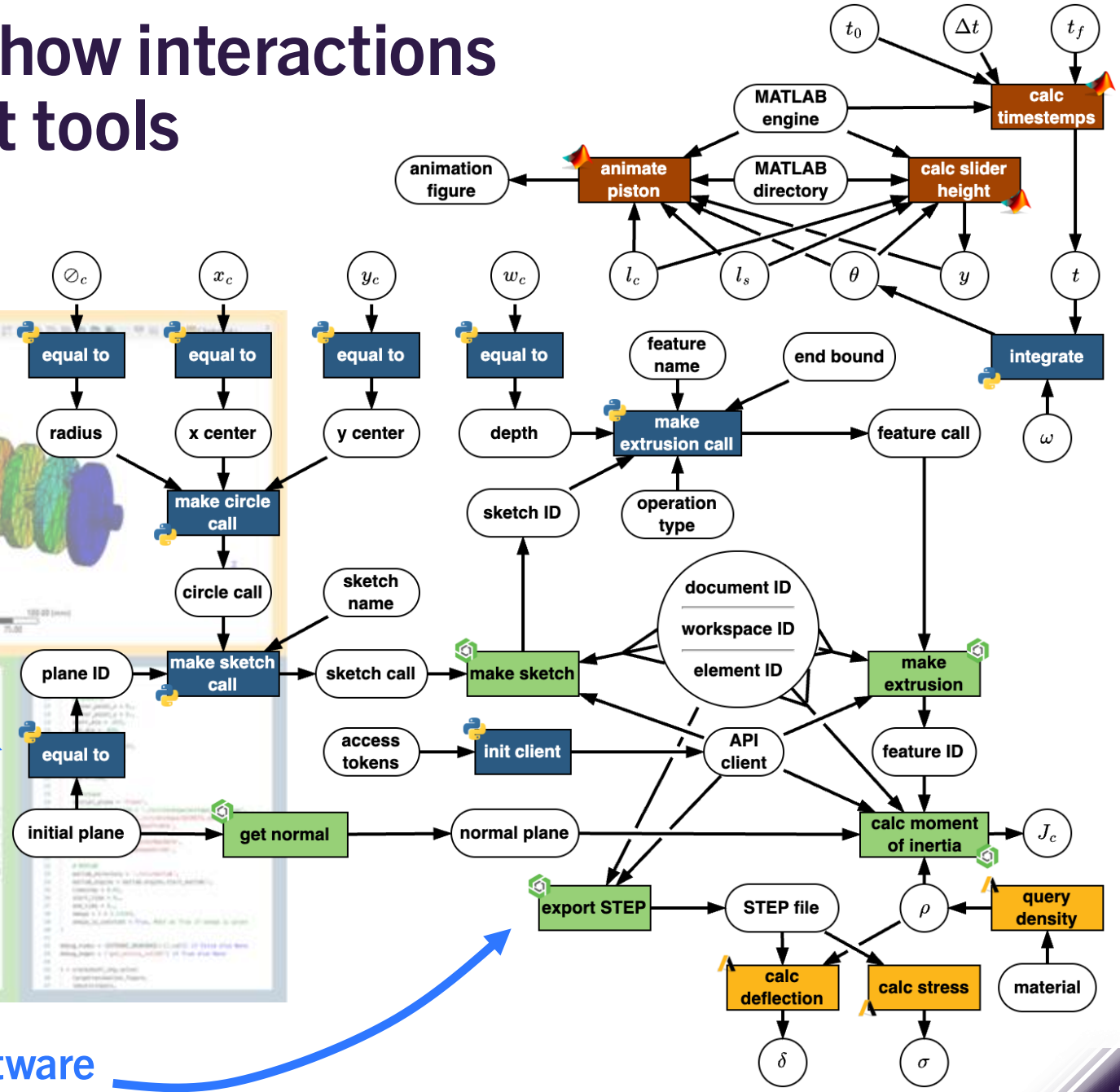
MATLAB® & SIMULINK®

onshape®

Intra-platform CHGs show interactions calculated by different tools

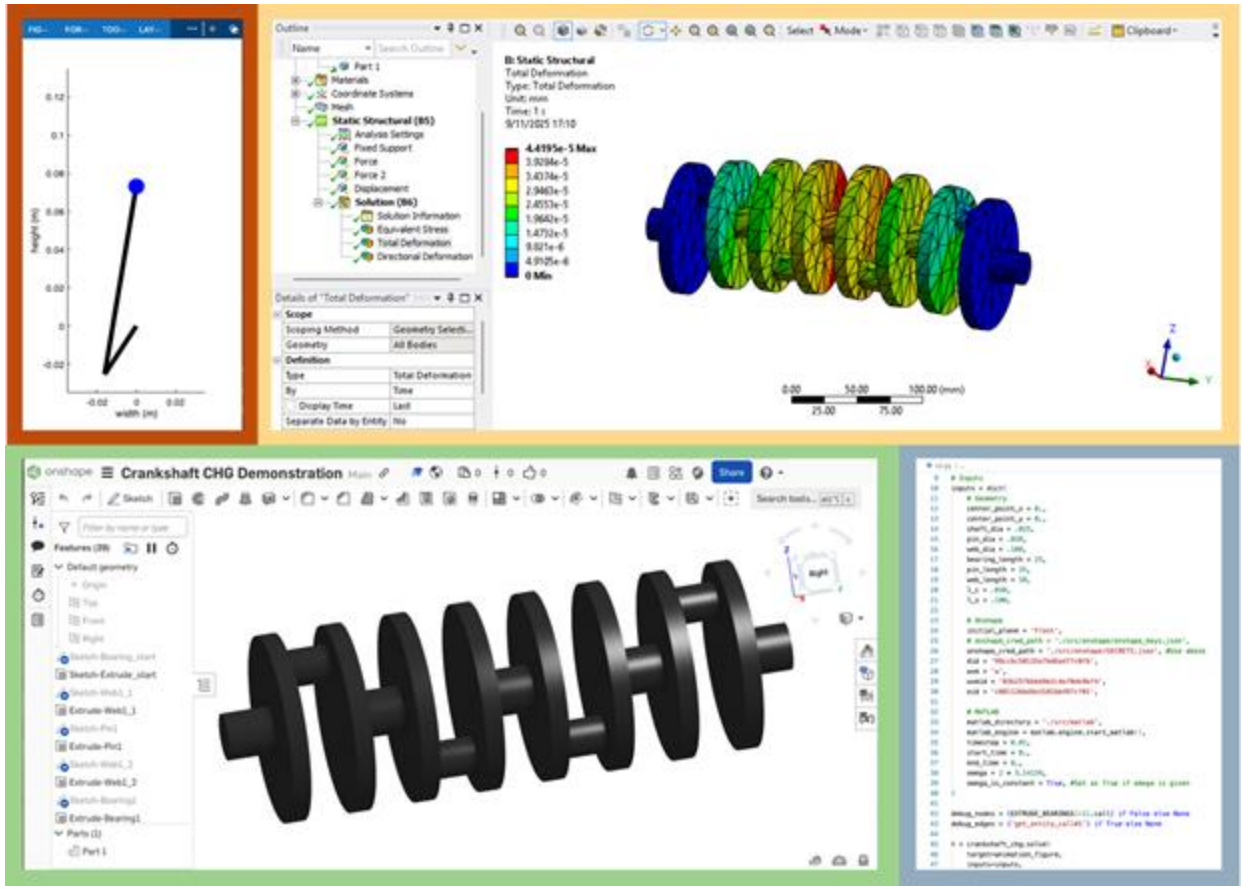






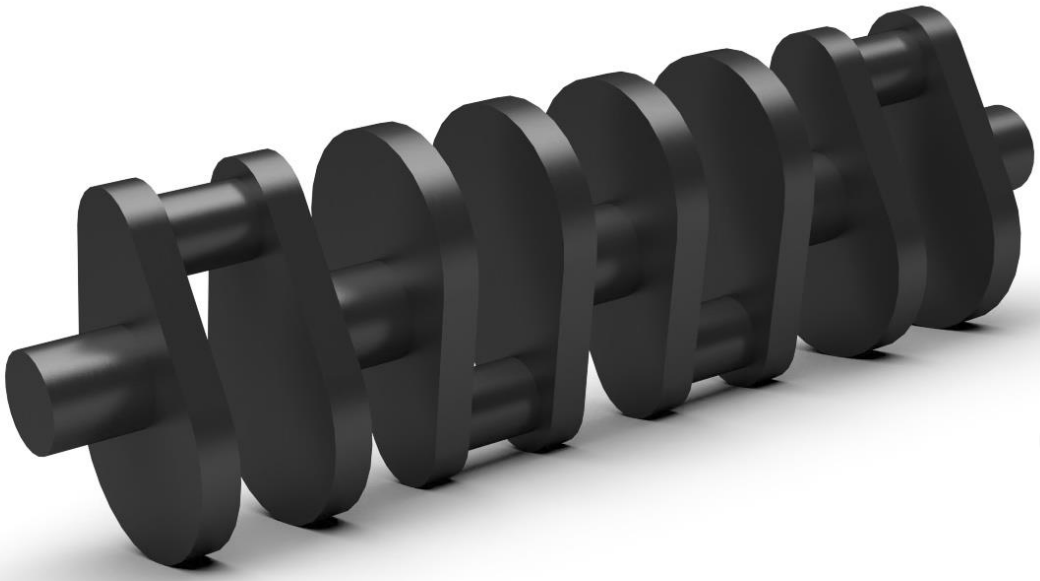
independent software

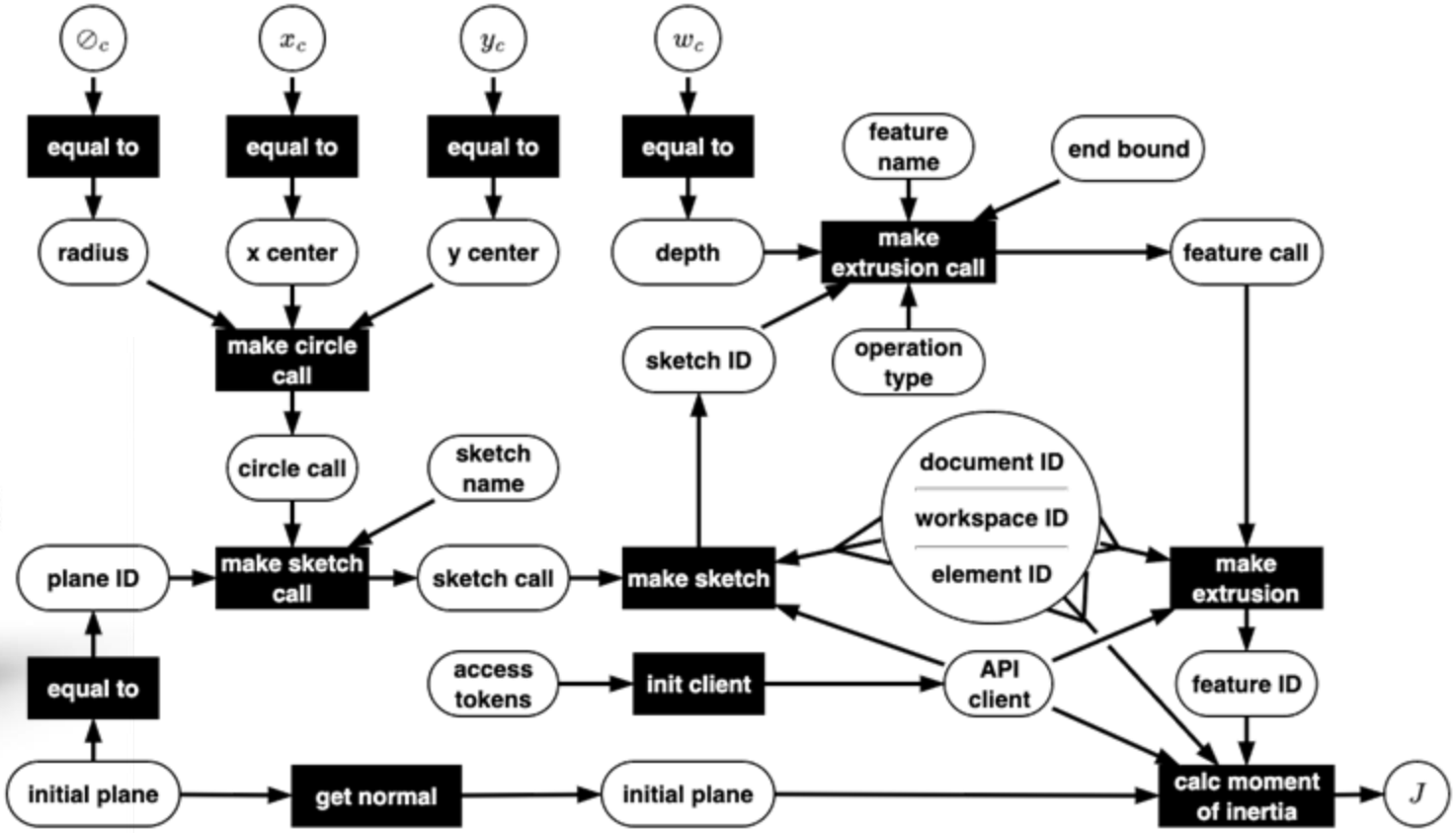
Representing software calculations as edges makes executable digital threads simpler



J. Morris, A. Indupally, G. Mocko, J. Wagner, and S. Ramnath, “Declarative, Multi-physics Simulation Between Applications via Constraint Hypergraphs,” Under review with J. Comput. Inf. Sci. Eng., (2025)

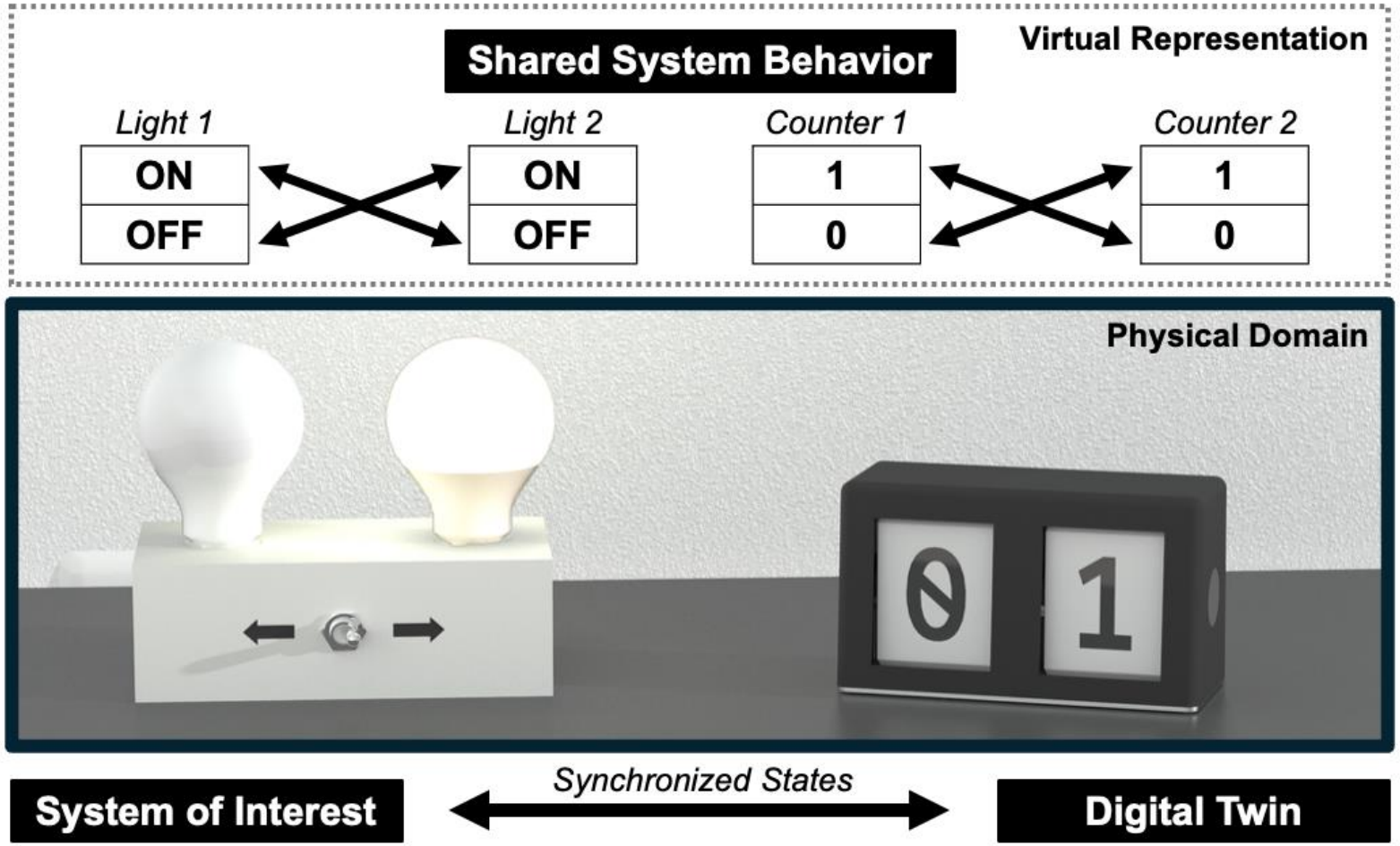
Limitation: not focused on being human readable





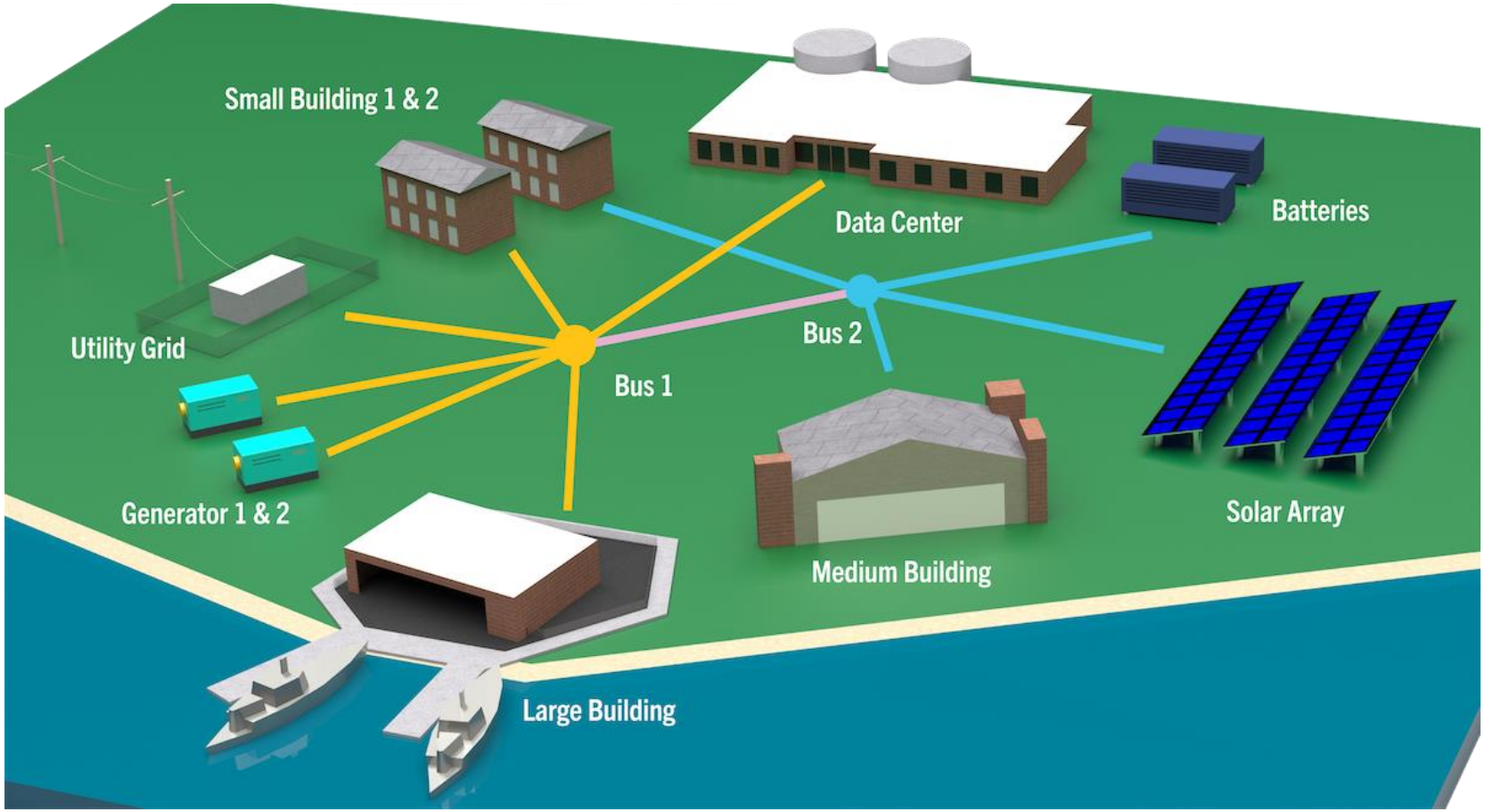
DTs are systems synchronized with another system of interest

Digital Twins are 2 systems sharing the same CHG

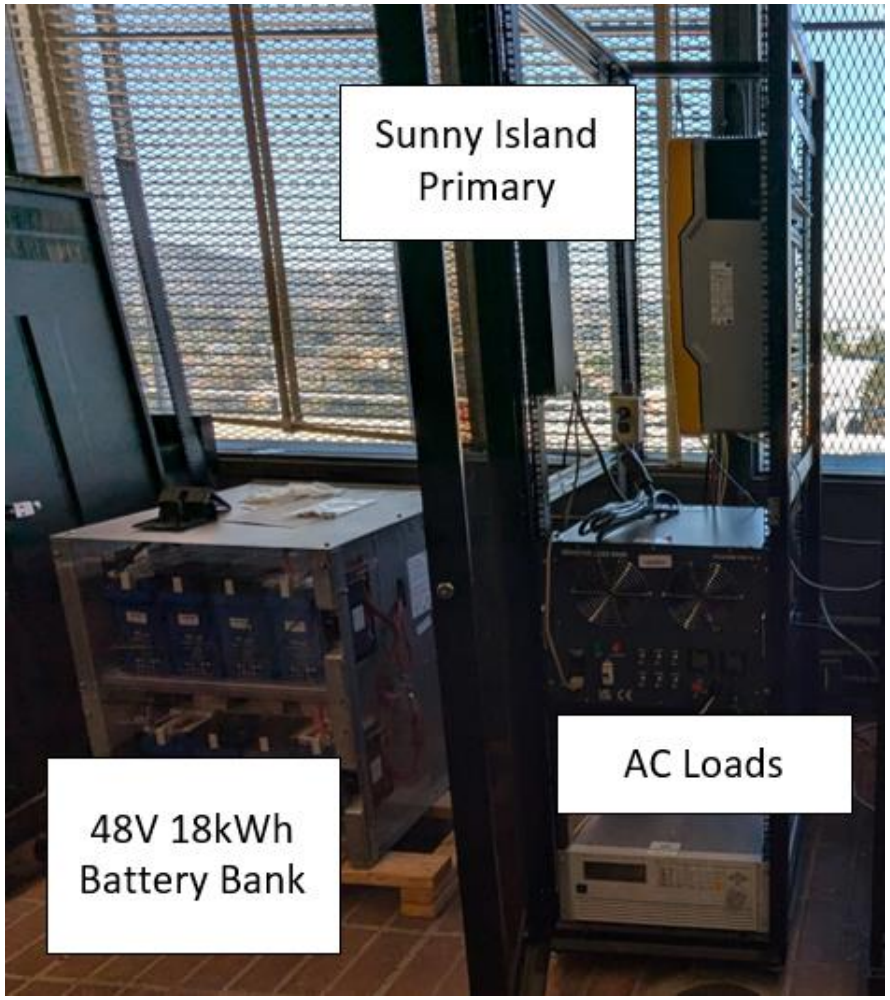


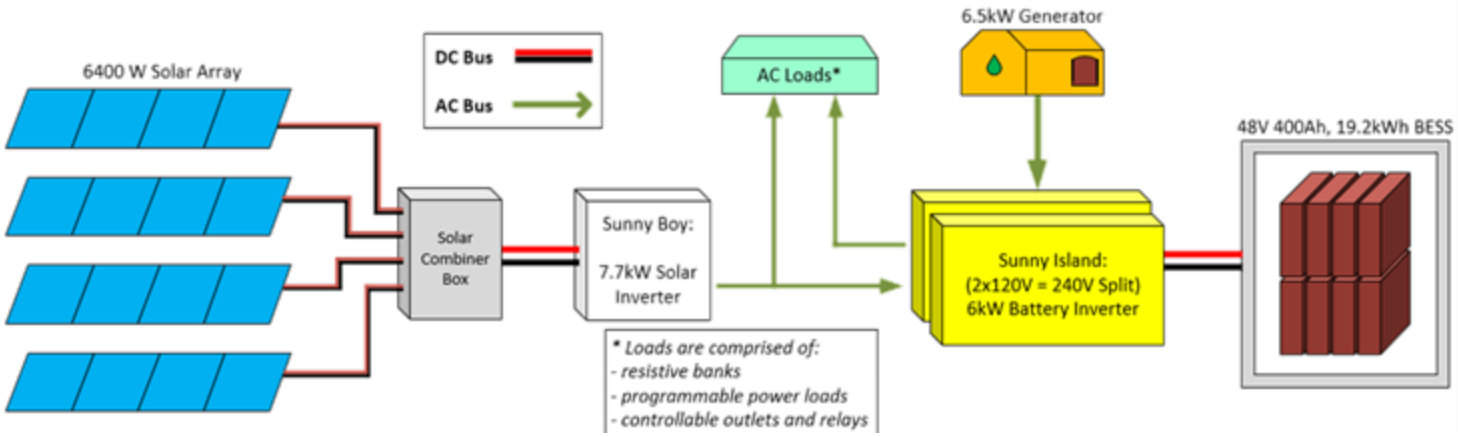
J. Morris, E. Louis, D. L. Van Bossuyt, G. Mocko, and J. Wagner, "Constraint Hypergraphs as a Unifying Framework for Digital Twins," Under review with IEEE Systems Journal, 2025

Demonstration: Naval Microgrid Digital Twin



Physical Microgrid at Naval Postgraduate School



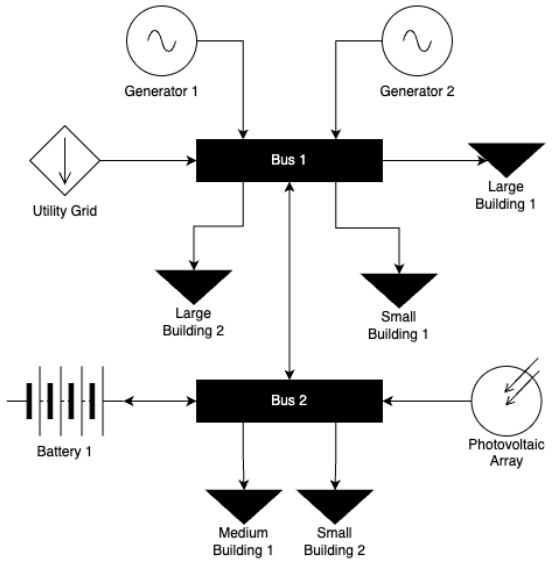


Previous Imperative Modeling of Microgrid

MG_LP_Test_v10.m

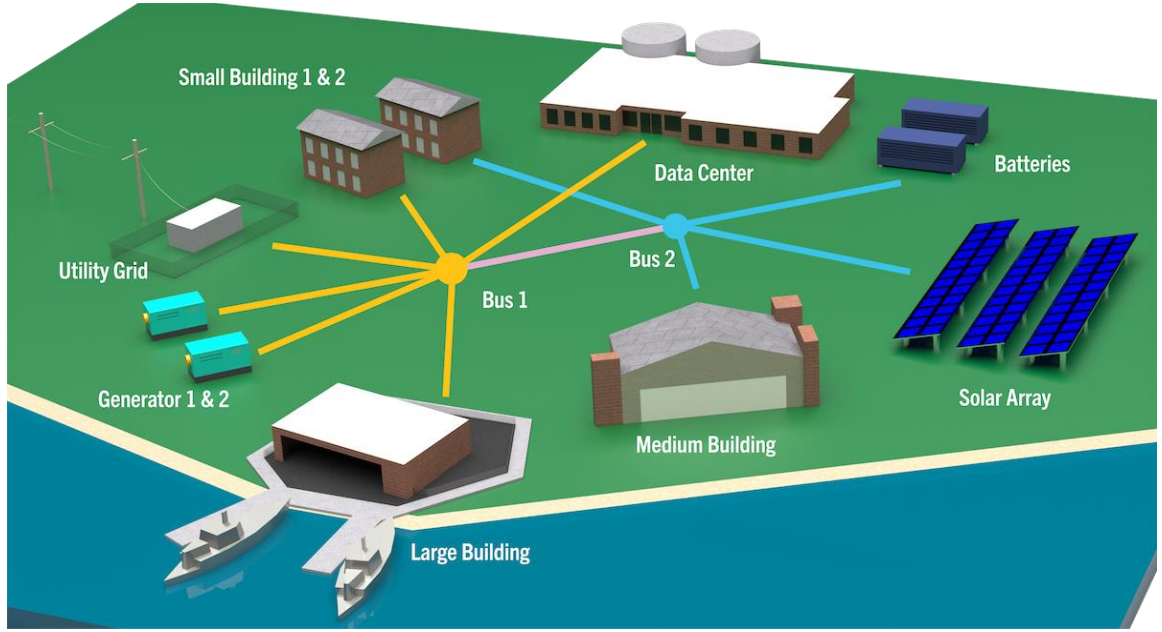
/Users/john_morris/Documents/Clemson/Research/MicrogridHg/archived_model/MG_LP_Test_v

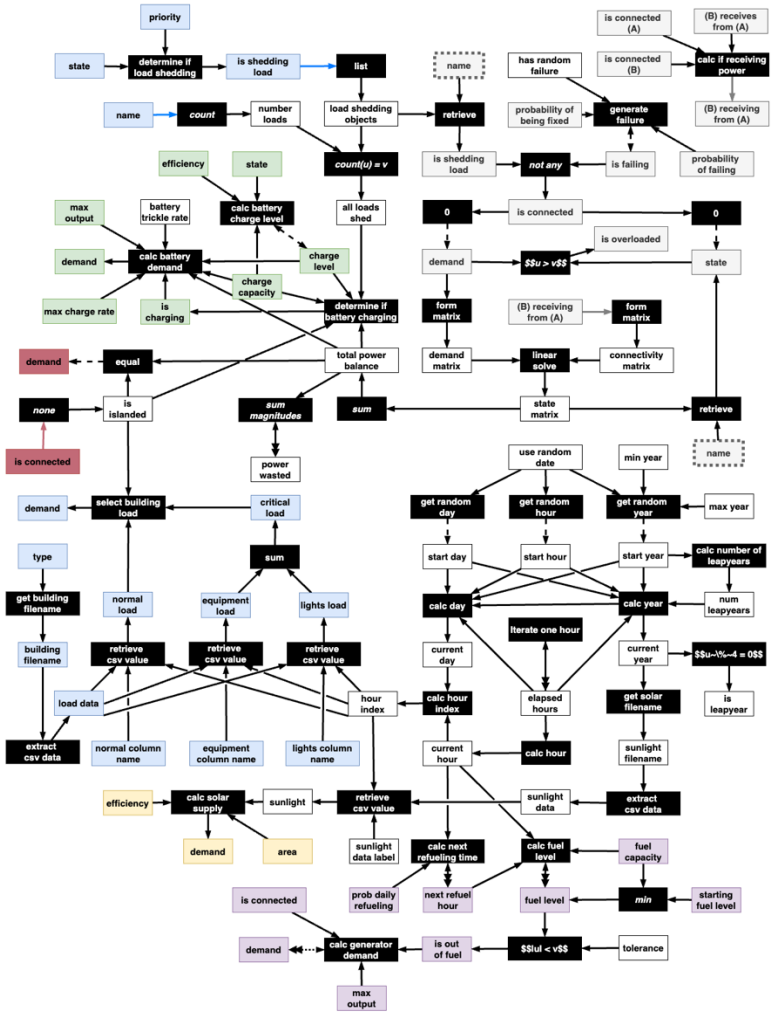
```
423         %Initial Solve
424         x(:,n) = linsolve(A2,b);
425
426         %Determine if generator demand exceeds generat
427         Gen_Demand = -x(pos.Gens,n);
428
429         if any(Gen_Demand > [Gens.Capacity]')
430             %If the B2 to B1 Buss line or BT1 line is
431             %exhausted cannot utilize ESS to make up g
432             if MG_State.B2_B1(n) == false || MG_State.
433                     || BT1_Charge(n) < 0
434                 % Below line was useful for debugging
435                 % disp("Gens Overloaded at time step "
436                 Overload.Gens = true;
437             else
438                 %Otherwise set Generators at full outp
439                 %make up unmet demand
440                 A2(6,:) = pos.Gens & MG_State{n,:};
441                 b(6) = -MG_State{n,{'B1_GEN1' 'B1_GEN2
442                 x(:,n) = linsolve(A2,b);
443             end
444         %If Generator demand is negative, then generat
445         %charge batteries with excess PV generation
446         elseif any(Gen_Demand < 0)
447             A2(6,:) = pos.Gens;
448             b(6) = 0;
449             x(:,n) = linsolve(A2,b);
450         end
451
452         % Check for Battery Output Exceeded
453         % Add 0.1 Due to Rounding Errors in Linear Solver
454         if -x(contains(LoadVars,'BT1'),n) - 0.01 > BT1.Output * (BT1_Charge(n) > 0)
455             Overload.BT1 = true;
456             % Below line was useful for debugging in single runs. Commented out
```



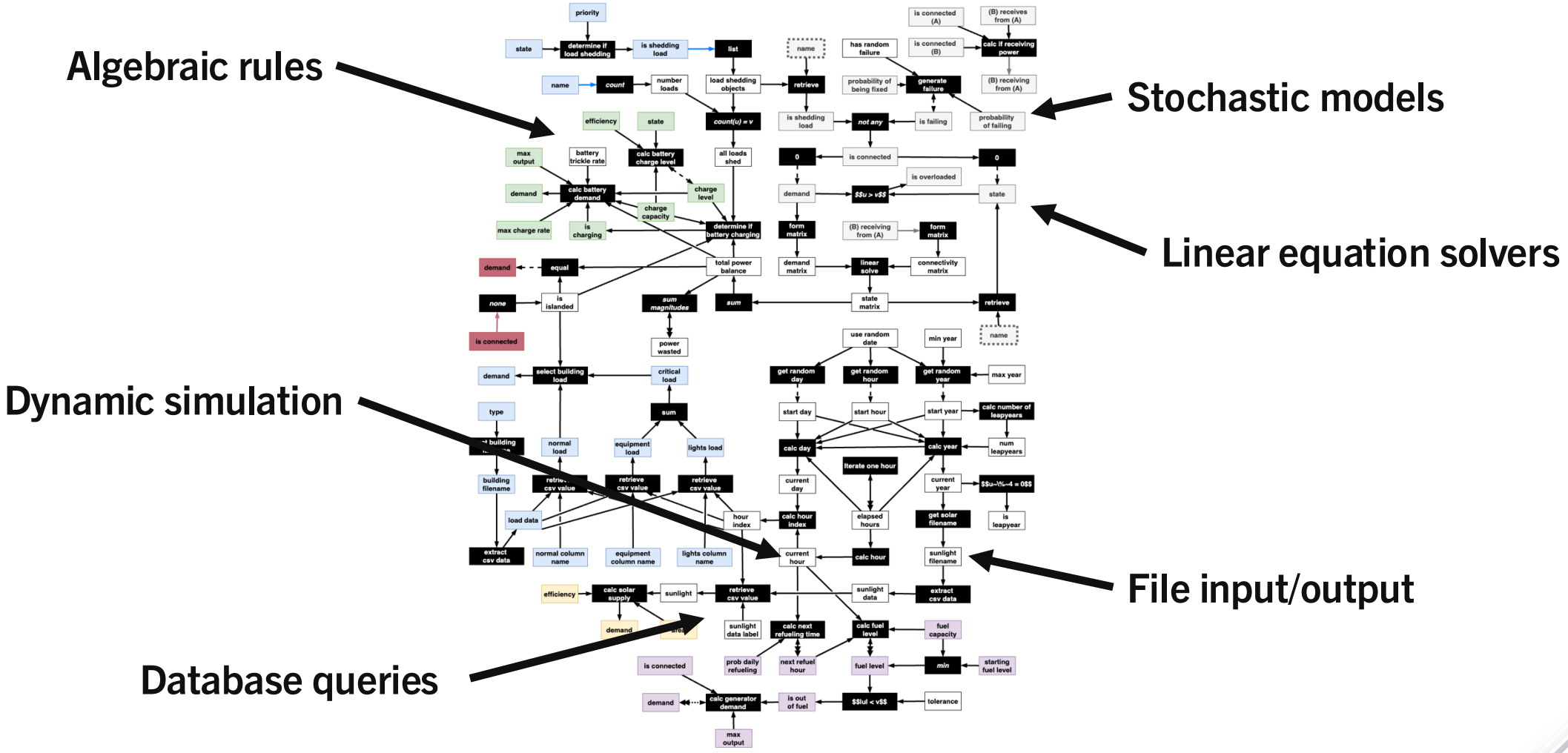
C. J. Peterson, "Systems Architecture Design and Validation Methods for Microgrid Systems," Master's thesis, Naval Postgraduate School, Monterey, CA, 2019

Transformed CHG contains 594 nodes and 334 edges

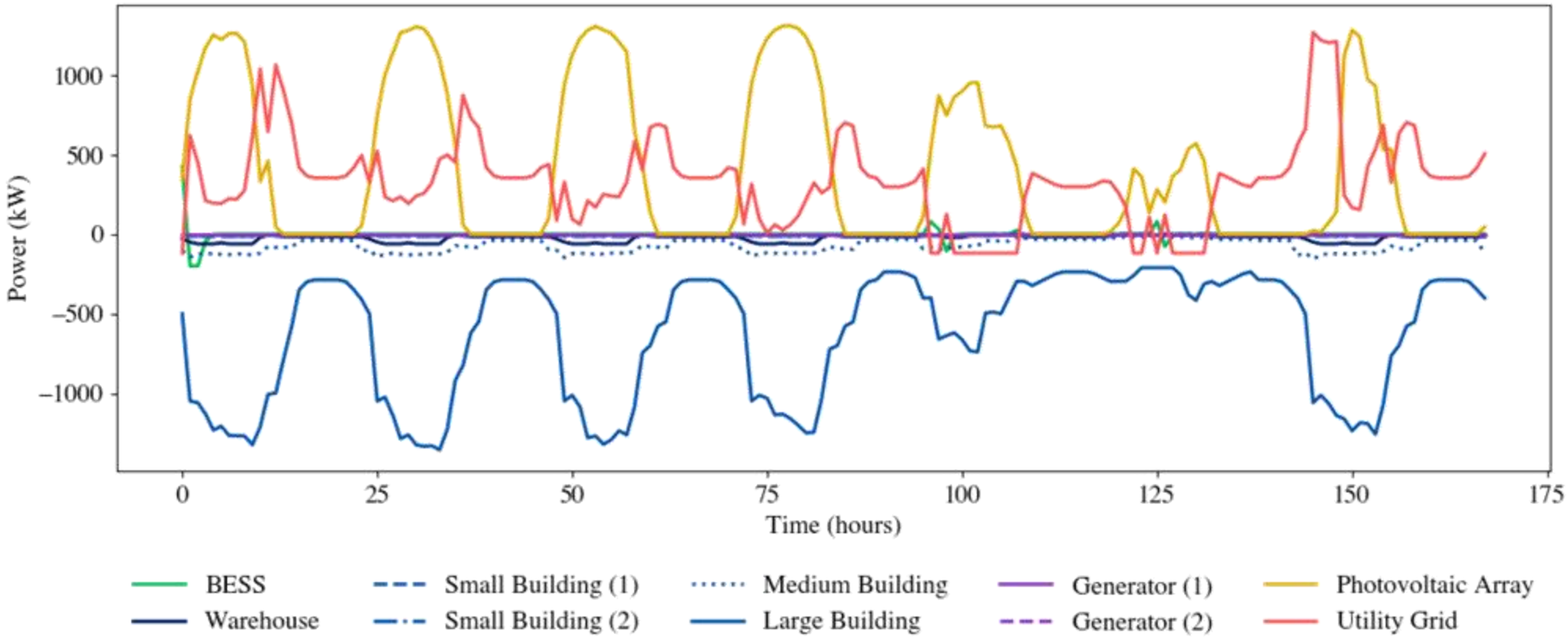




Includes all kinds of simulation and modeling



Autonomously observes and simulates the real system

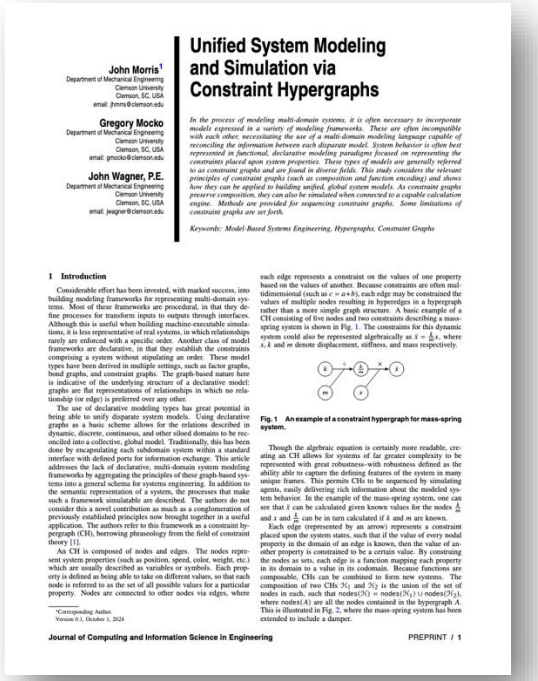


Establishing strong theory of CHGs enables new abilities in advanced systems modeling

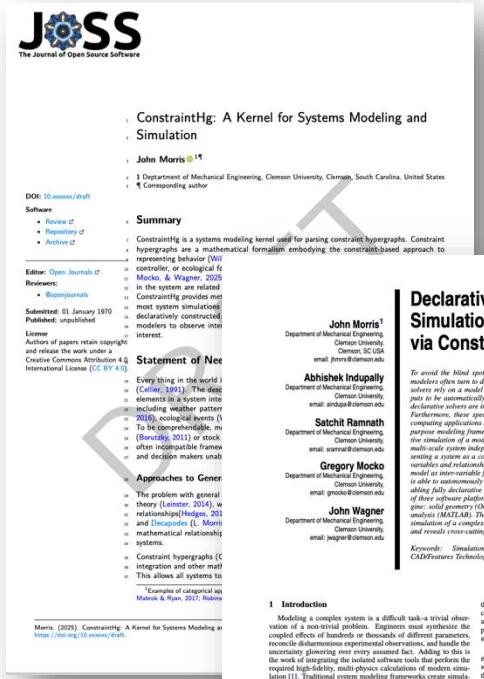
Constraint Hypergraphs as a Unifying Framework for Digital Twins

John Morris, Douglas L. Van Bossuyt, Edward Louis, Gregory Mocko, John Wagner, *Senior Member, IEEE*

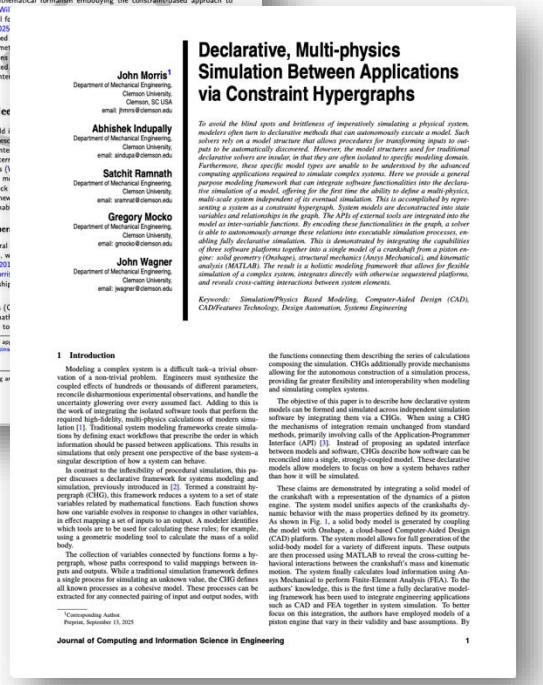
Effects of Functional and Declarative Modeling Frameworks on System Simulation



Unified System Modeling and Simulation via Constraint Hypergraphs



ConstraintHg: A Kernel for Systems Modeling and Simulation



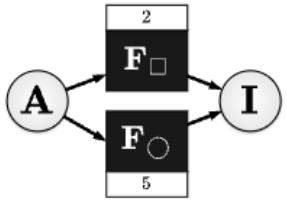
Declarative, Multi-physics Simulation Between Applications via Constraint Hypergraphs

Future work: best practices for model selection

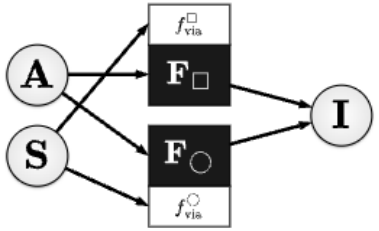
Model Selection

Abstracted Behavior

Model Preferences

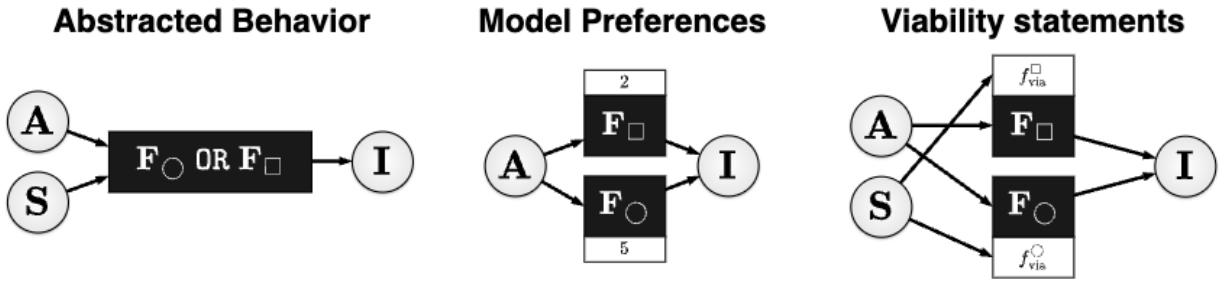


Viability statements

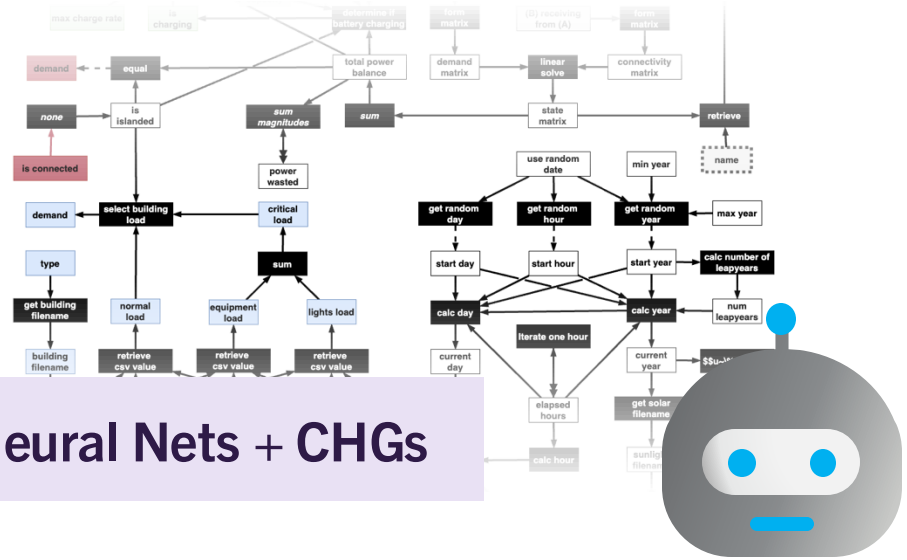


Future work: integrating AI with CHGs

Model Selection

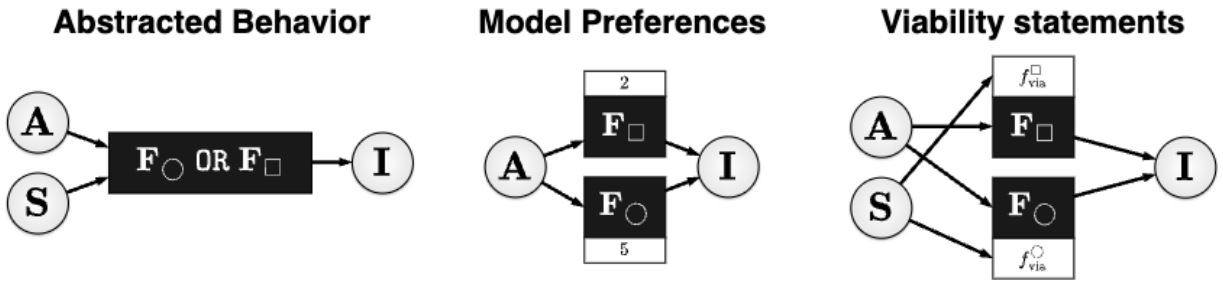


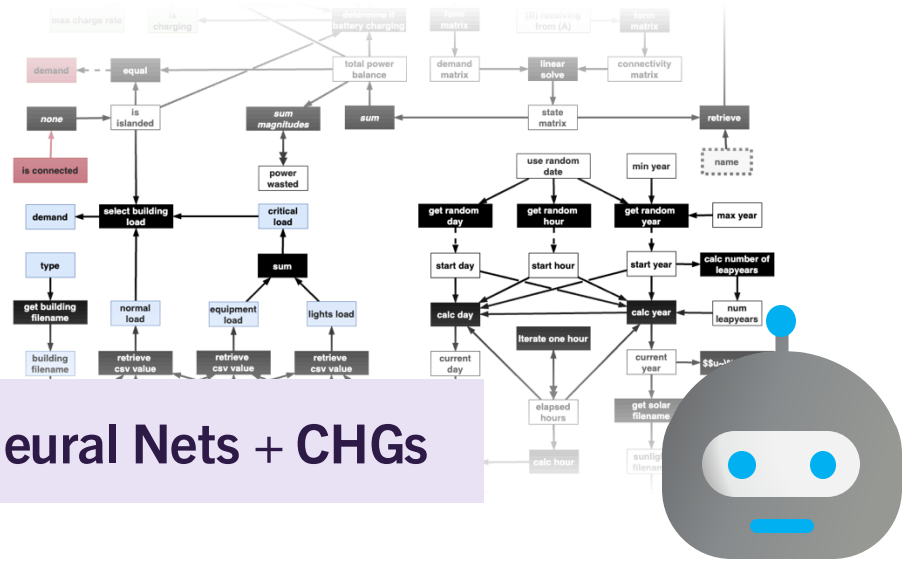
Neural Nets + CHGs



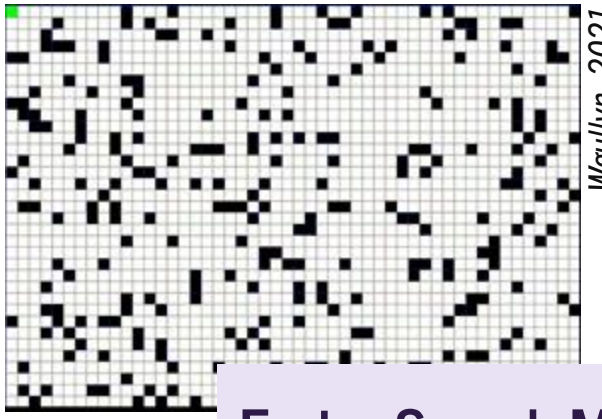
Future work: heuristic search methods

Model Selection





Neural Nets + CHGs

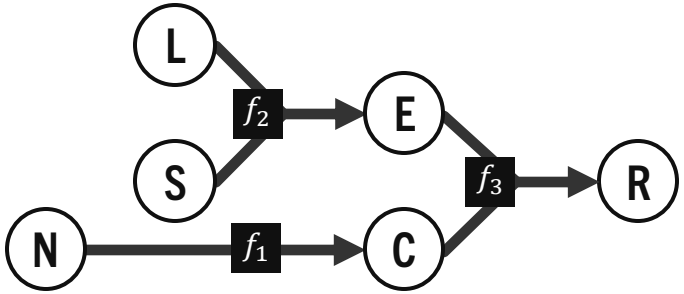


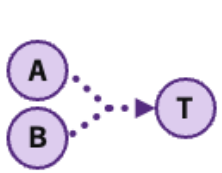
Wgullyn, 2021

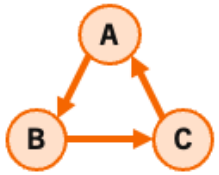
Faster Search Methods

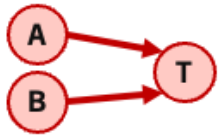
The structure of a CHG allows it to represent any system

Universal Modeling



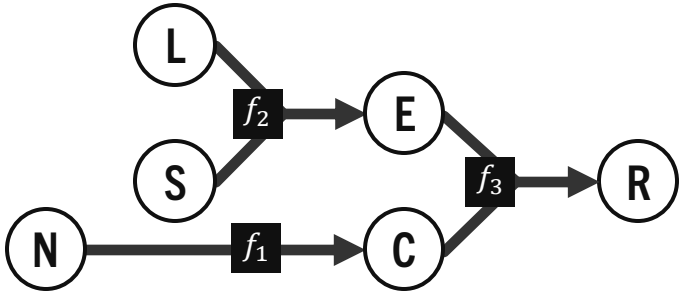


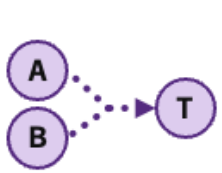


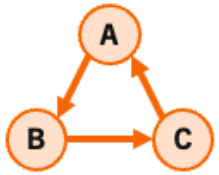


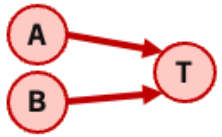
Information about a system can be discovered autonomously

Universal Modeling





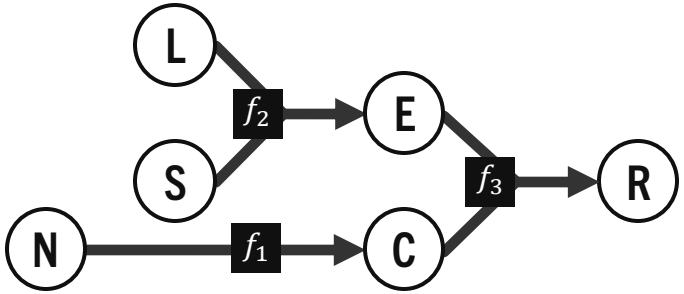


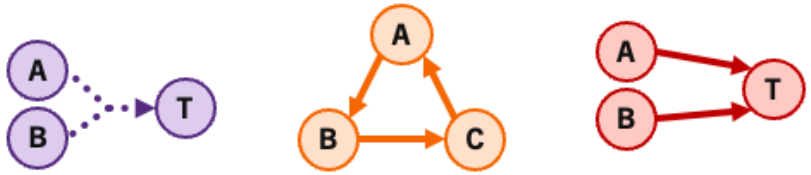


Declarative Simulation

Models can be connected across domains, scales, and even tools

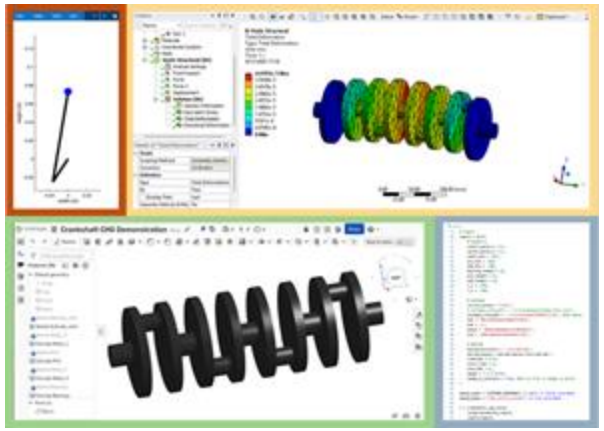
Universal Modeling

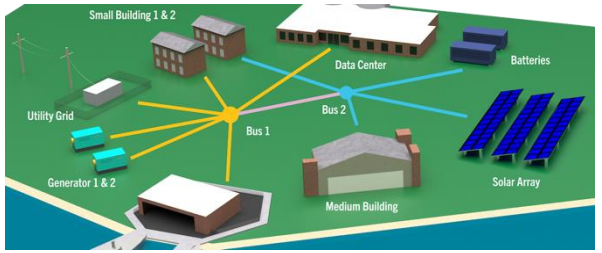




Declarative Simulation

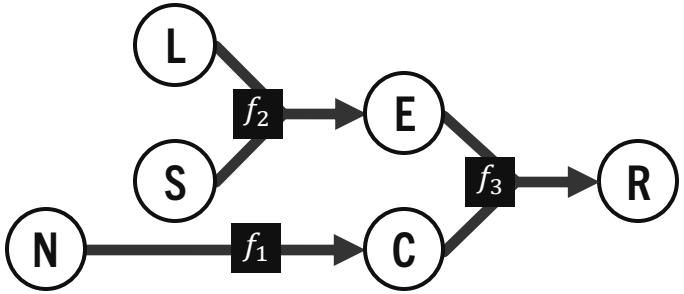
Executable Digital Threads

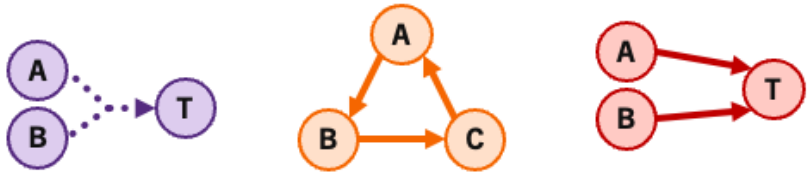




CHGs provide a universal language for declarative systems modeling

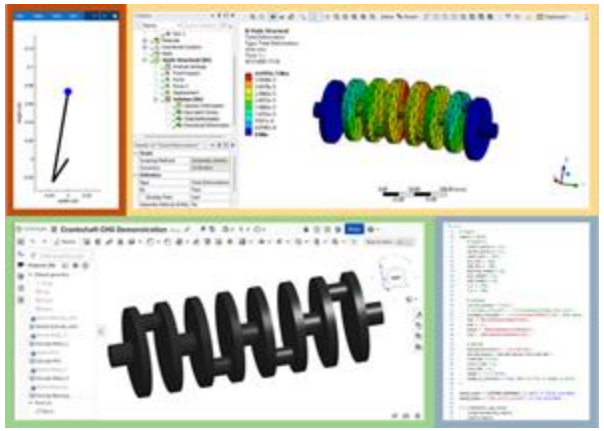
Universal Modeling

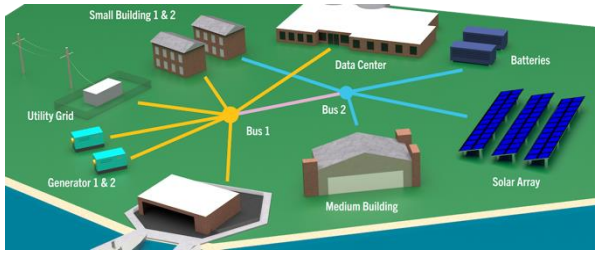




Declarative Simulation

Executable Digital Threads





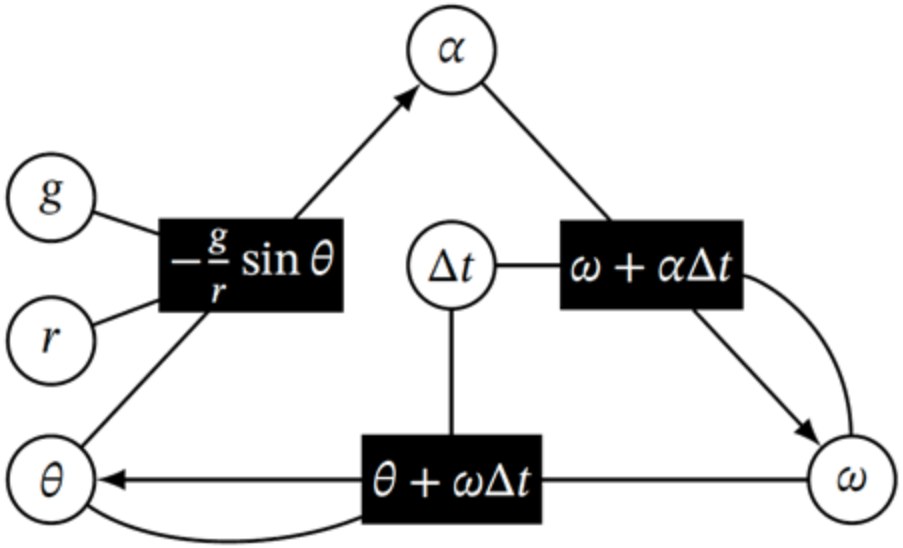
PLMC

CEDAR
Clemson Engineering Design Applications and Research

CLEMSON

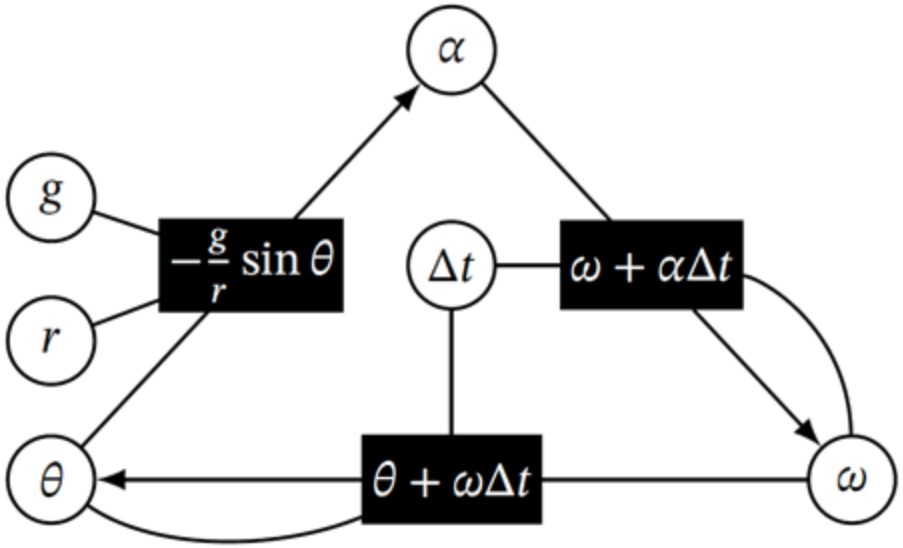
Add Relations

```
def Rtheta_to_alpha(theta, g, r):
    return -g / r * sin(theta)

def Rintegrate_omega(omega, alpha, delta_t):
    return omega + alpha * delta_t

def Rintegrate_theta(theta, omega, delta_t):
    return theta + omega * delta_t
```



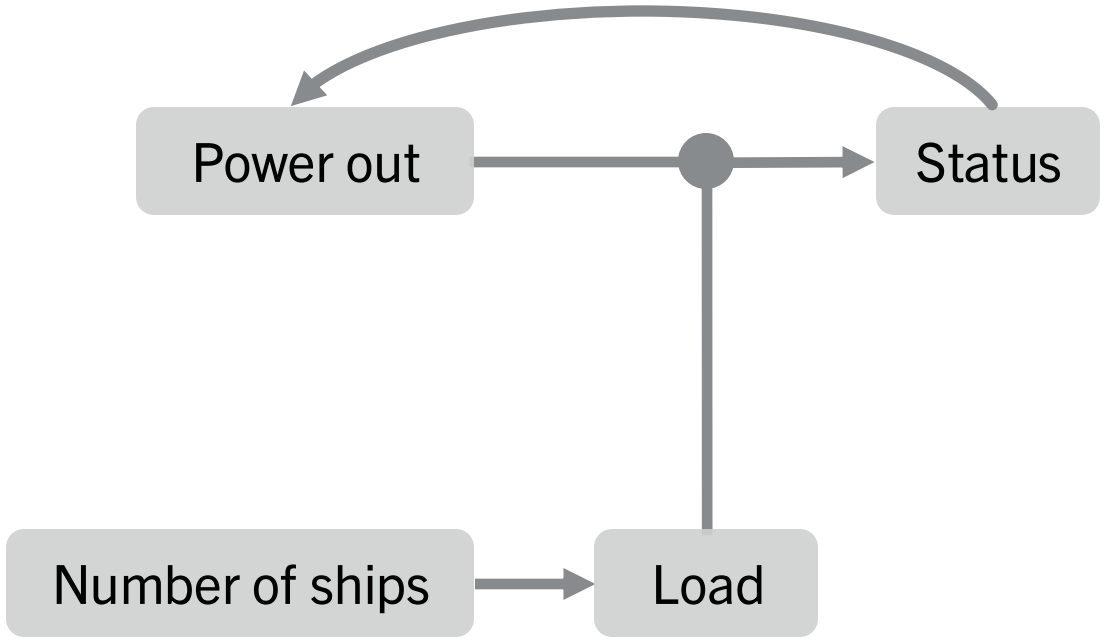
Add Edges

```
hg = Hypergraph()

hg.add_edge(
    sources=('theta', 'g', 'r’),
    target='alpha’,
    rel=Rtheta_to_alpha,
)
hg.add_edge(
    sources=('omega', 'alpha’, ‘delta_t’),
    target='omega’,
    rel=Rintegrate_omega,
)
...
```



Cycles violate transitivity, indicating conflicting models

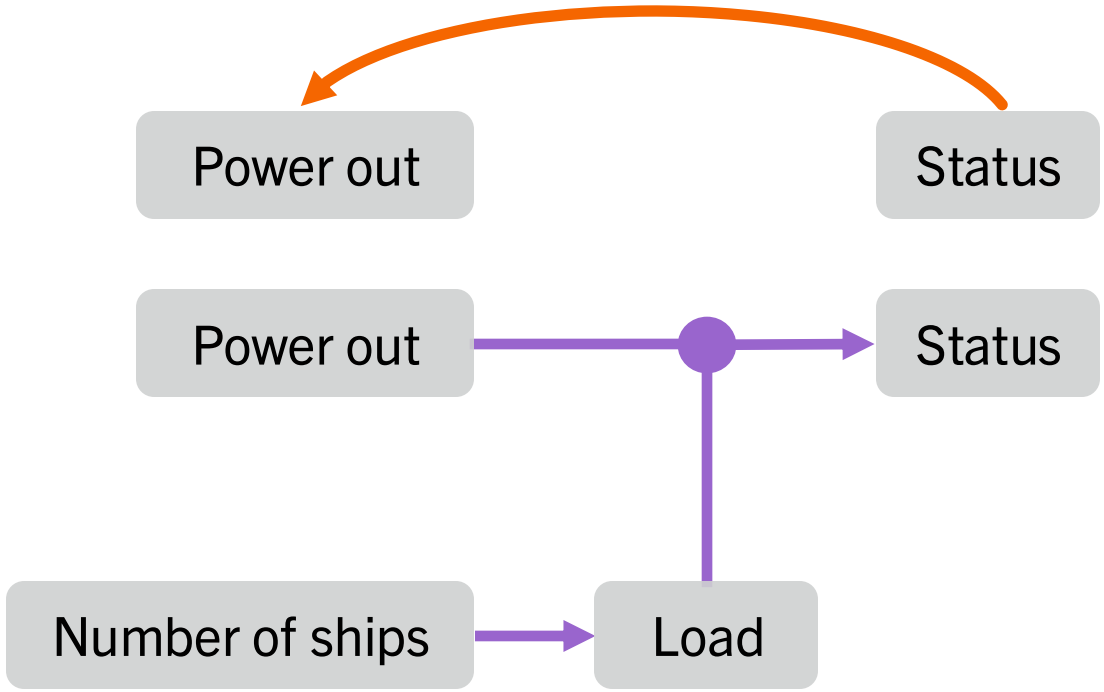
Can be dealt with in two ways:
1. Separation
2. Unraveling



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].

Separating cycles into distinct graphs

Good for isolating *competing* models

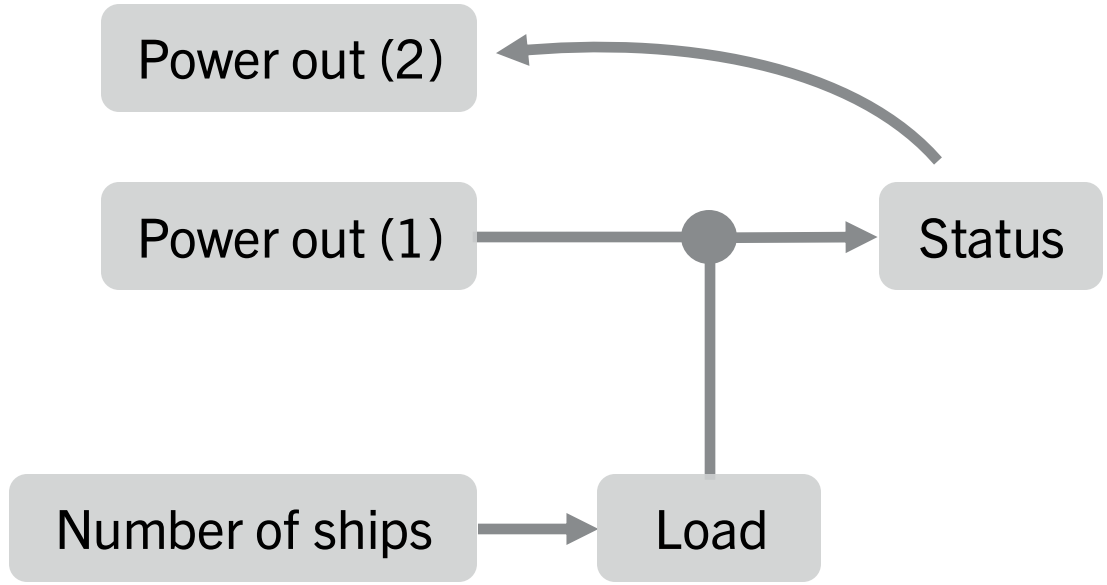


Robinson, M. (2017). Sheaf and Duality Methods for Analyzing Multi-Model Systems. In Recent Applications of Harmonic Analysis to Function Spaces, Differential Equations, and Data Science. Springer International Publishing.

Unraveling nodes allows cycles to extend the hypergraph

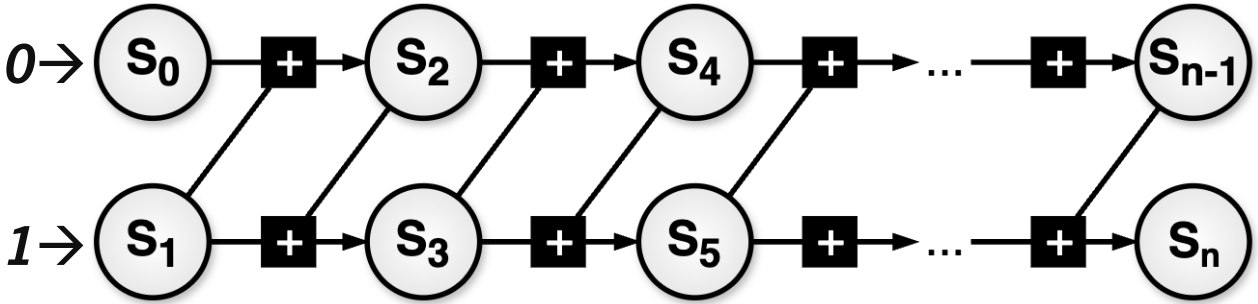
Good for *pattern* adaption

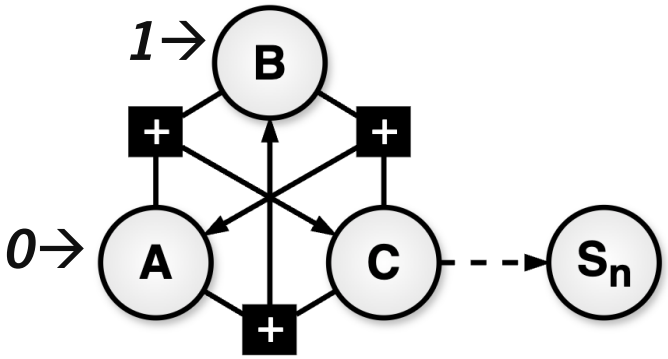
Unraveling occurs while searching for simulation paths and requires an exit edge

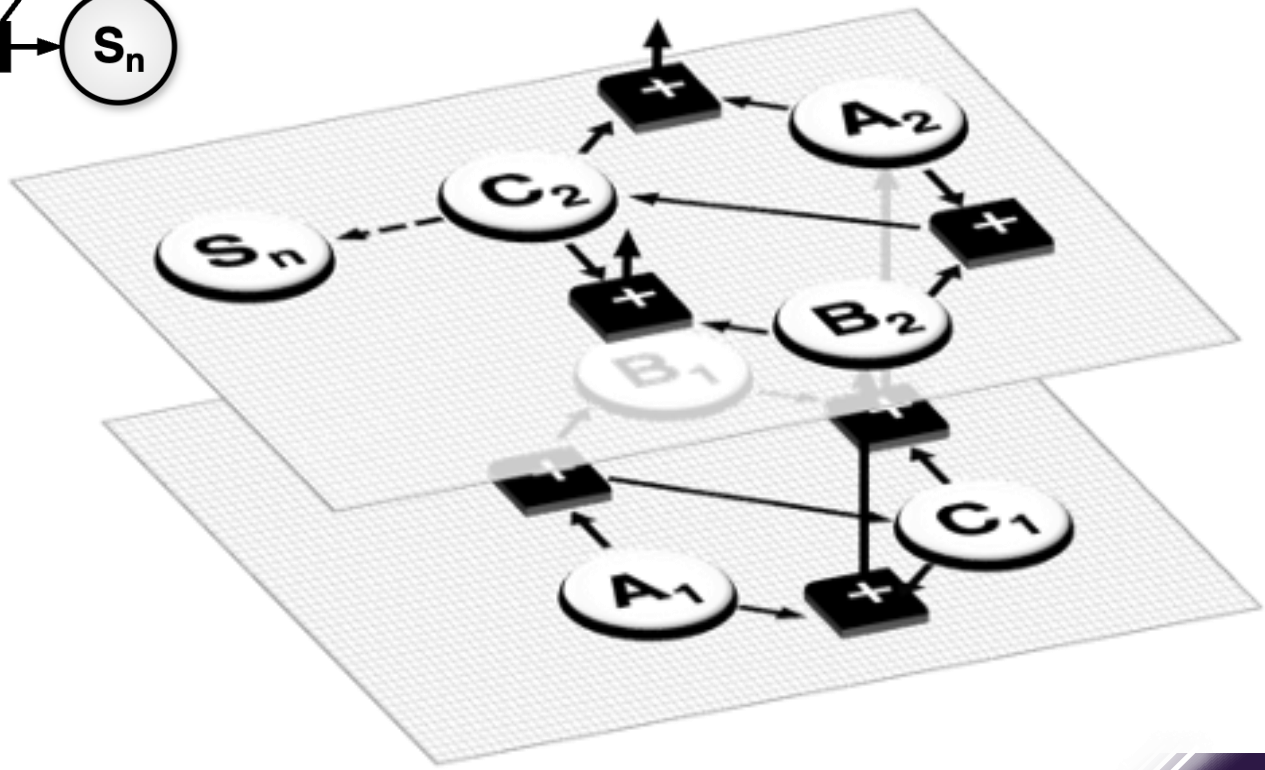


Morris, J., Mocko, G., & Wagner, J. (2025). Effects of Functional and Declarative Modeling Frameworks on System Simulation. [Manuscript Submitted for Review to the ASME Journal of Dynamic Systems, Measurement and Control].

Example computing Fibonacci sequence using cloning

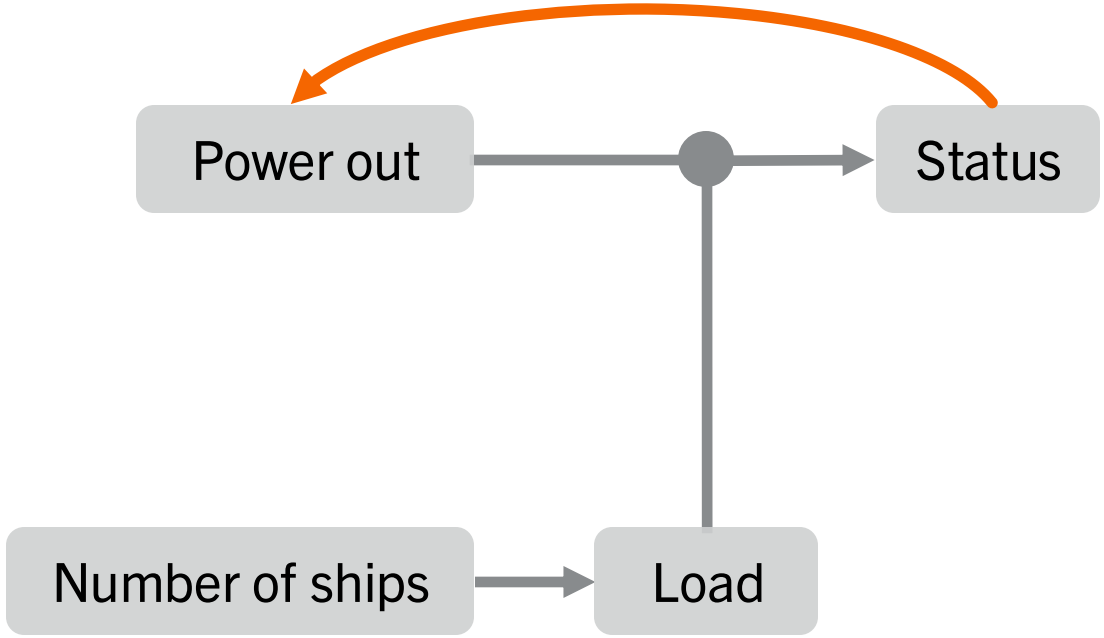






Independent behaviors make the hypergraphs immediately modifiable

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

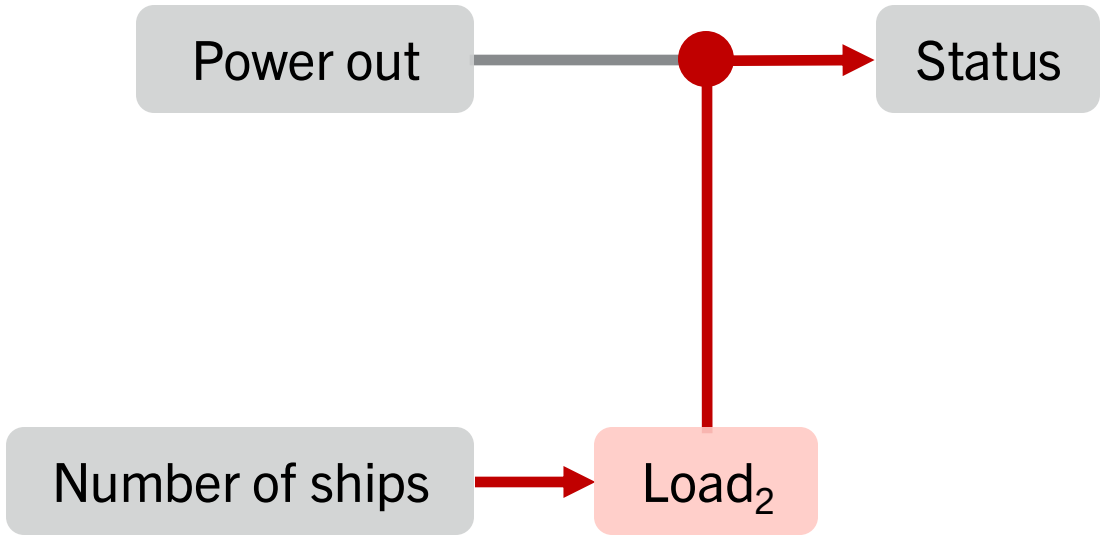


Morris, J., Mocko, G., & Wagner, J. (2025). Effects of Functional and Declarative Modeling Frameworks on System Simulation. [Manuscript Submitted for Review to the ASME Journal of Dynamic Systems, Measurement and Control].

Caveat: scope changes are not guaranteed to be consistent

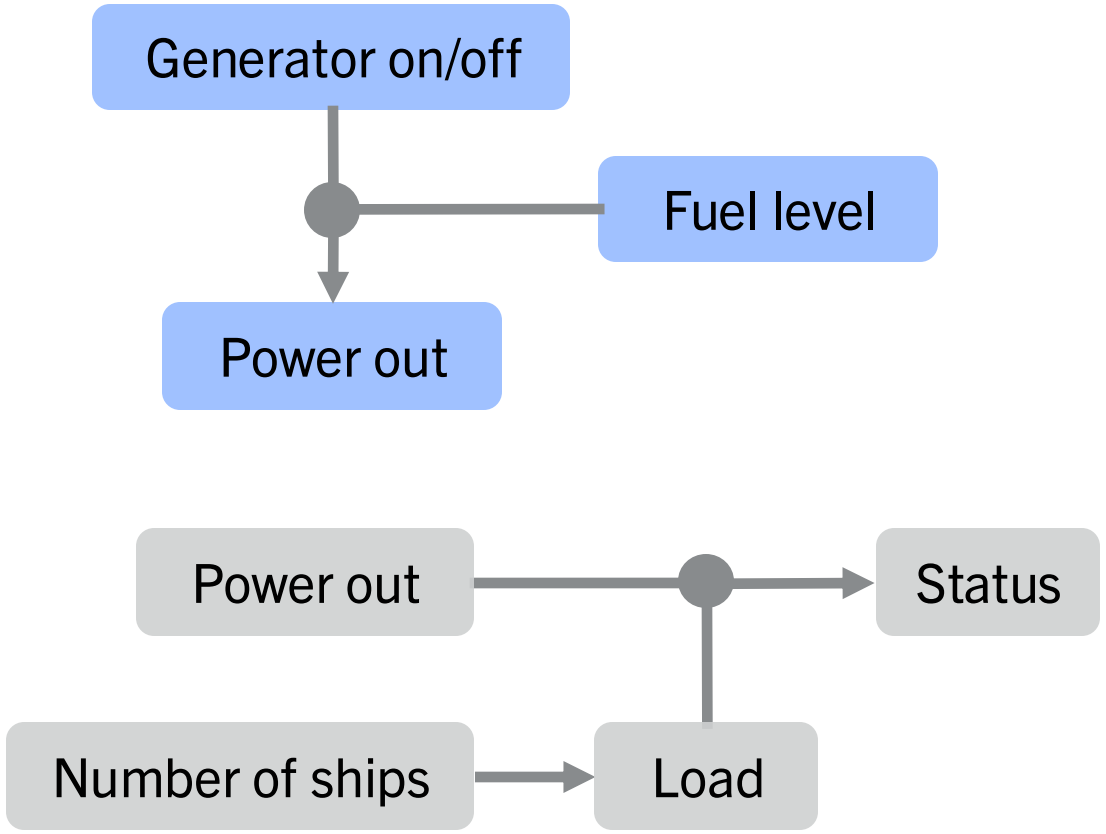
Nodes can be modified only if all connecting edges are also updated to maintain consistency *(only local side effects)*

Behavioral assumptions may be affected



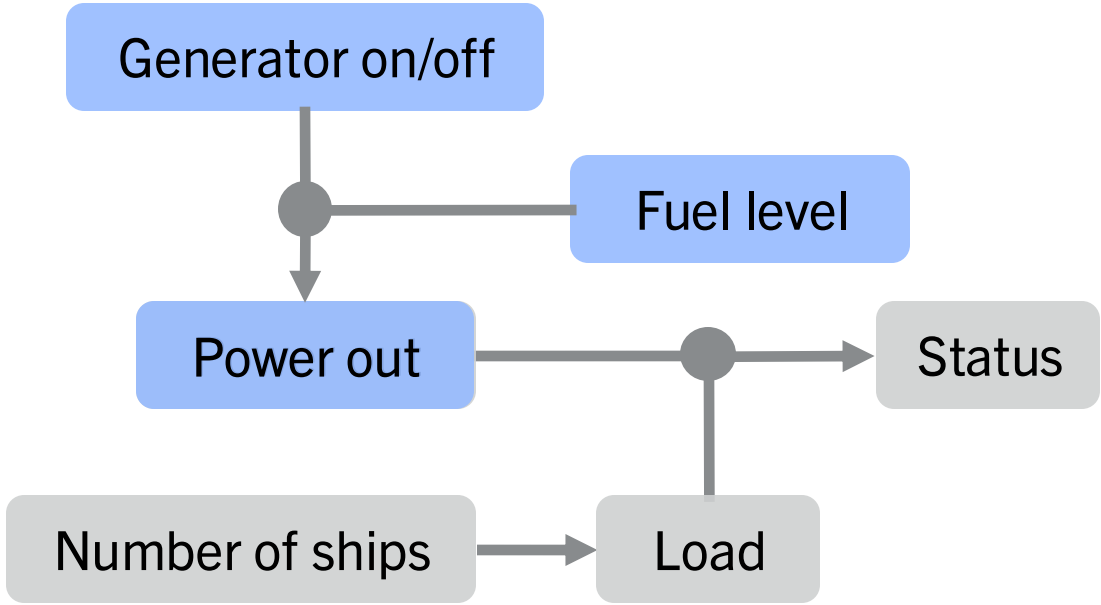
Composable with other graphs along shared nodes (union)

All simulation paths remain consistent *if* new scope does not violate behavioral assumptions



Emergent behavior is provided entirely from composition

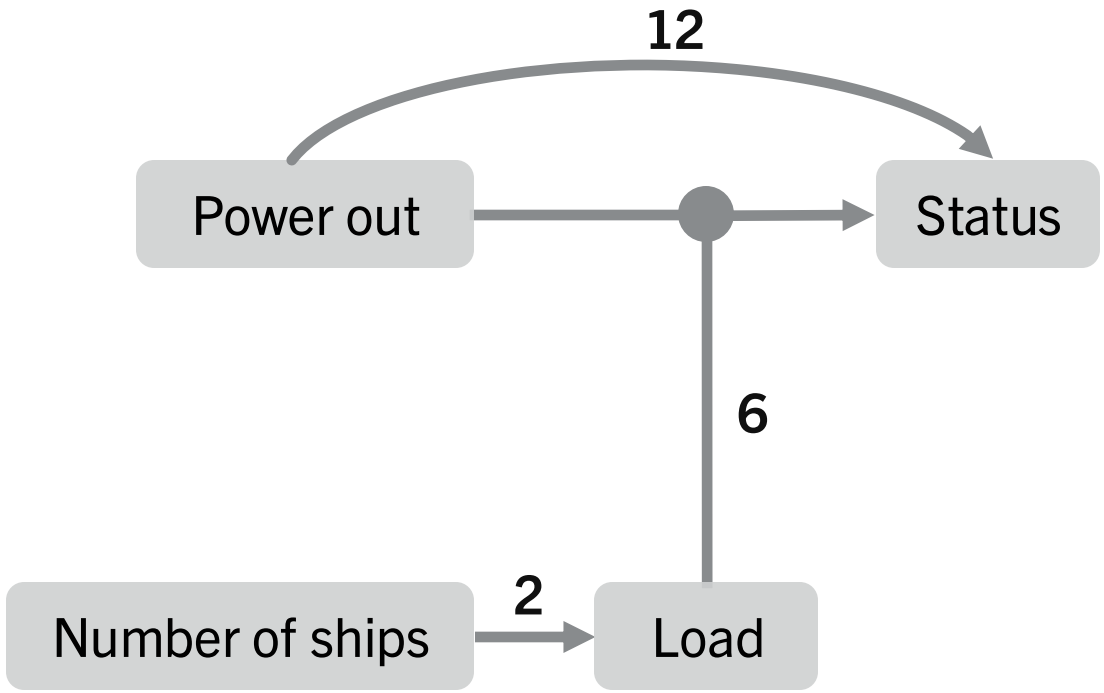
All simulation paths remain consistent *if* new scope does not violate behavioral assumptions



Weights allow model characterization

Edge labels allow models to be compared (model selection)

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

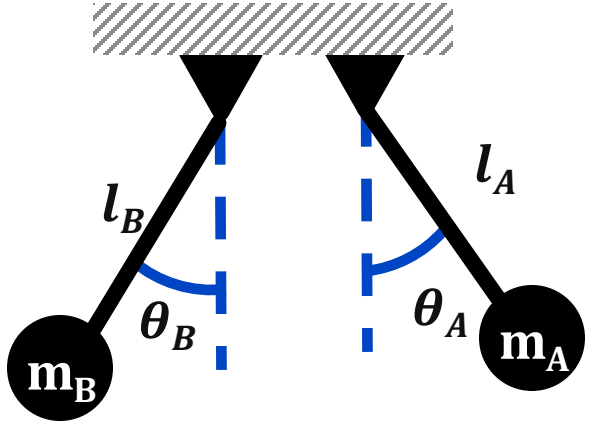


Morris, J., Mocko, G., & Wagner, J. (2025). Unified System Modeling and Simulation via Constraint Hypergraphs. Journal of Computing and Information Science in Engineering, [In Press]

Emergent behavior arises out of composed functions

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

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

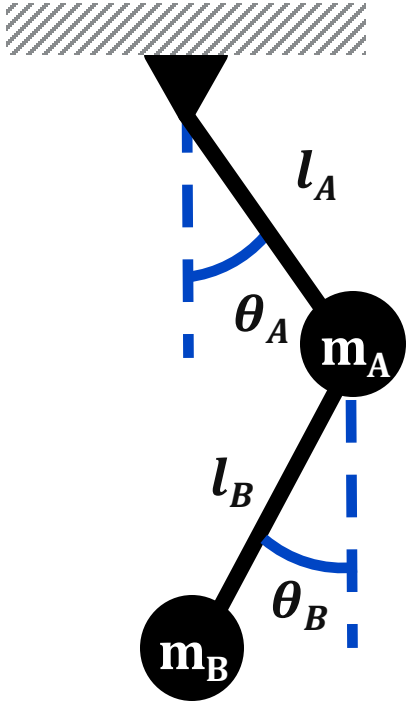


System is fully defined

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

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

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

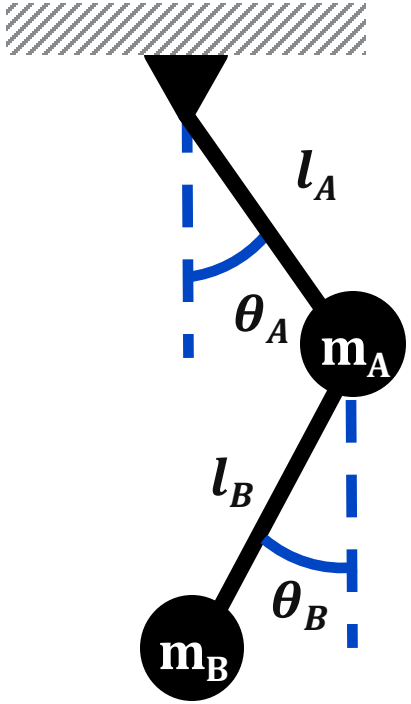


System cannot be represented as desired with the given nodes

Composition must be provided by showing the relationships between nodes

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

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



New model shows how the pendulums can be joined

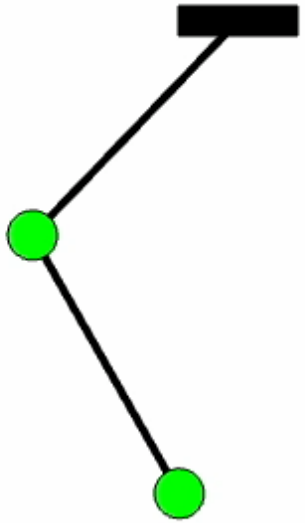
When composition is given, the emergent behavior can be fully realized

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

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