Universal Systems Simulation via Constraint Hypergraphs

John Morris

12 January 2026
Presentation to East Tennessee State University

East Tennessee State University
PLMC
CEDAR
Clemson Engineering Design Applications and Research

Biographical Sketch

BS: Mech Engr, BYU (2021)
Minor in Computer Science

Started as PLM Applications Engineer (2021)

MS: Mech Engr, Clemson (2023)
Emphases in dynamics, manufacturing, and design

Collaboration with NPS (2024)

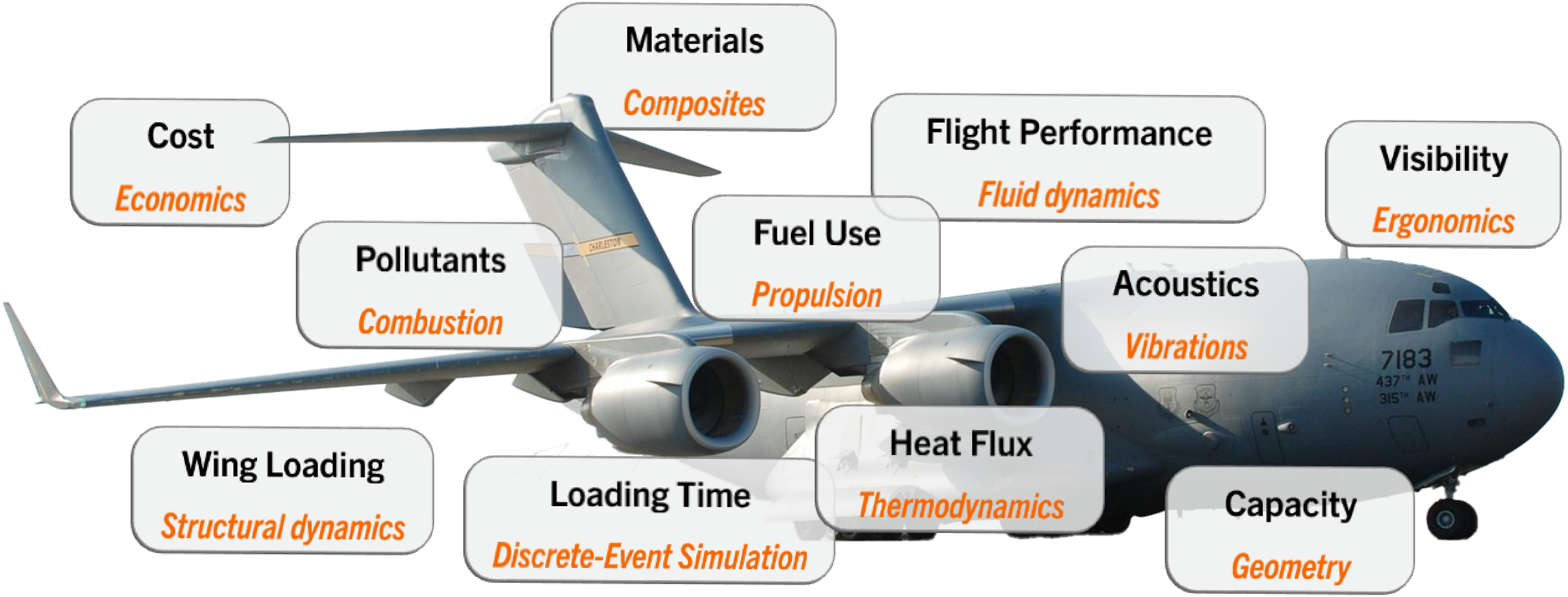
Visiting Researcher at NIST (2025)

PhD: Mech Engr, Clemson (2025)

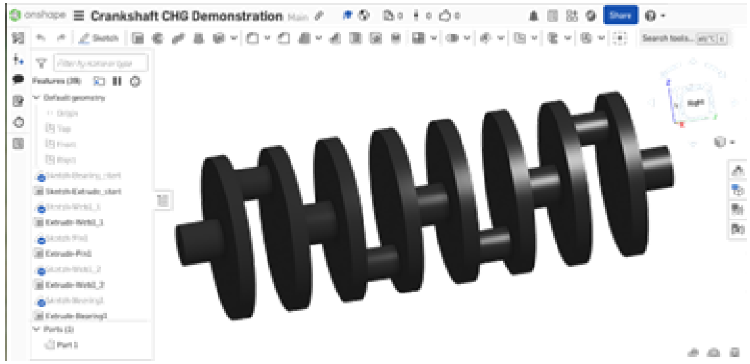
Collaboration with Alan Turing Institute (2026)

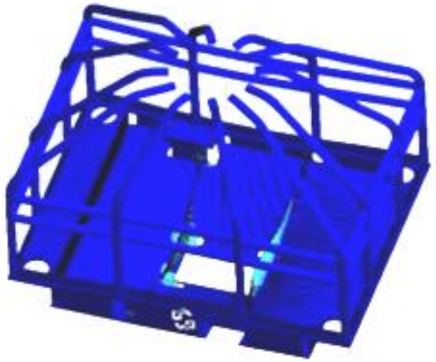
Science is the work of describing the worlds around us

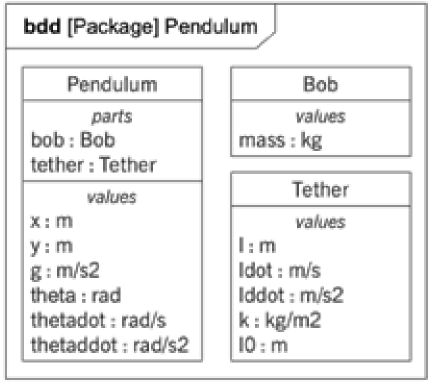
Knowing information about complex systems require models across many domains

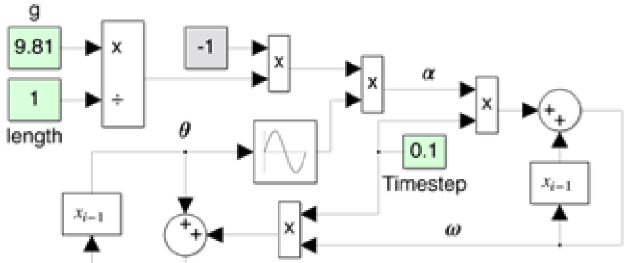


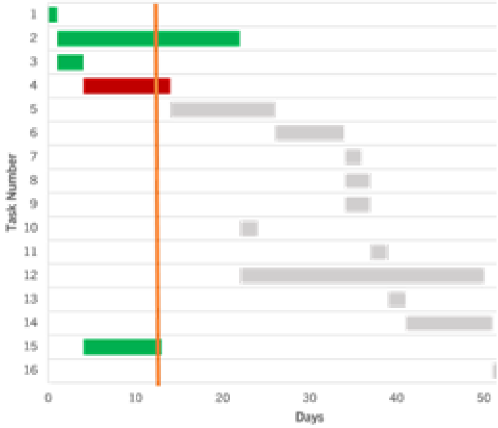
Most modeling frameworks have limited universality and descriptiveness

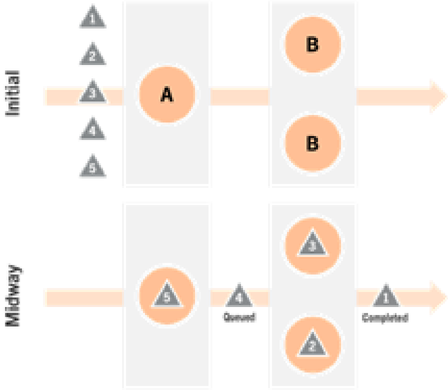


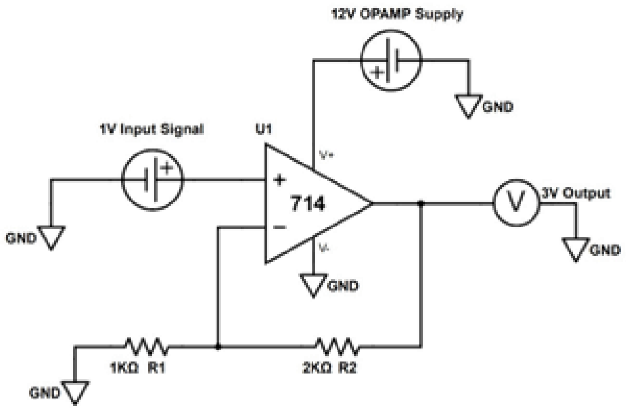


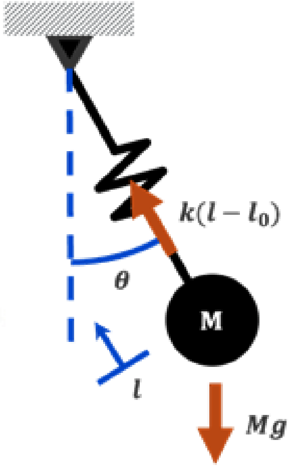






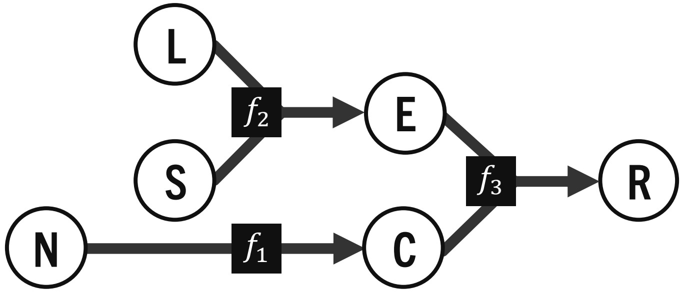


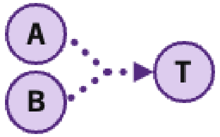


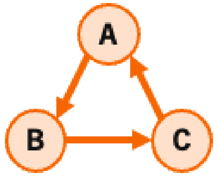


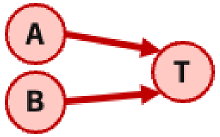
Constraint hypergraphs are a framework for universal, declarative systems modeling

CHG Definition

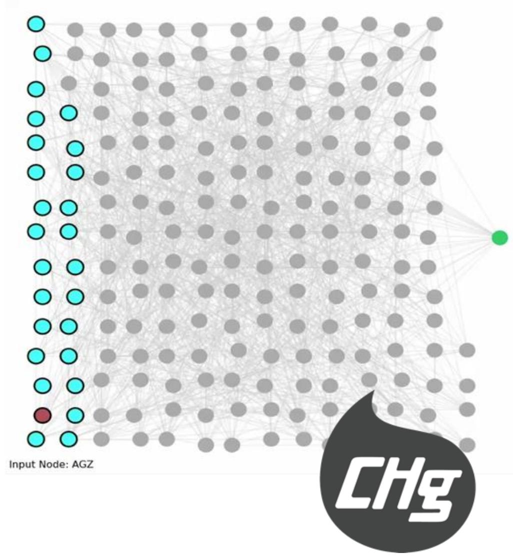




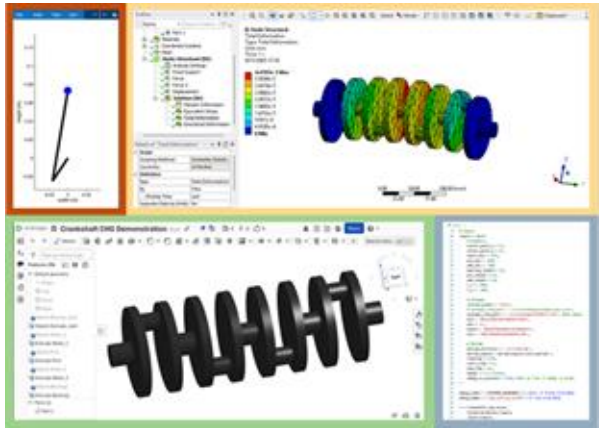


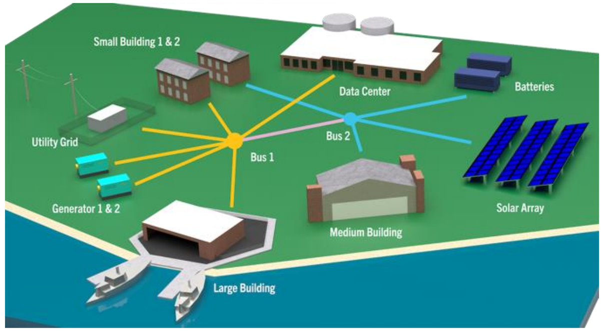


Declarative Simulation



Applications





Introduction to CHGs

Universal System Simulation via Constraint Hypergraphs

Information is what we can distinguish about a system

“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

Systems are described 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

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

(Database lookup)

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

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

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

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

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

$$f_{n\rightarrow \mathrm{r}}(\mathrm{Name}) \rightarrow \mathbf{Range} = f_{ce\rightarrow \mathrm{r}}(f_{n\rightarrow \mathrm{e}}, f_{n\rightarrow \mathrm{c}})$$

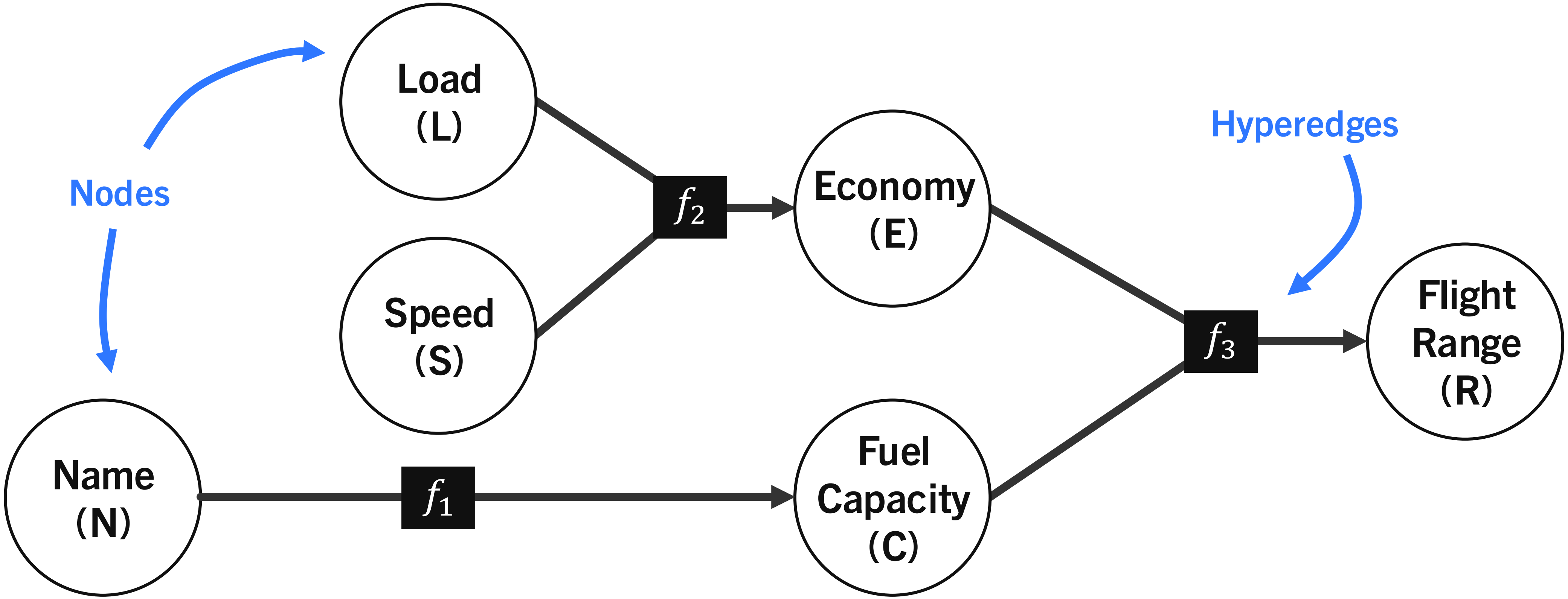
Name
C-17 Globemaster
C-5 Galaxy
C-130 Hercules

Economy (L/km)
10
11
12

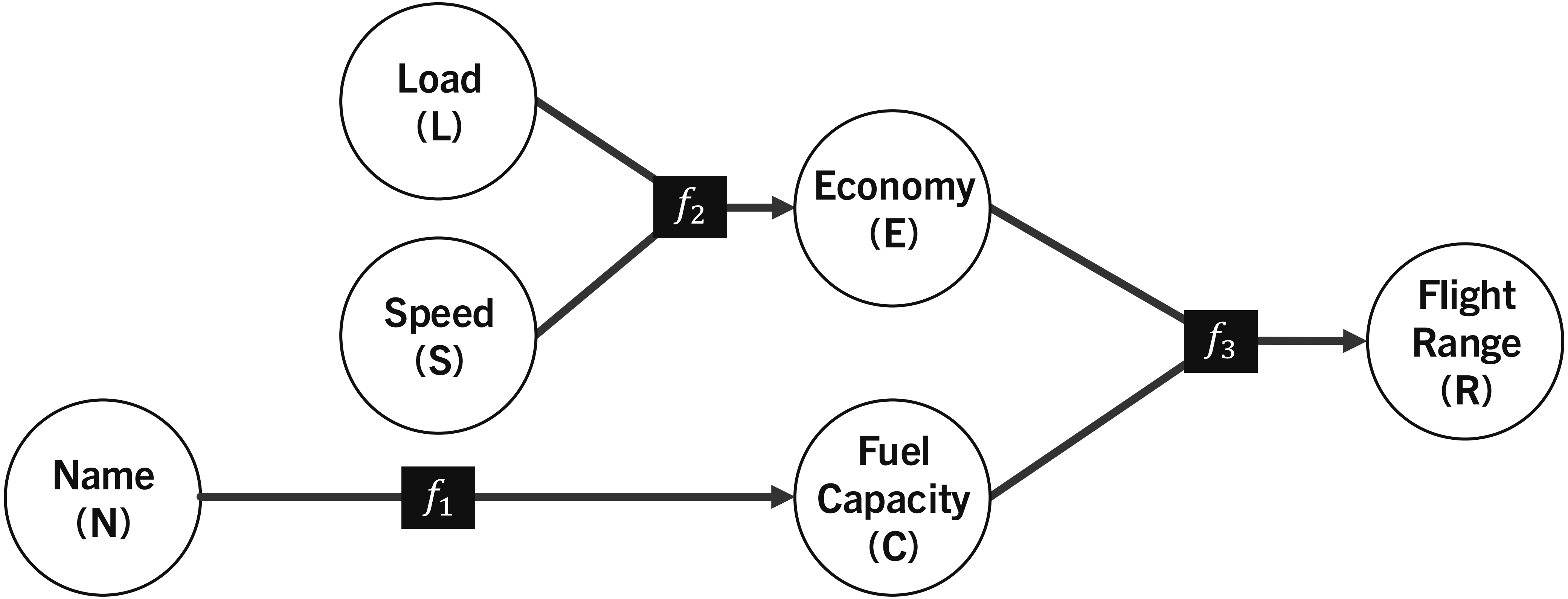
Fuel Capacity (L)
100,000
115,000
130,000

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

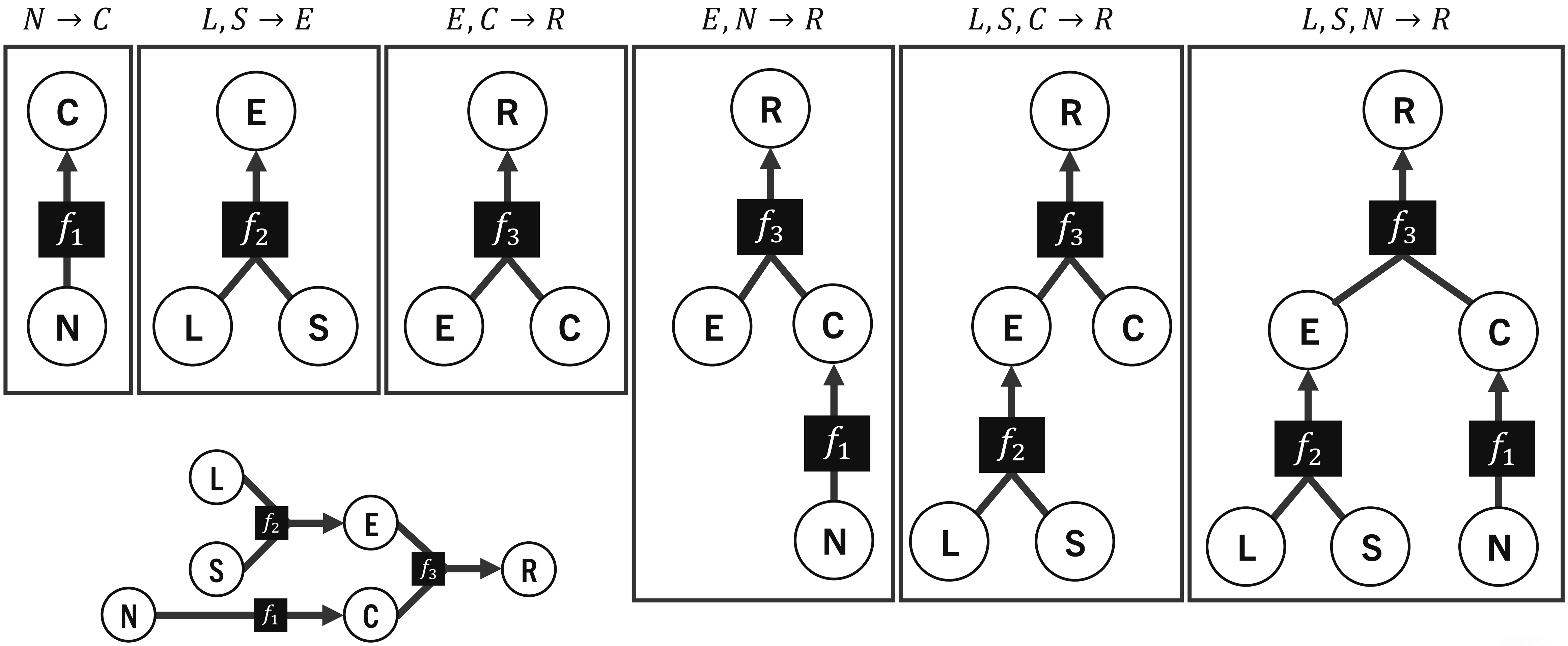
We can represent all models and variables as a constraint hypergraph (CHG)



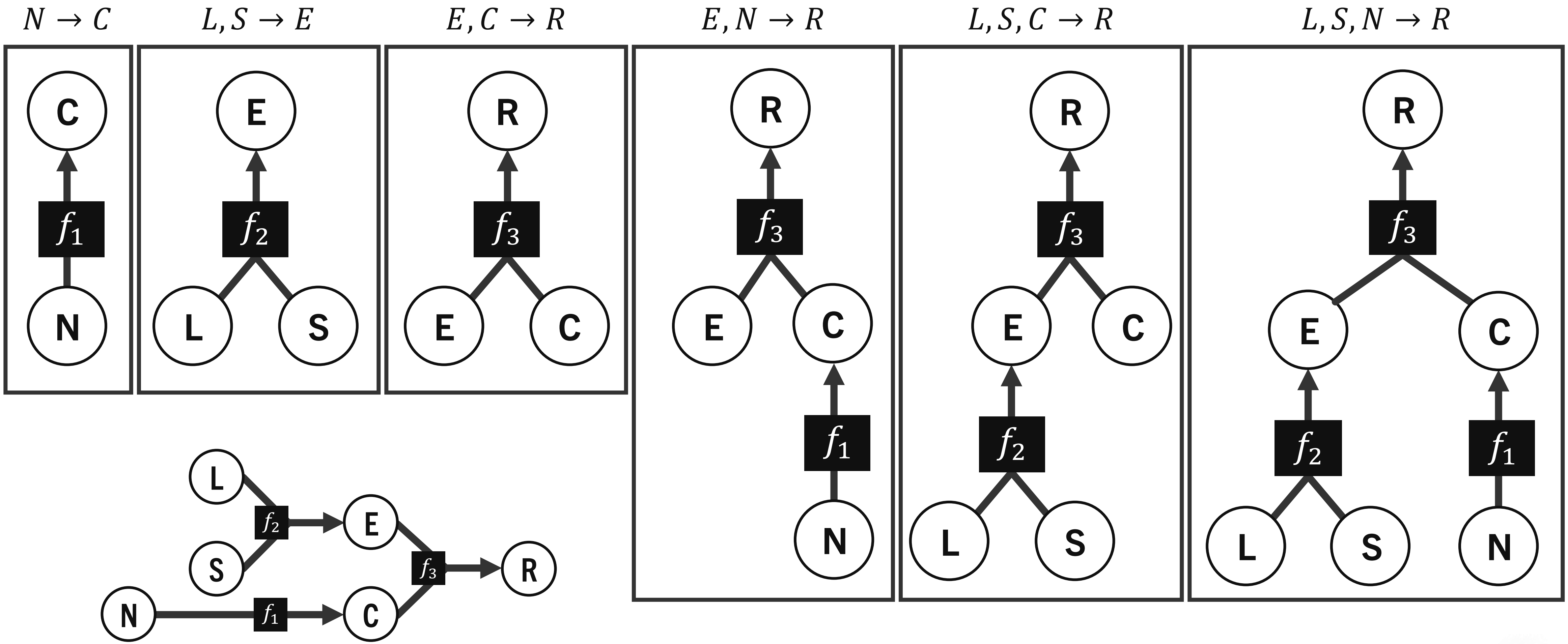
Every path in a CHG is a simulation



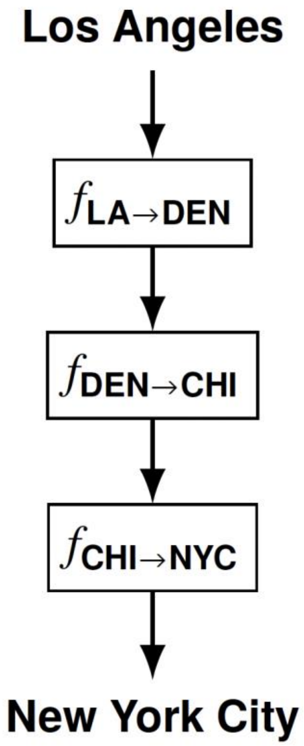
Every path in a CHG is a simulation



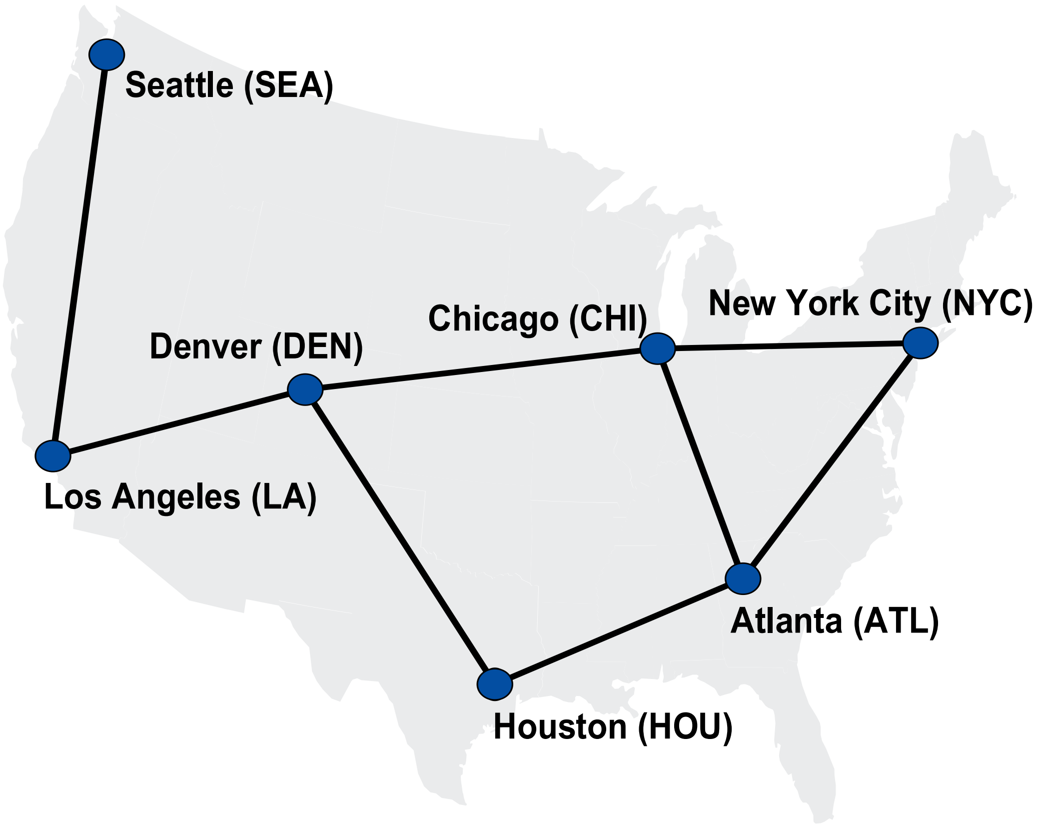
Embedded behavior (paths) enables declarative simulation



Declarative models capture the semantics of the system

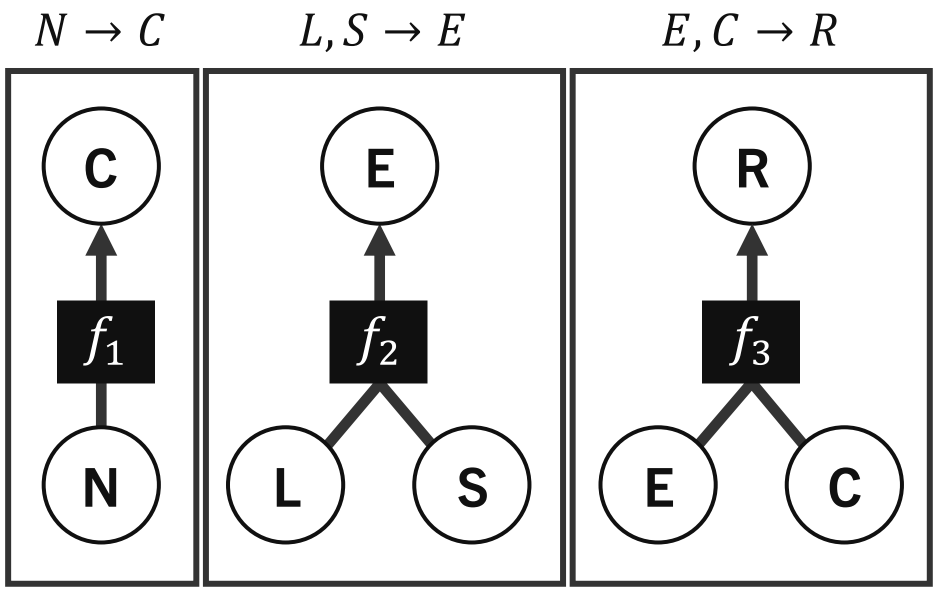


Imperative

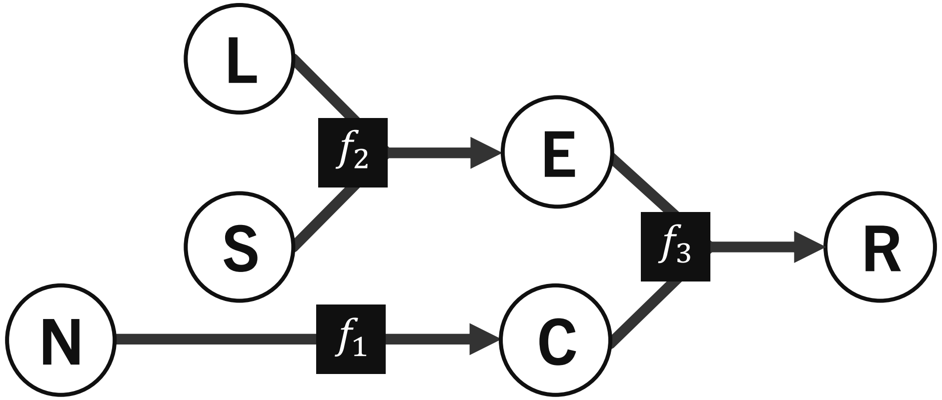


Declarative

Declarative simulation in a CHGs is pathfinding

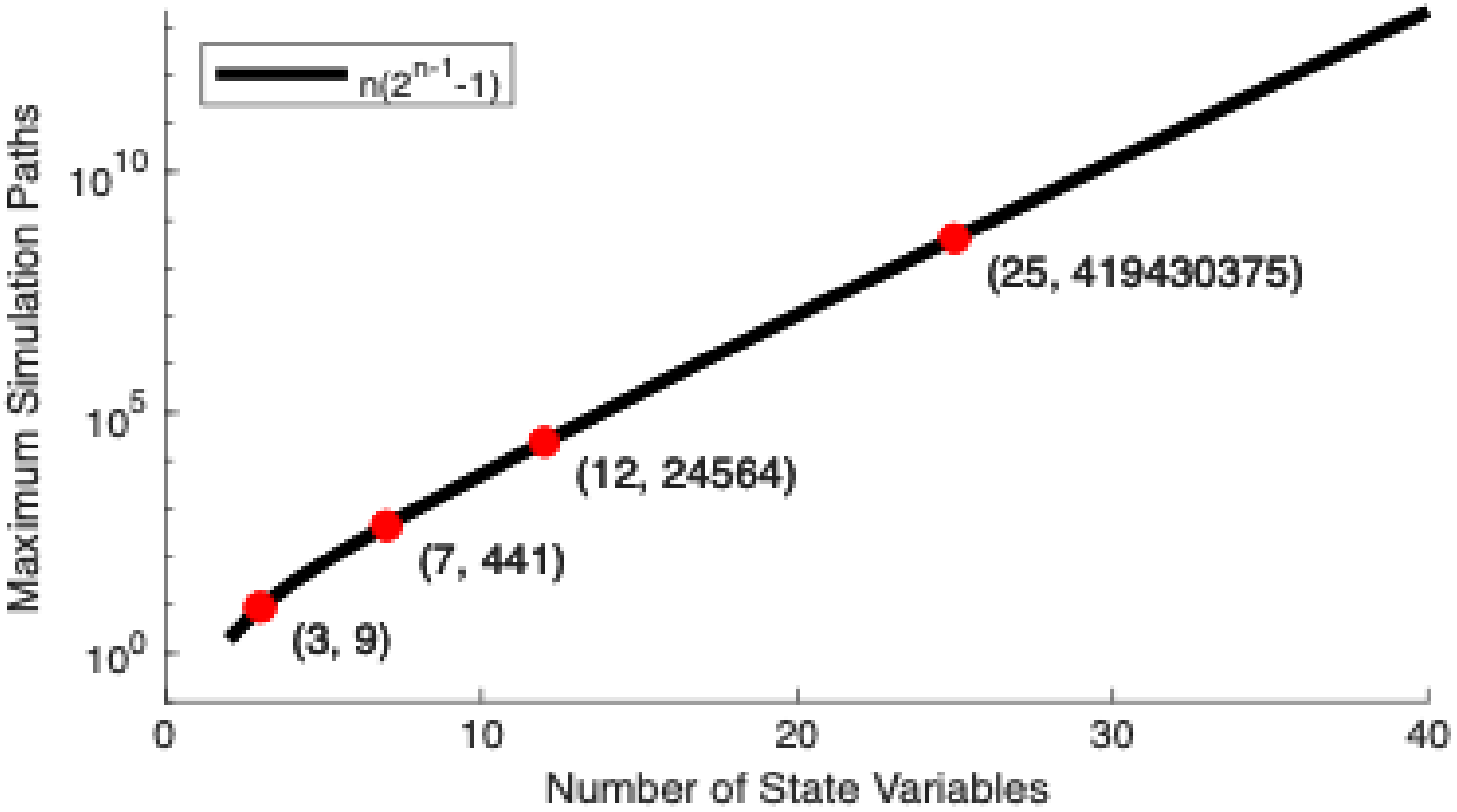


Imperative



Declarative

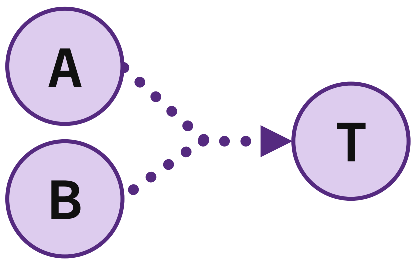
CHGs reduce complexity of expressing simulation paths



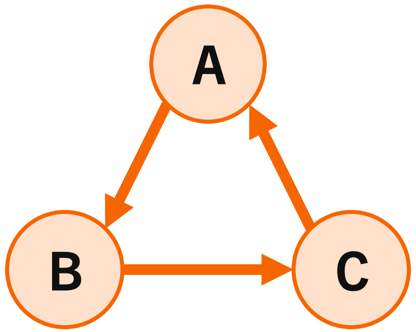
Structure of a CHG

Universal System Simulation via Constraint Hypergraphs

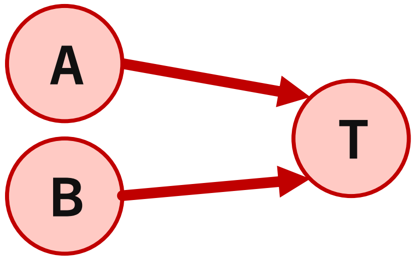
There are three structures in a CHG that a declarative agent needs to process:



Partial Edges



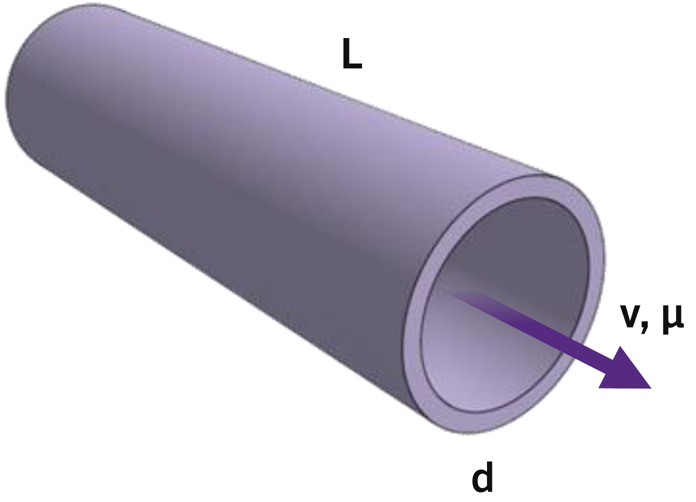
Cycles

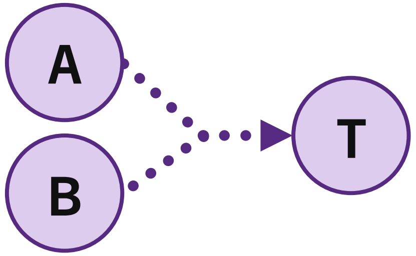


Multi-Edges

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

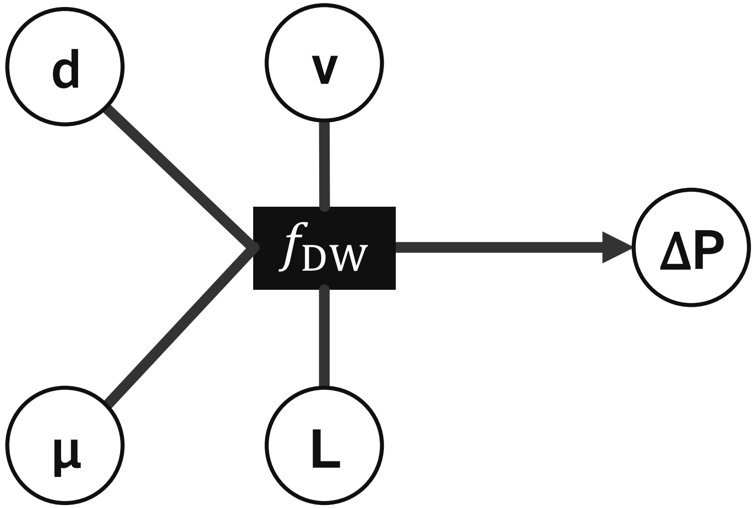
Partial edges occur when we don't have an explicit mapping for every variable

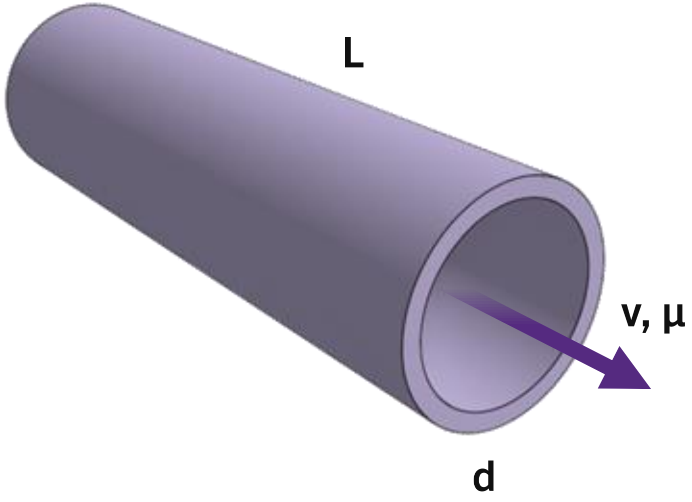


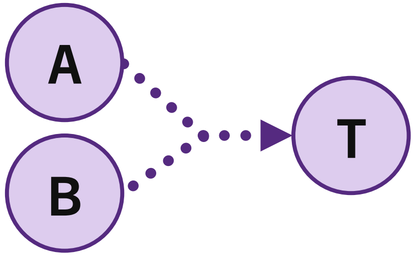


Goal is to calculate pressure loss from fluid flow in a pipe

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

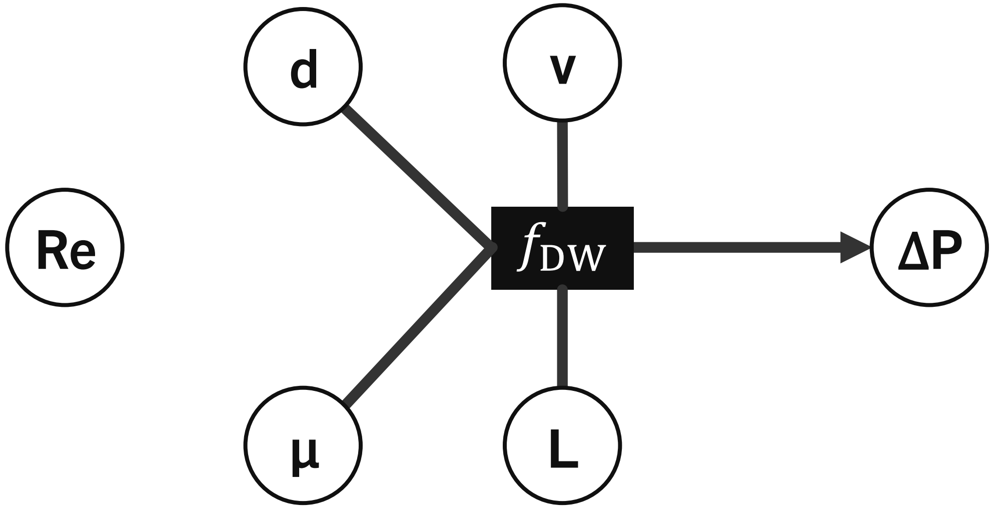


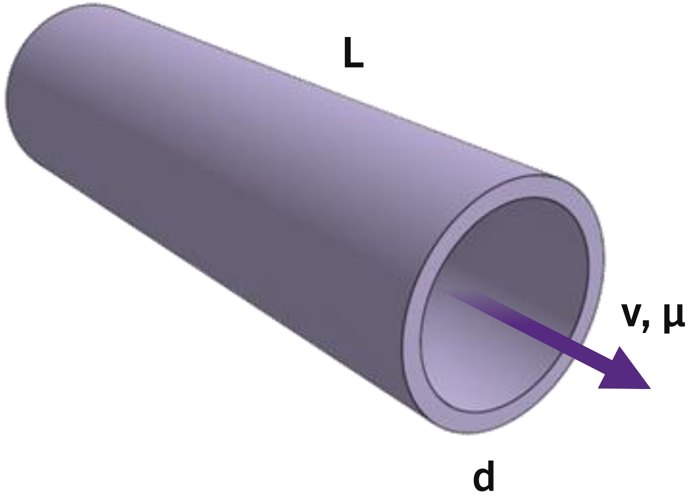


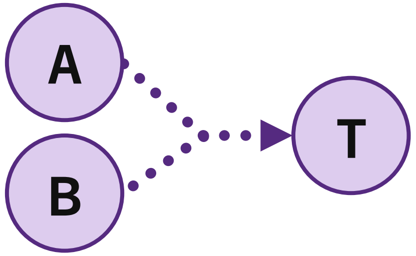


The hypergraph captures the Darcy-Weisbach relationship $\Delta P = \frac{32\mu v L}{\mathrm{d}^2}$

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

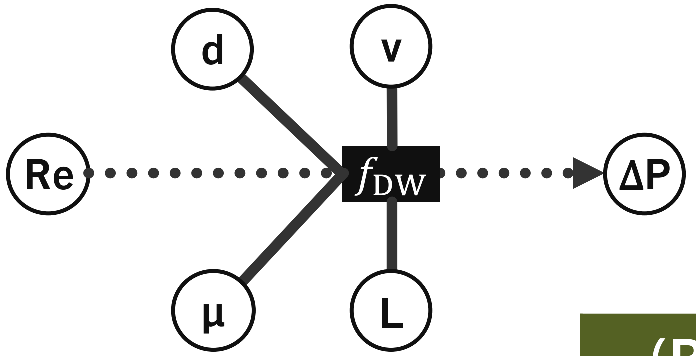






This relationship is only valid if the flow is laminar (Re < 2300)

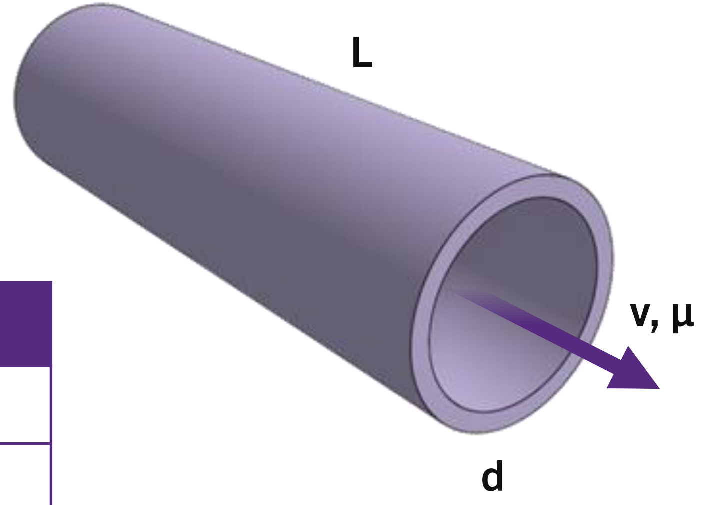
Partial edges occur when we don't have an explicit mapping for every variable

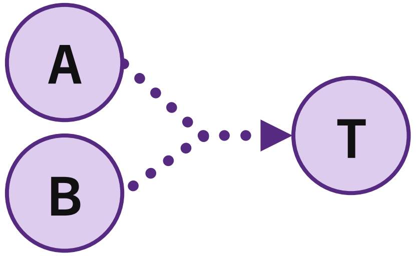


(Re, L, μ, d, v)
…
(1000, L, μ, d, v)
(2000, *L*, *μ*, *d*, *v*)
(3000, *L*, *μ*, *d*, *v*)
…

f_{DW}

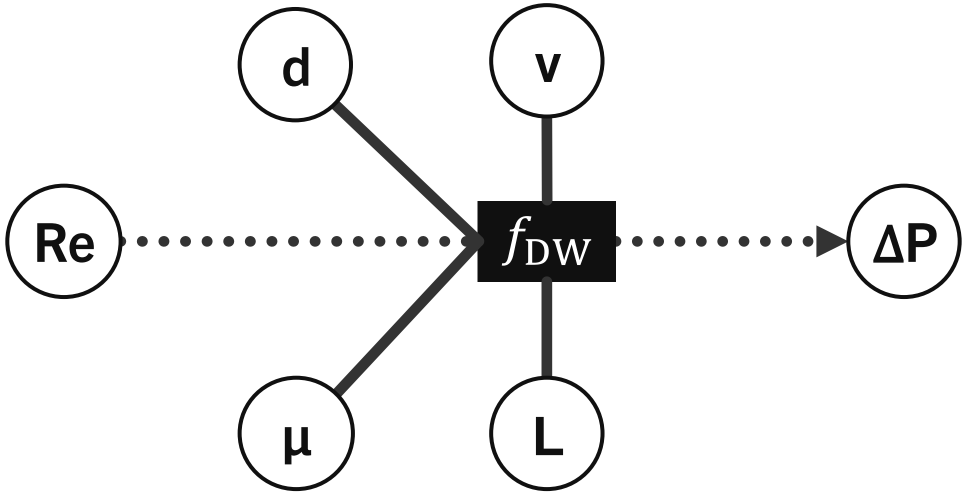
ΔP
…
0.5
1
1.5
…

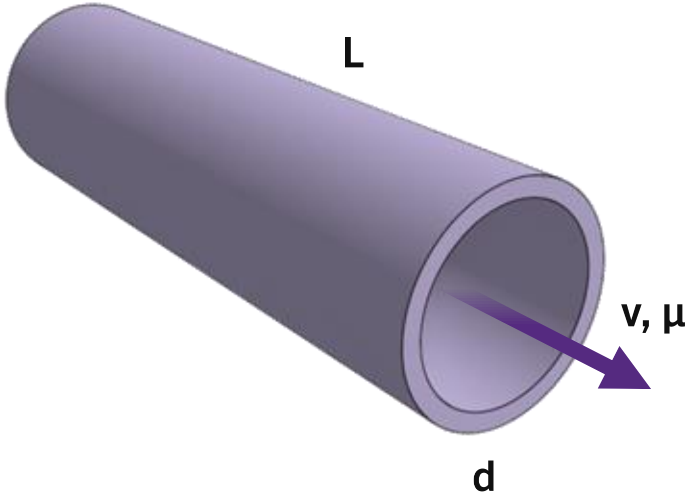


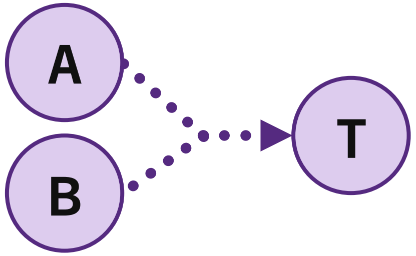


We show this by mapping a subset of the domain

Partiality means that functions don't automatically compose

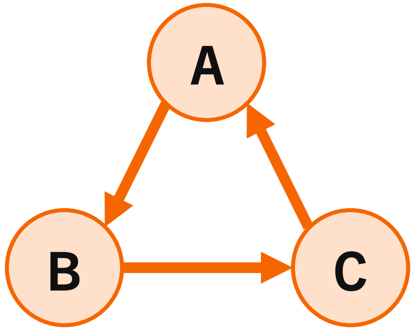






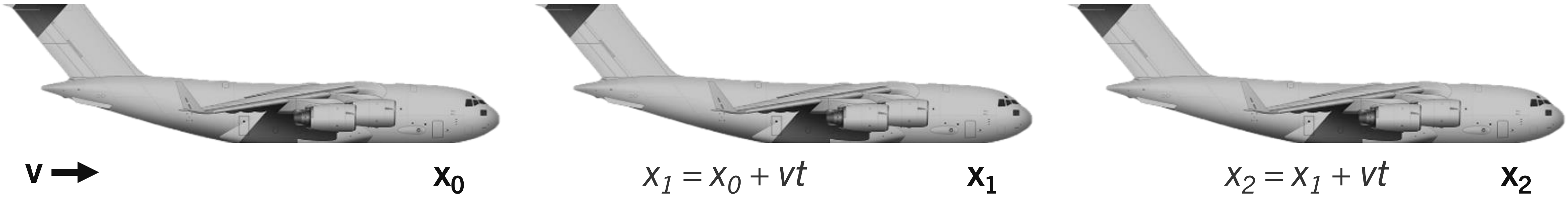
The dotted line shows that our function doesn't handle every value of Re

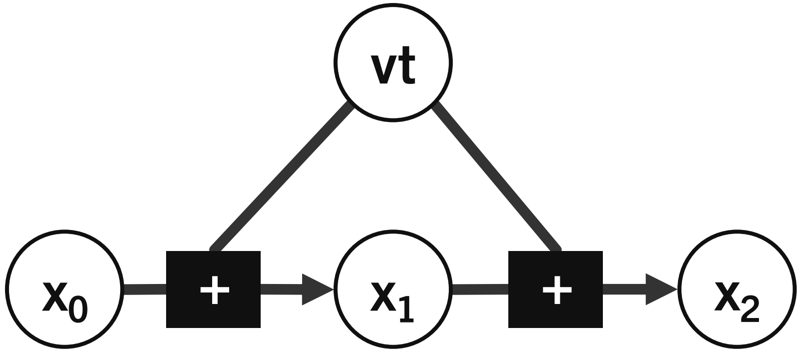
Cycles indicate behavioral patterns



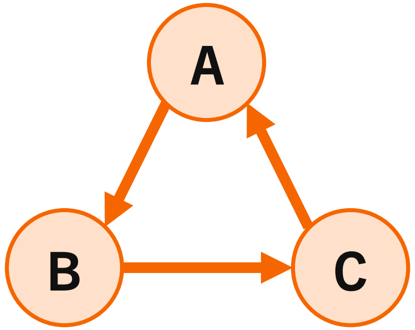
Plane moving at constant velocity

Cycles indicate behavioral patterns

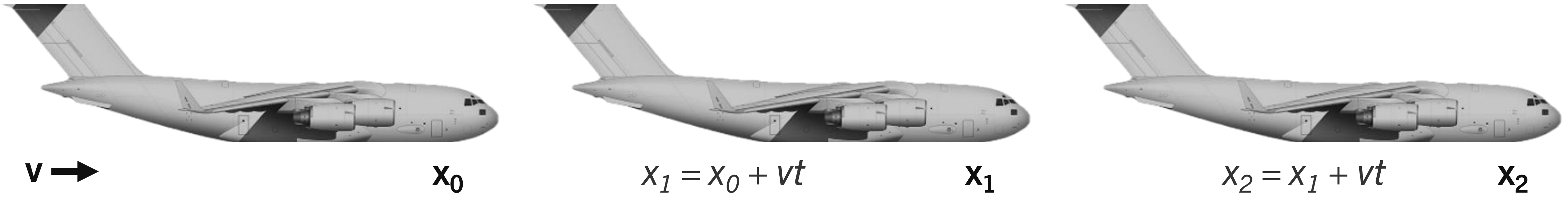


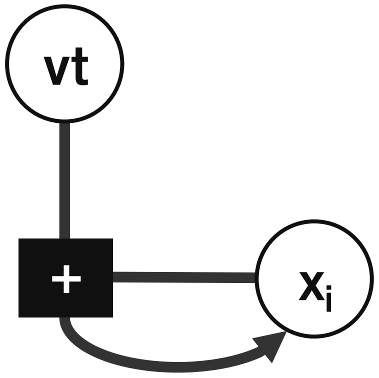


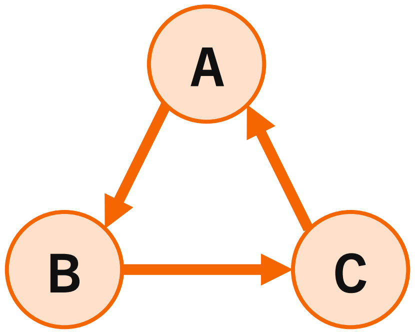
x_0, x_1, and x_2 are all different nodes



Cycles indicate behavioral patterns

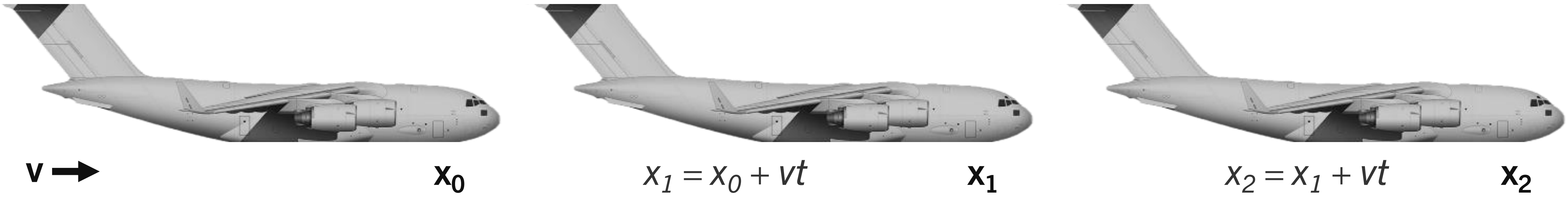


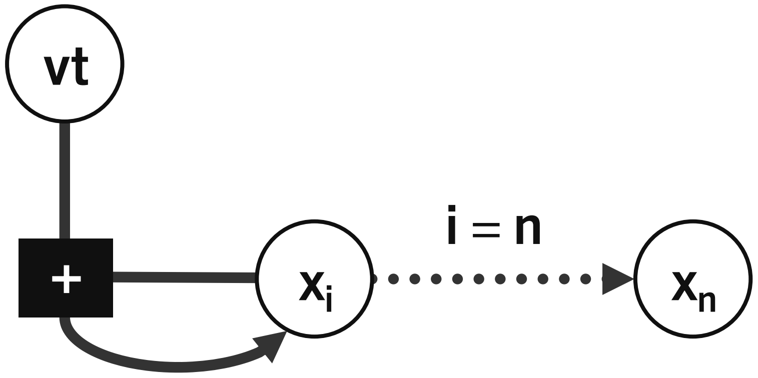


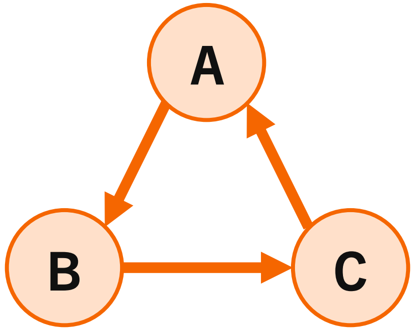


A more expressive way to show this is $x_{i+1} = x_i + vt$ (unraveling)

Unraveling requires an exit condition to be solvable

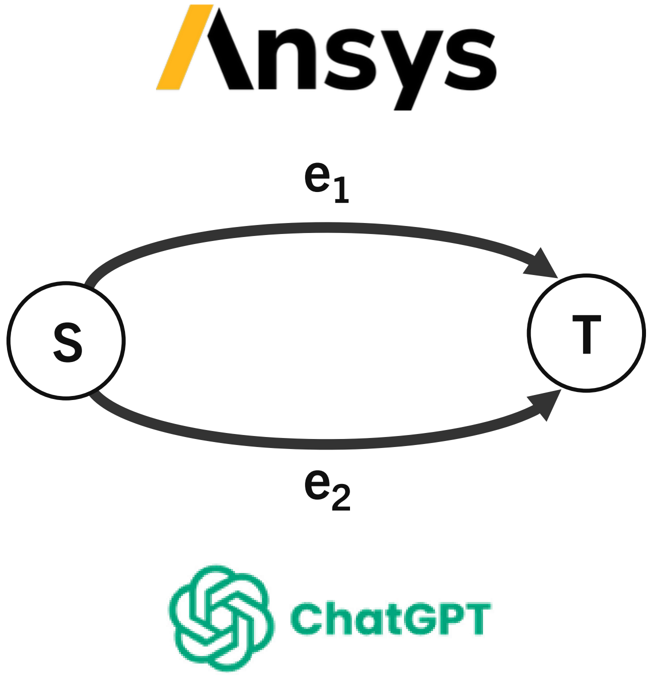


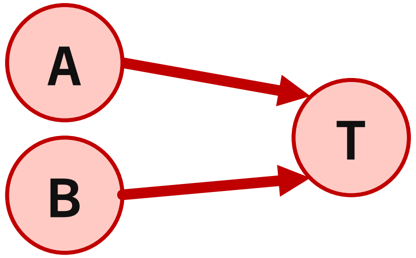




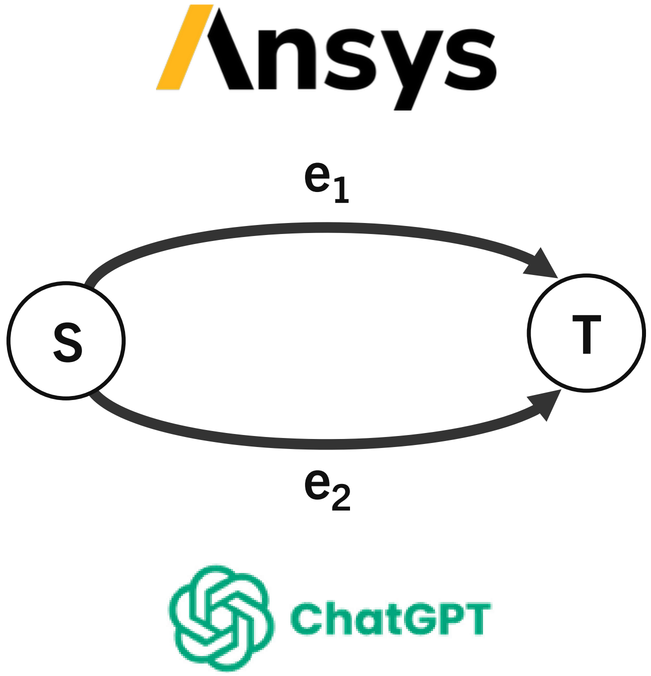
The partial edge shows which edges of x_i are mapped to x_n

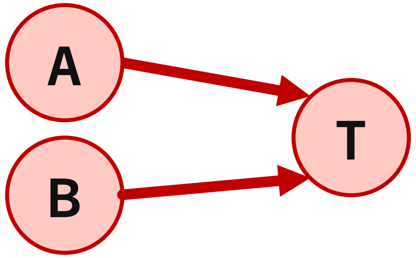
Multi-edges indicate competing models





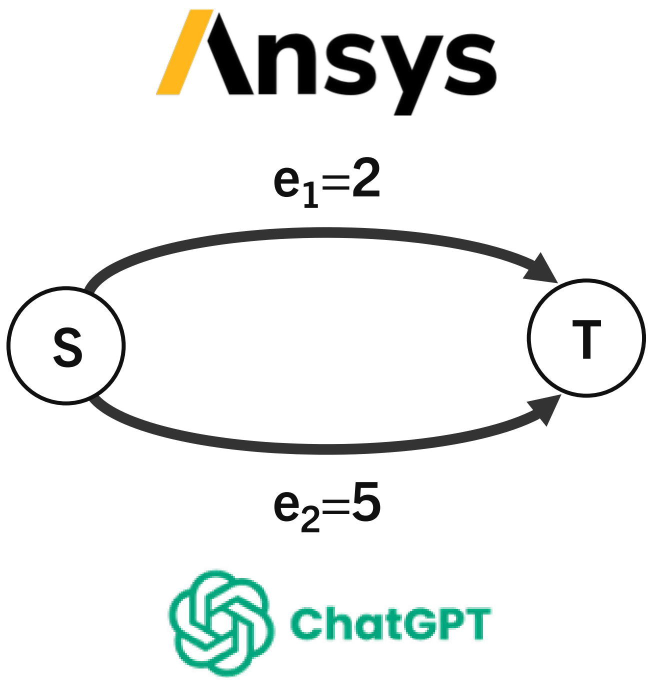
Both edges are viable ("correct"), but only one can be selected for simulation

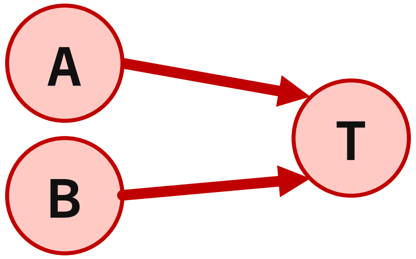




Differences might be on accuracy, computation time, software availability, etc.

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

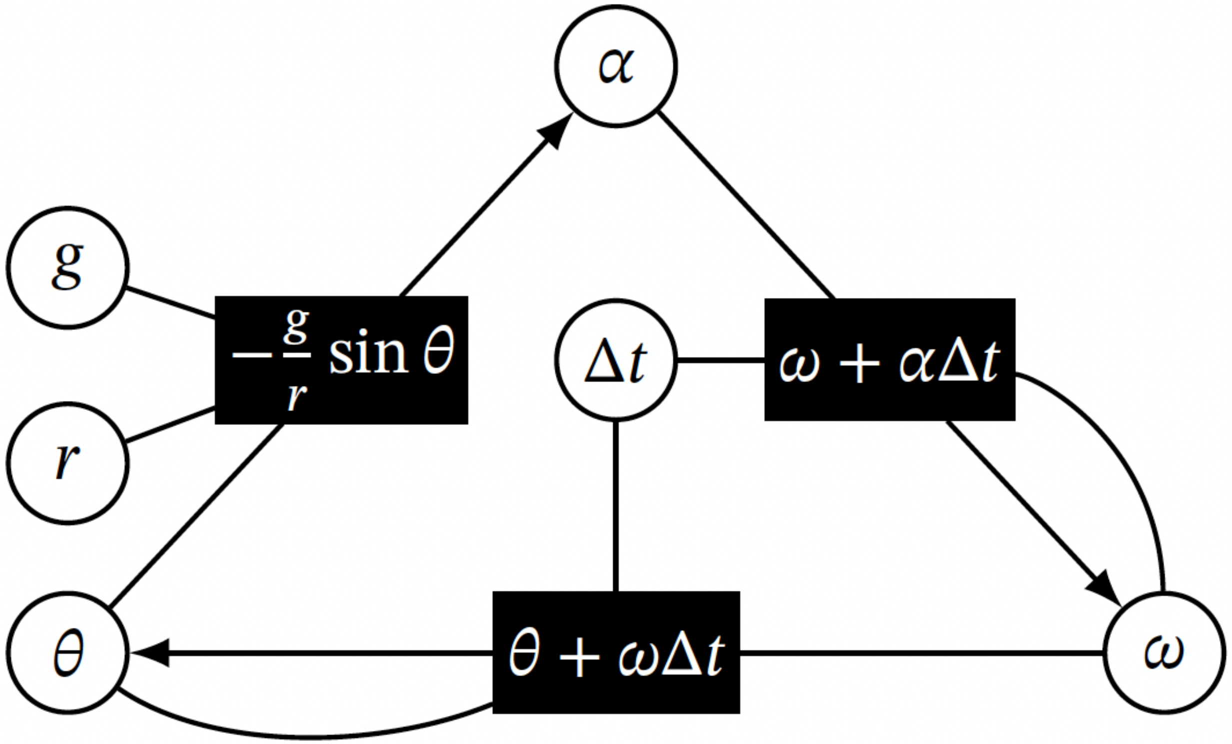


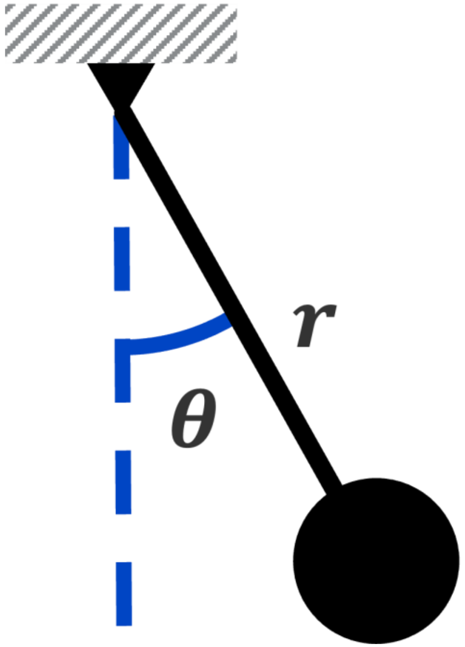


Demonstration

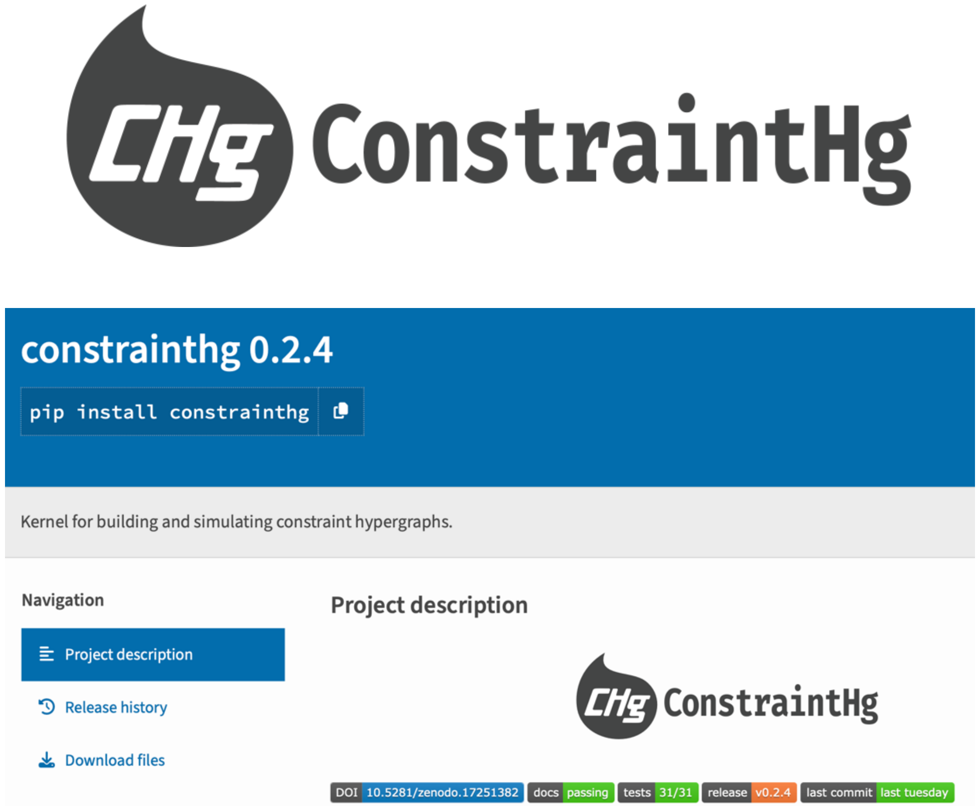
Universal System Simulation via Constraint Hypergraphs

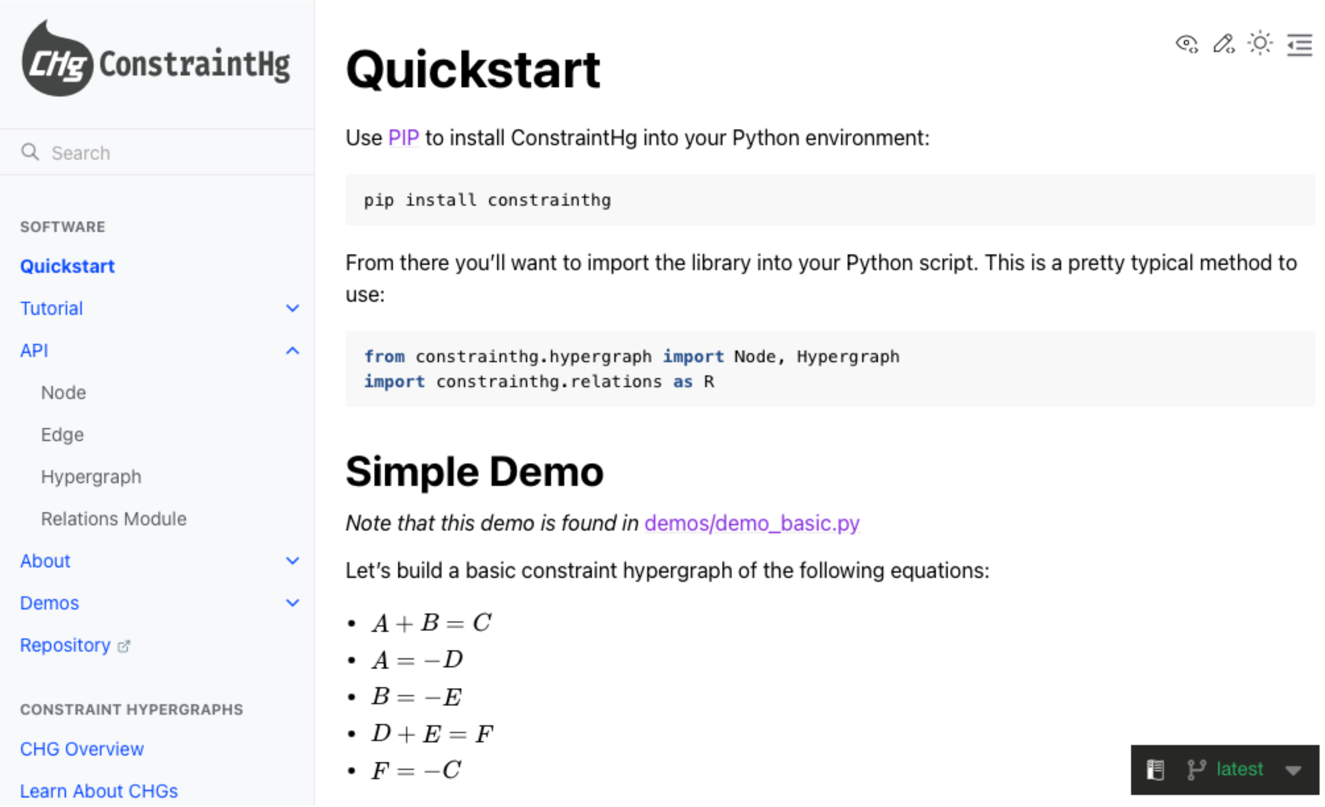
Pendulum System





Declarative simulation provided by ConstraintHg

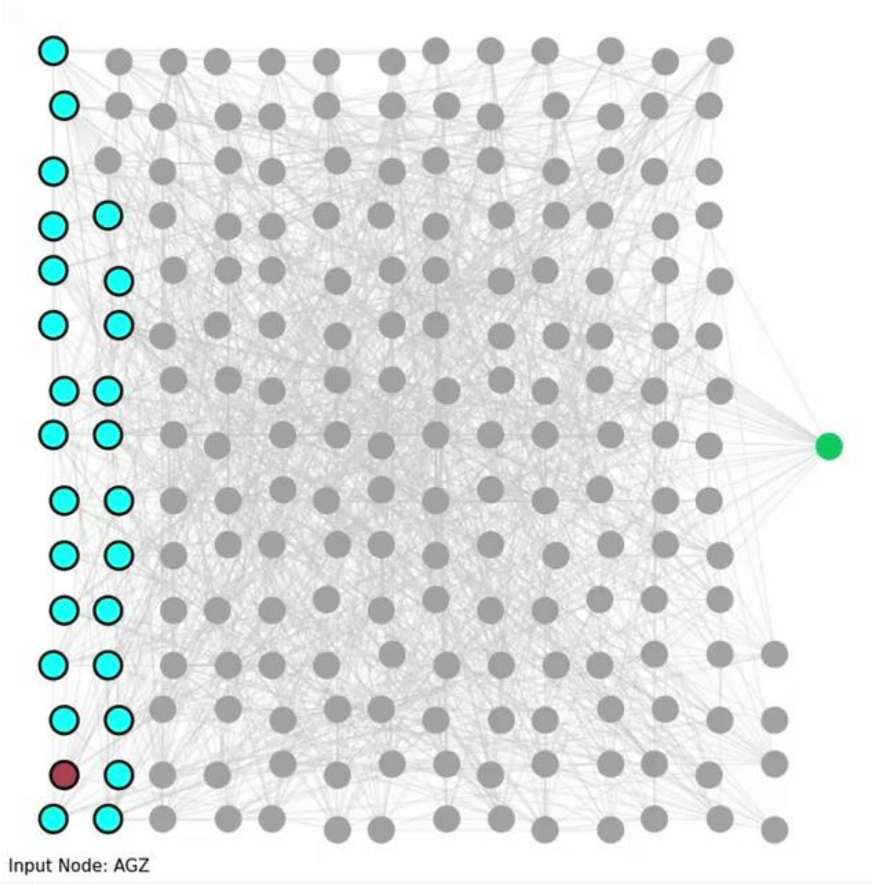




ConstraintHg Functionality

Provides methods for:

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



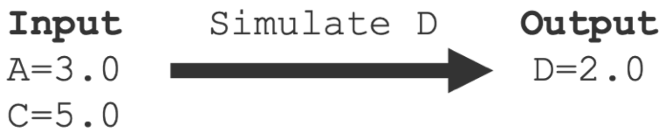
Process of making a CHG

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

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

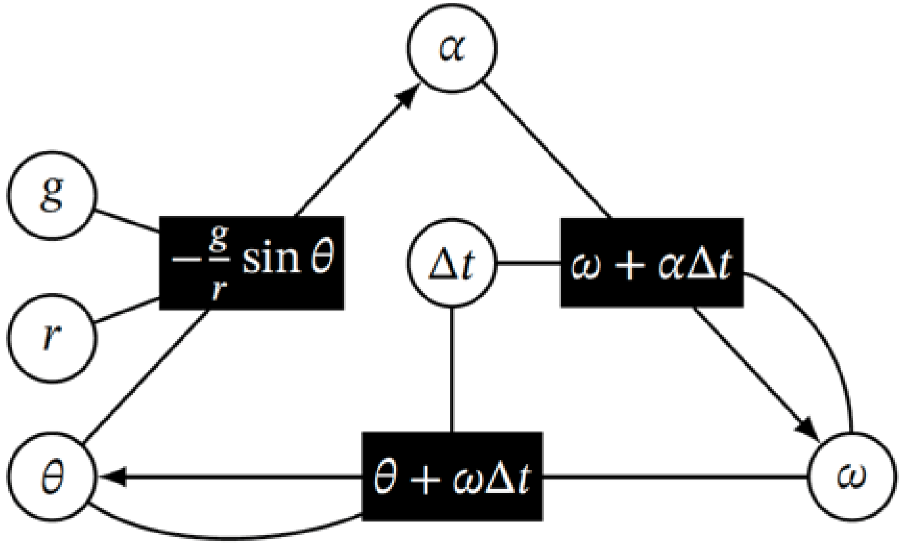
Declarative simulation requires only start & end points

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

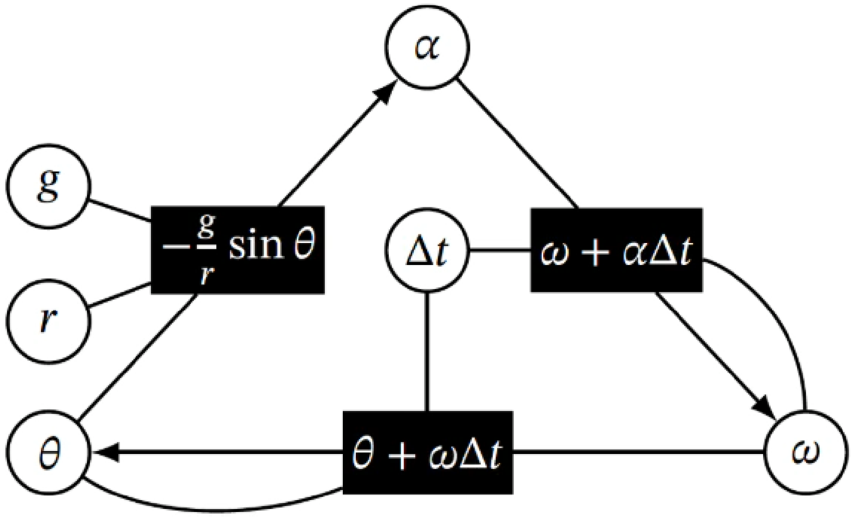
Simulation call

```
└──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



```python
"""
Demonstration of ConstraintHg for a simple pendulum.
More examples and information available online at constrainthg.readthedocs.io
"""

Run Cell | Run Below | Debug Cell
# %%
# Setup constraint hypergraph
from constrainthg import Hypergraph
from math import sin

# Define relational rules as methods in Python
def Rtheta_to_alpha(theta, g, r):
    return -g / r * sin(theta)

def Rintegrate(initial, slope, step):
    return initial + slope * step

def Rcheck_increment(initial, slope):
    return slope == initial + 1

# Initialize constraint hypergraph and add edges
hg = Hypergraph()

hg.add_edge(
    sources=dict(theta='theta', g='g', r='r'), target='alpha',
    rel=Rtheta_to_alpha, index_offset=1,
    label='-g/r*sin(theta)'
)

hg.add_edge(
    sources=dict(initial='omega', slope='alpha', step='del_t'), target='omega',
    rel=Rintegrate, index_via=Rcheck_increment, disposable=['initial', 'slope'],
    label='Integrate alpha'
)

```



Interactive-1

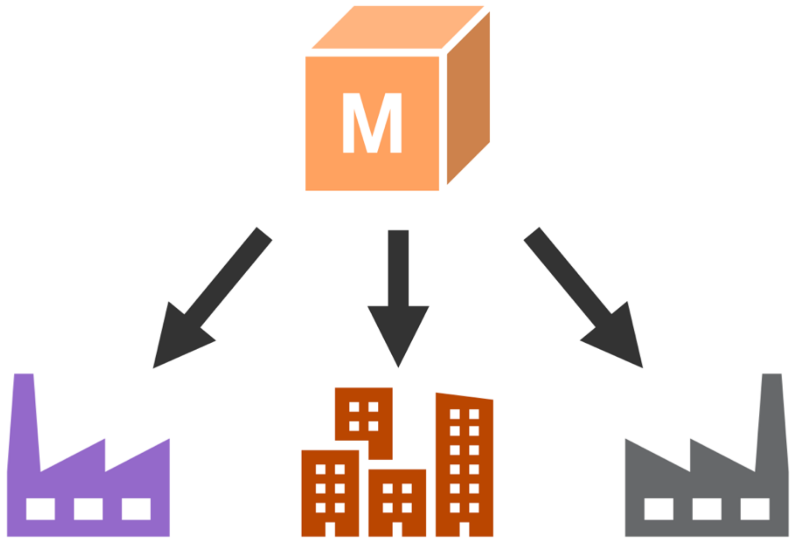
Interrupt | Clear All | Restart | Jupyter Variables | Save | ··· | .venv (

Applications of CHGs

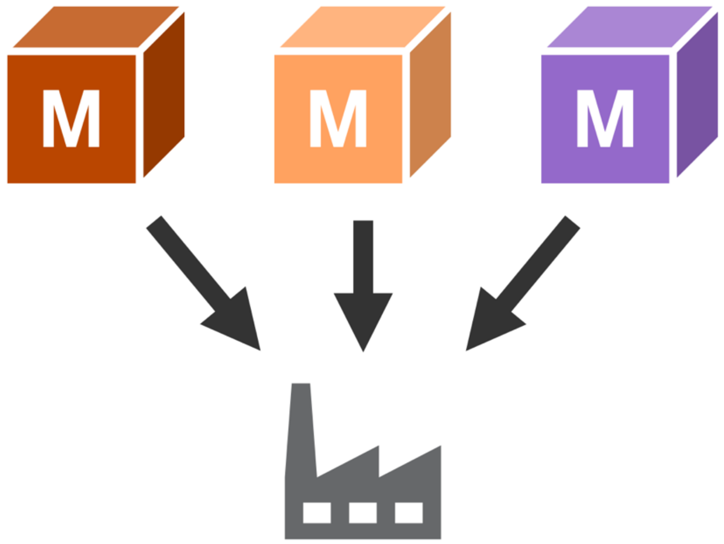
Universal System Simulation via Constraint Hypergraphs

CHGs provide platforms for model-based engineering and digital twins

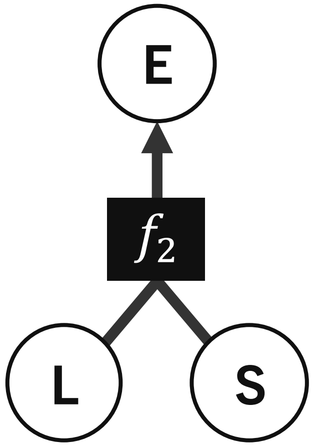
Maintains information in different contexts



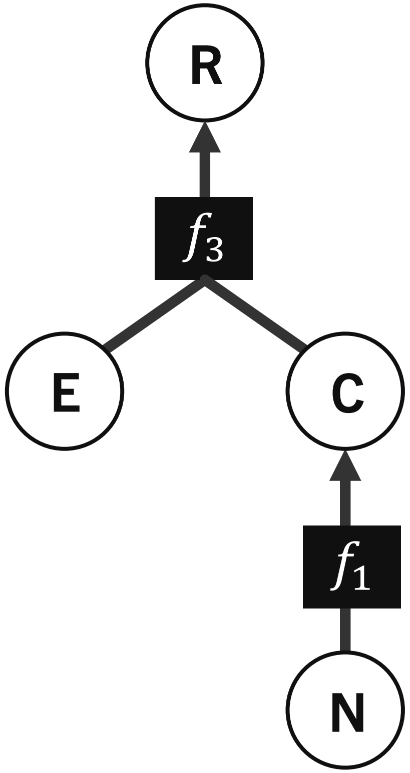
Maintains information with different models



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

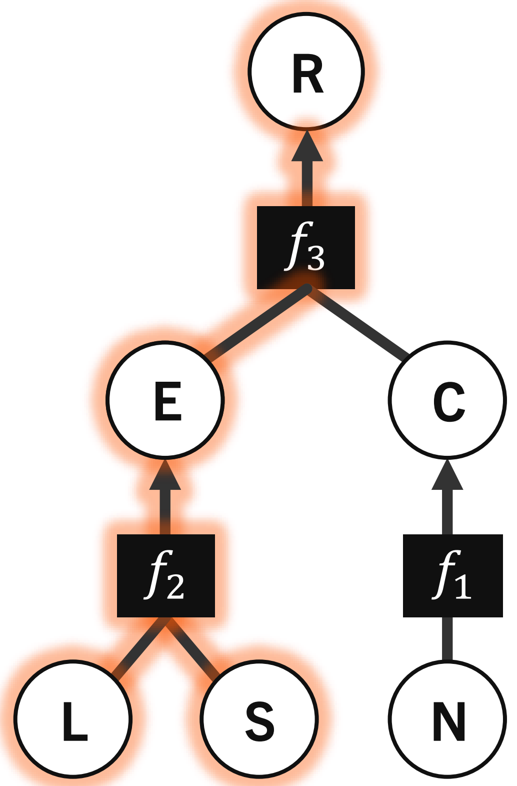


Graph 1



Graph 2

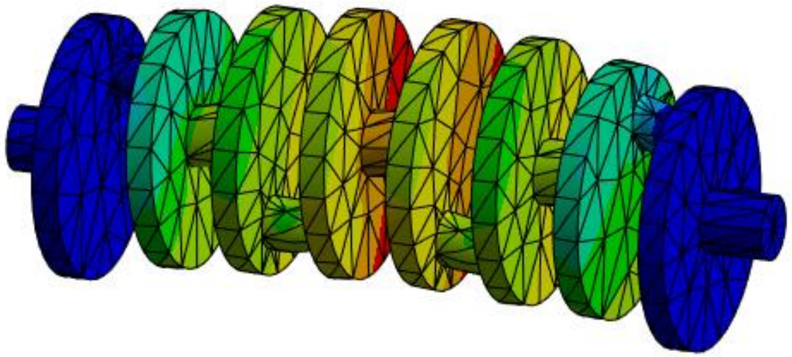
New behaviors can be expressed as new paths in the union of two CHGs



Graph 1 ∪ 2

New behaviors include simulation paths that go across calculating software

Material Mechanics

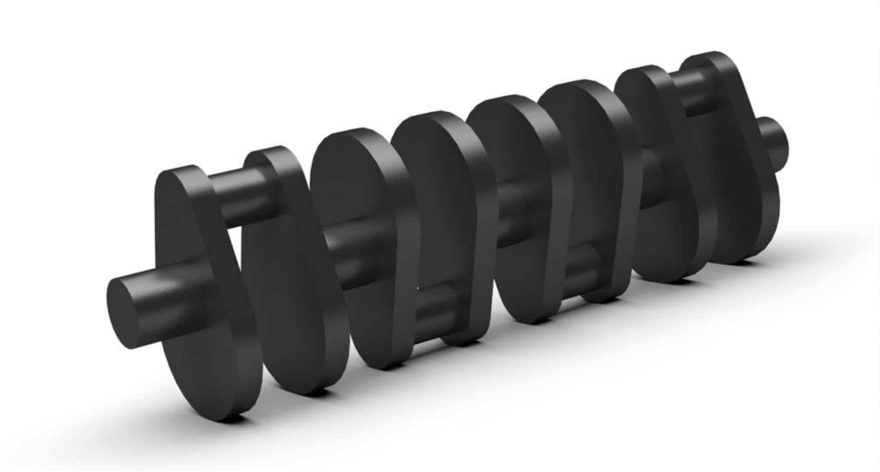


Ansys

Kinematic Analysis

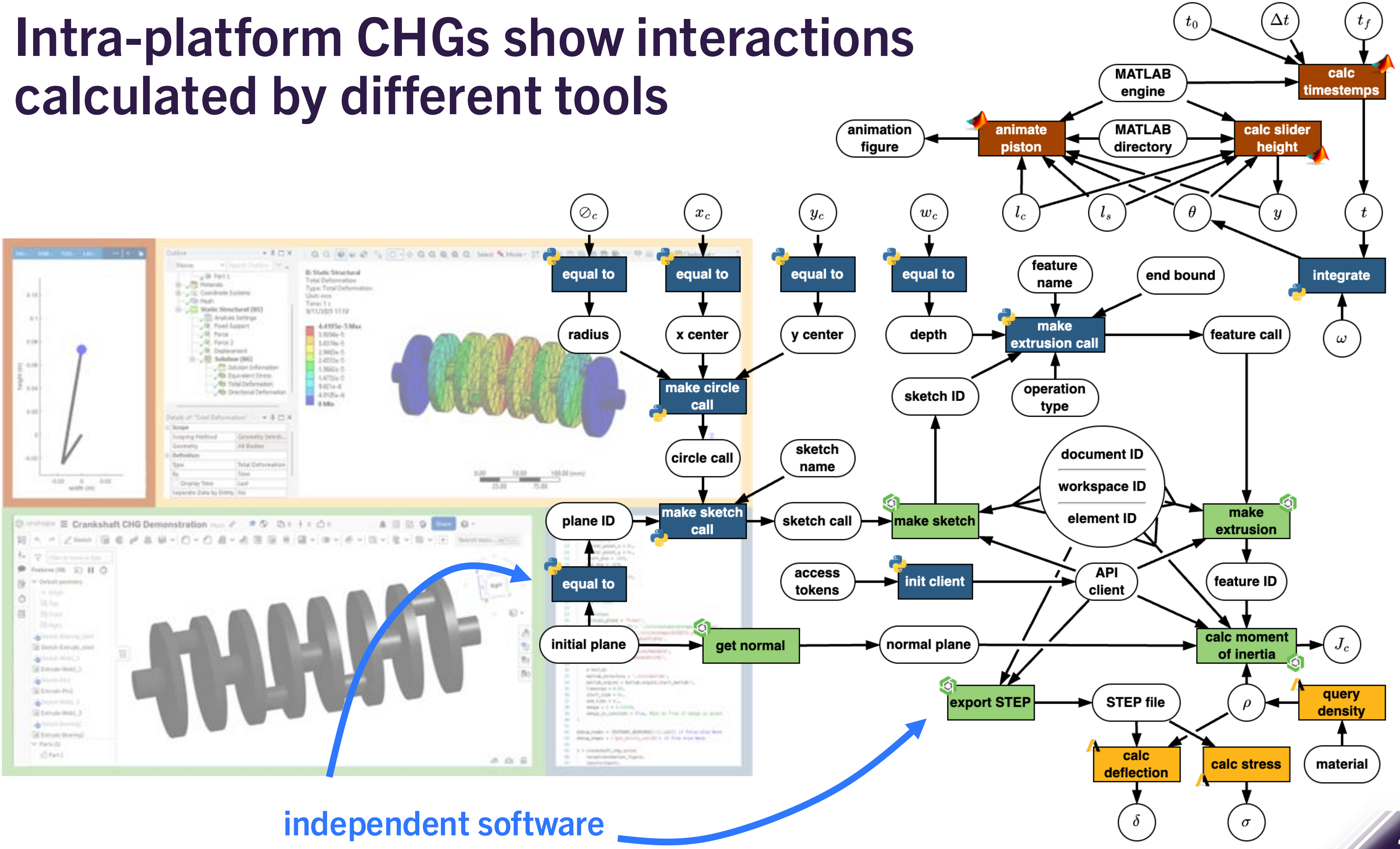
MATLAB & SIMULINK

Mass Properties

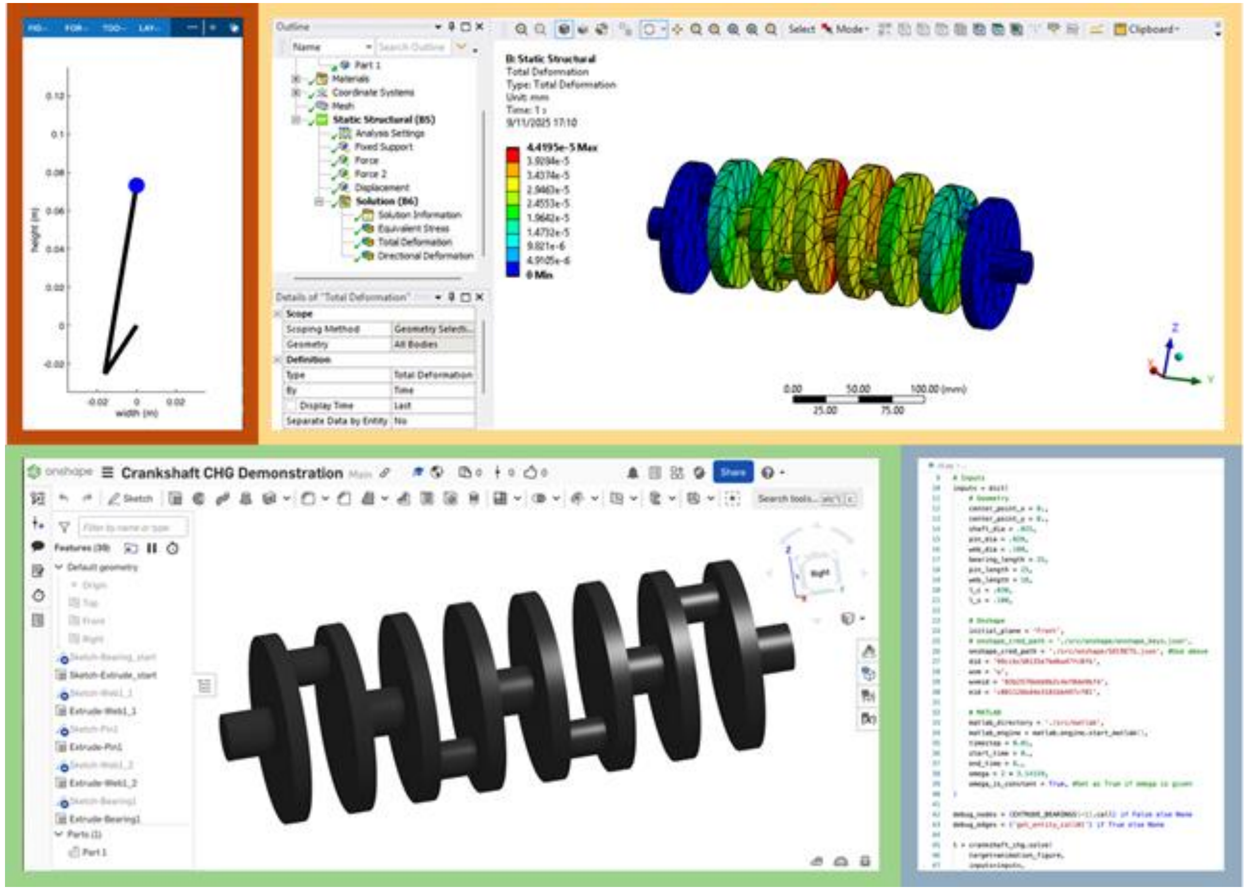


onshape

Intra-platform CHGs show interactions calculated by different tools



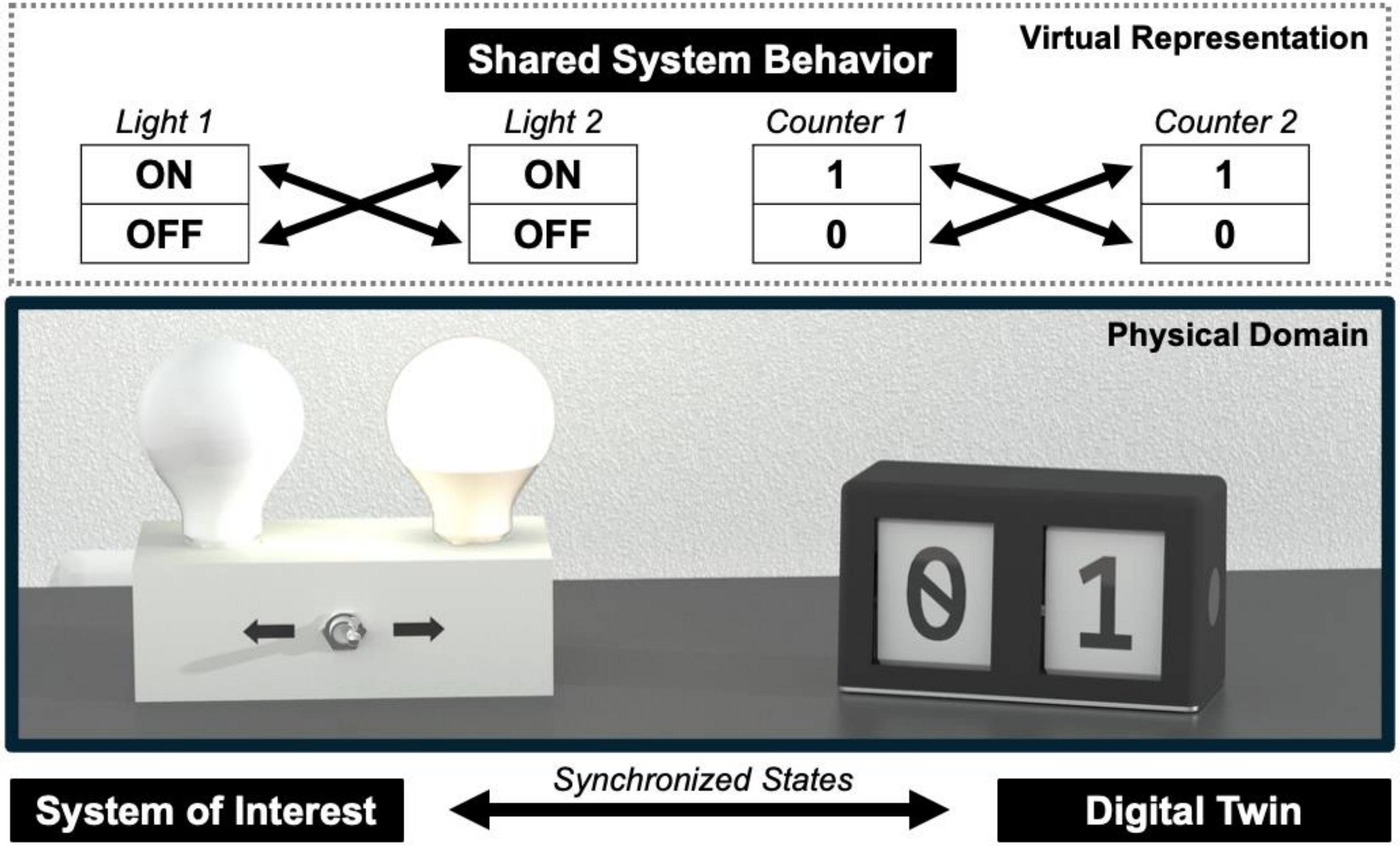
CHGs resolve inter-application APIs into a cohesive ecosystem



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)

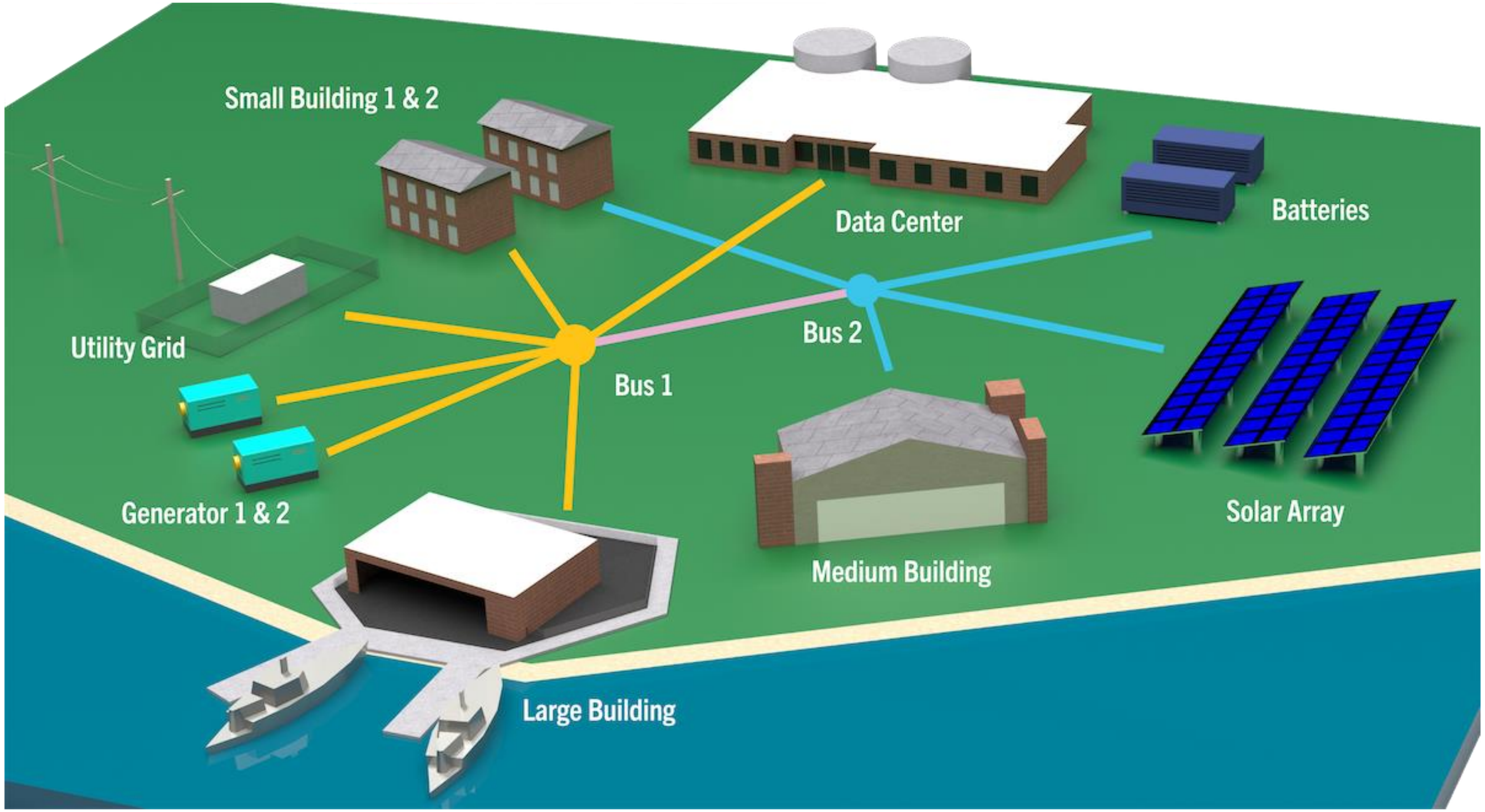
DTs are systems synchronized with another system of interest

Digital Twins are 2 systems sharing the same CHG

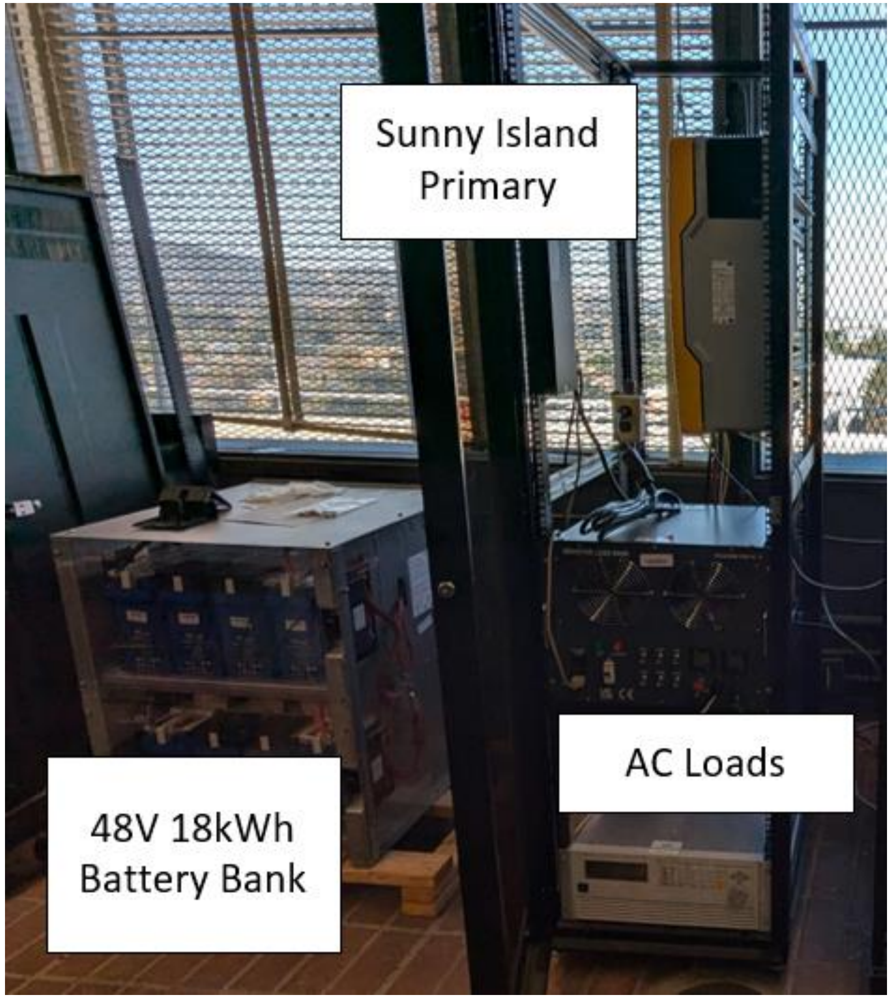


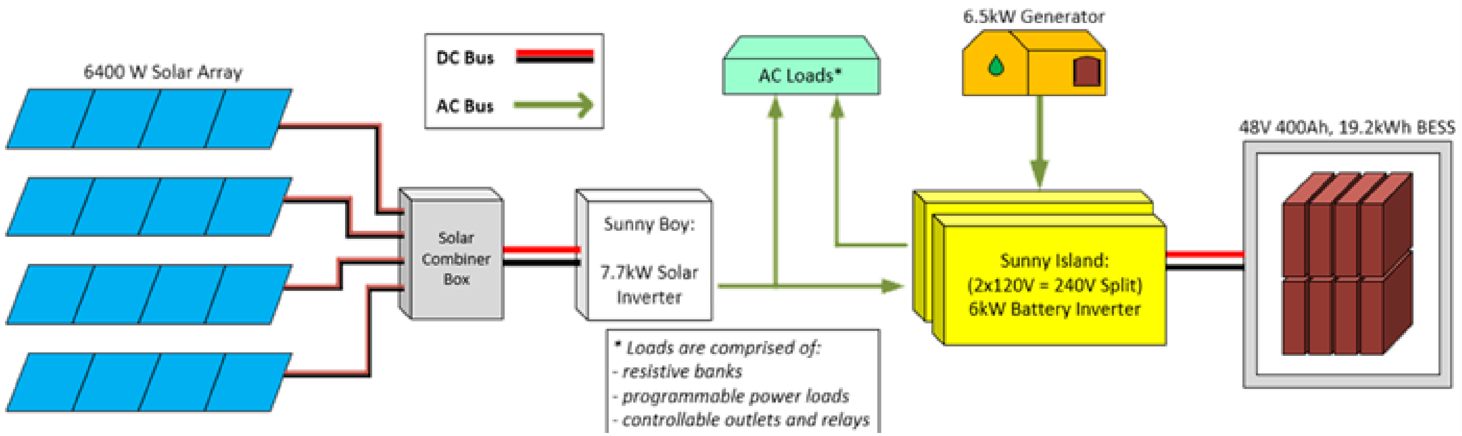
J. Morris, E. Louis, D. L. Van Bossuyt, G. Mocko, and J. Wagner, "Constraint Hypergraphs as a Unifying Framework for Digital Twins," Under review with IEEE Systems Journal, 2025

Demonstration: Microgrid Digital Twin



Physical Microgrid at Naval Postgraduate School



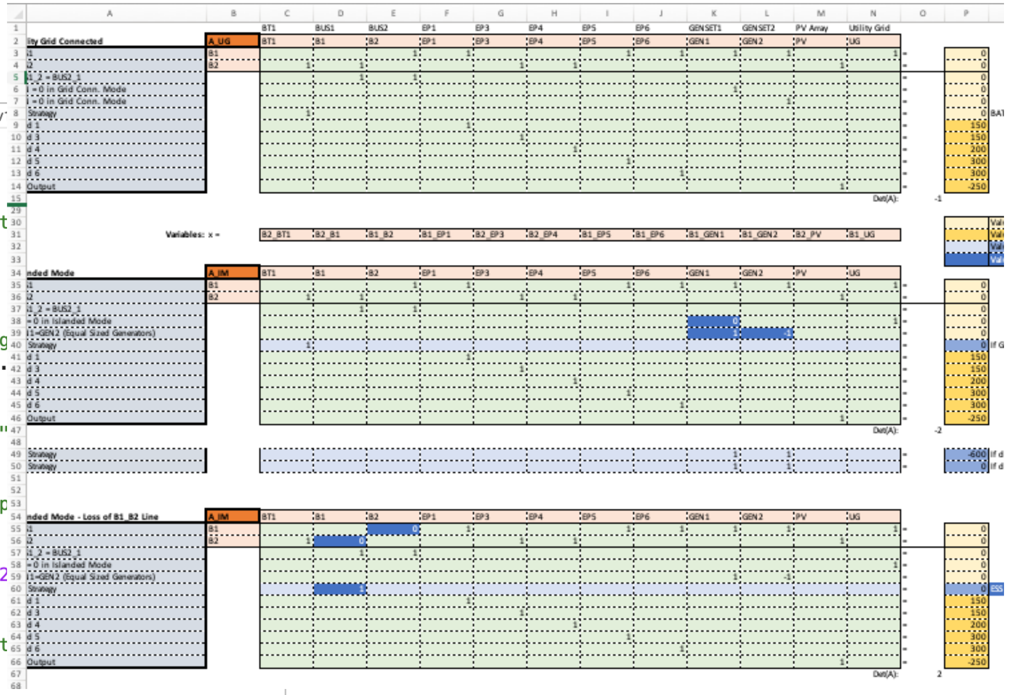


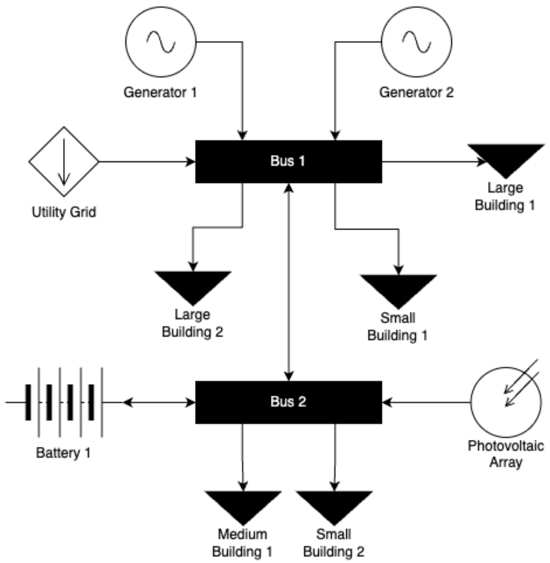
Previous Imperative Modeling of Microgrid

MG_LP_Test_v10.m

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

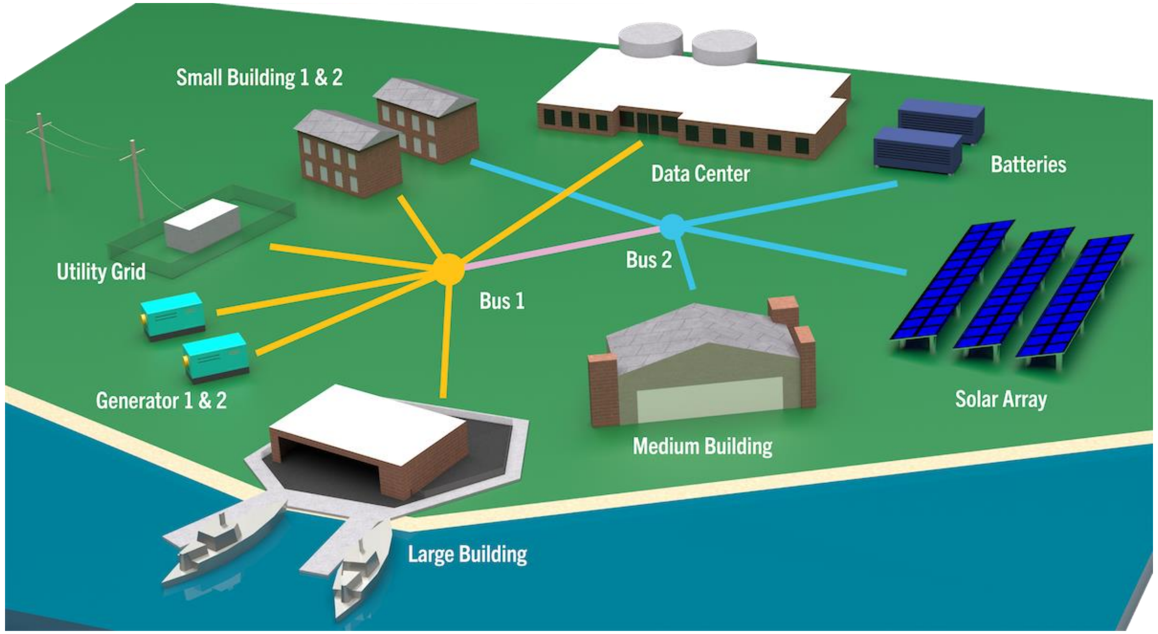
```
        %Initial Solve
        x(:,n) = linsolve(A2,b);

        %Determine if generator demand exceeds generat
        Gen_Demand = -x(pos.Gens,n);

        if any(Gen_Demand > [Gens.Capacity]')
            %If the B2 to B1 Buss line or BT1 line is
            %exhausted cannot utilize ESS to make up g
            if MG_State.B2_B1(n) == false || MG_State.
                    || BT1_Charge(n) < 0
                % Below line was useful for debugging
                % disp("Gens Overloaded at time step "
                Overload.Gens = true;
            else
                %Otherwise set Generators at full outp
                %make up unmet demand
                A2(6,:) = pos.Gens & MG_State{n,:};
                b(6) = -MG_State{n,{'B1_GEN1' 'B1_GEN2
                x(:,n) = linsolve(A2,b);
            end
        %If Generator demand is negative, then generat
        %charge batteries with excess PV generation
        elseif any(Gen_Demand < 0)
            A2(6,:) = pos.Gens;
            b(6) = 0;
            x(:,n) = linsolve(A2,b);
        end

        % Check for Battery Output Exceeded
        % Add 0.1 Due to Rounding Errors in Linear Solver
        if -x(contains(LoadVars,'BT1'),n) - 0.01 > BT1.Output * (BT1_Charge(n) > 0)
            Overload.BT1 = true;
            % Below line was useful for debugging in single runs. Commented out
```

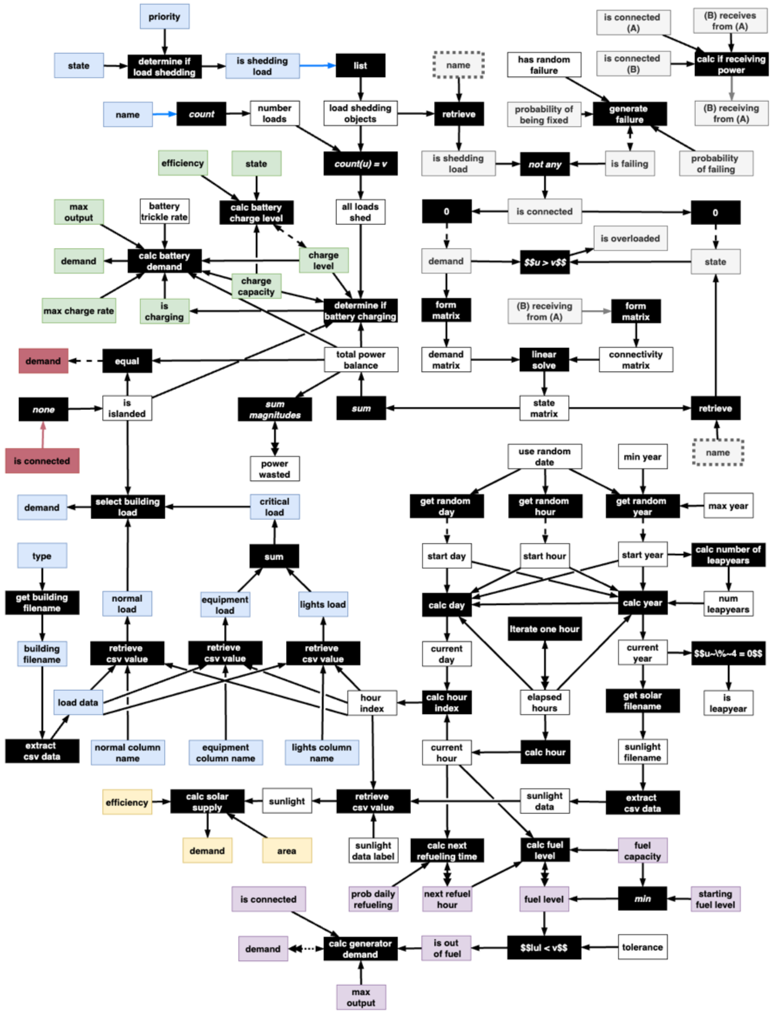




C. J. Peterson, “Systems Architecture Design and Validation Methods for Microgrid Systems,” Master’s thesis, Naval Postgraduate School, Monterey, CA, 2019

Transformed CHG contains 594 nodes and 334 edges





Includes all kinds of simulation and modeling

Algebraic rules

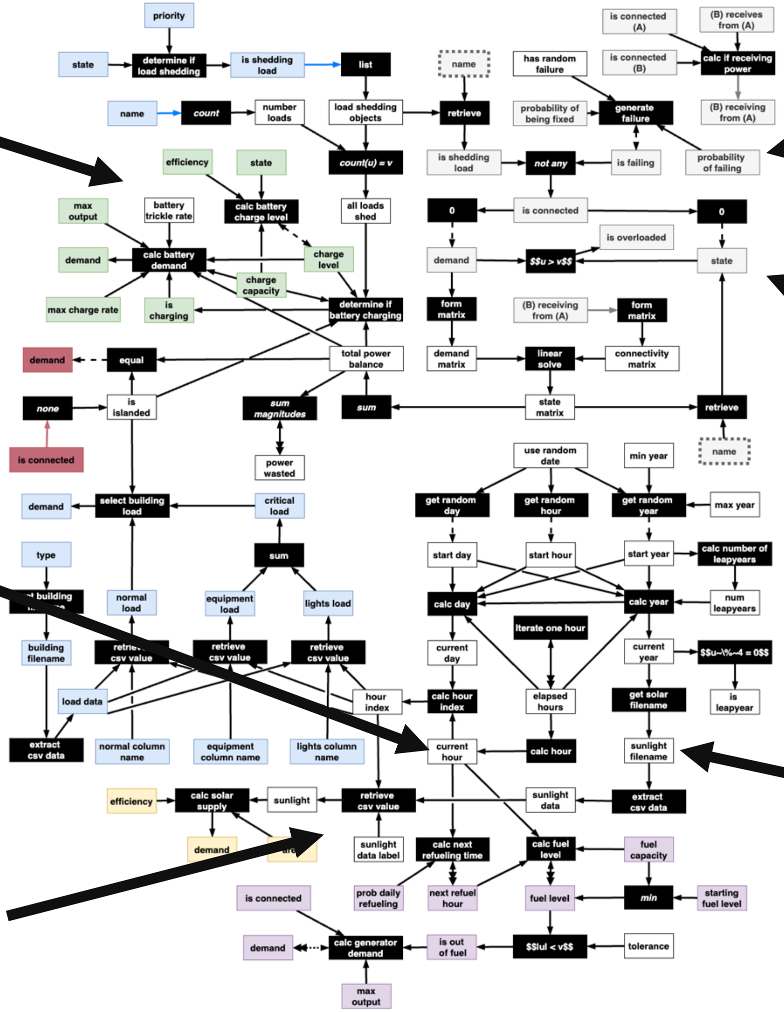
Stochastic models

Linear equation solvers

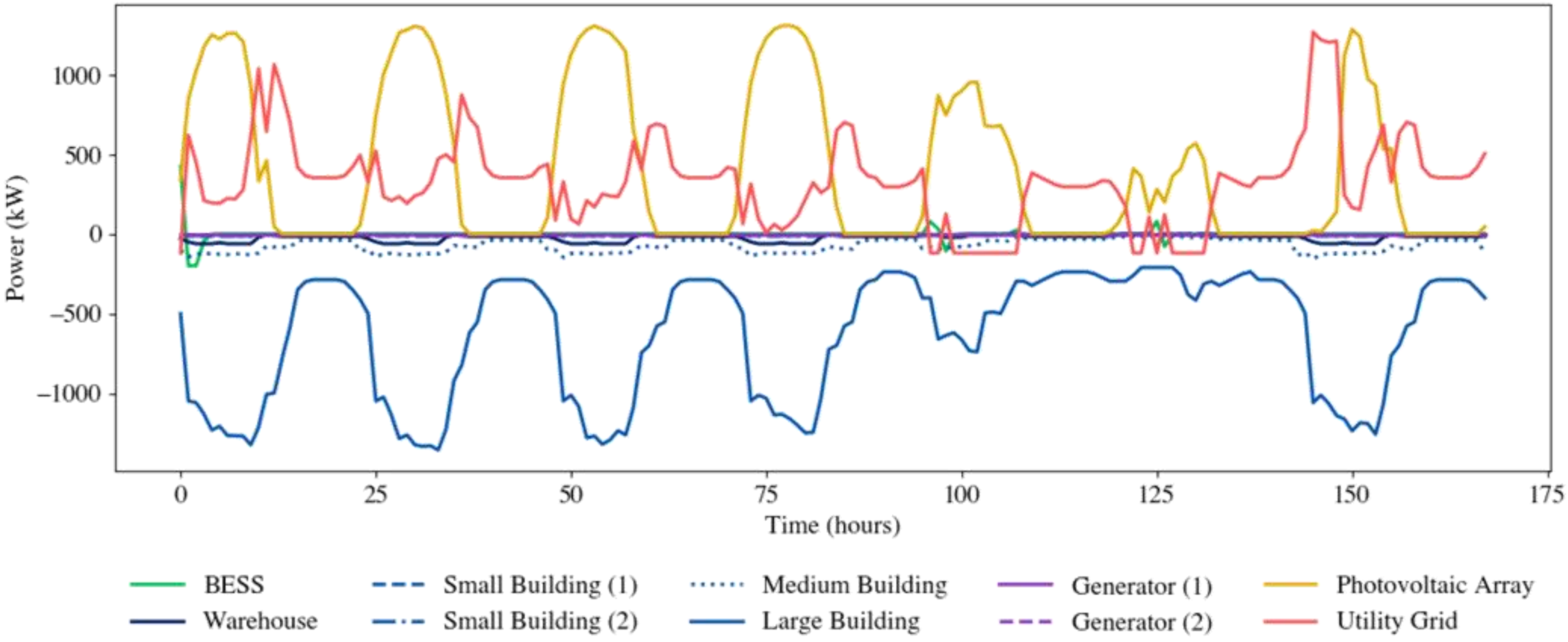
Dynamic simulation

File input/output

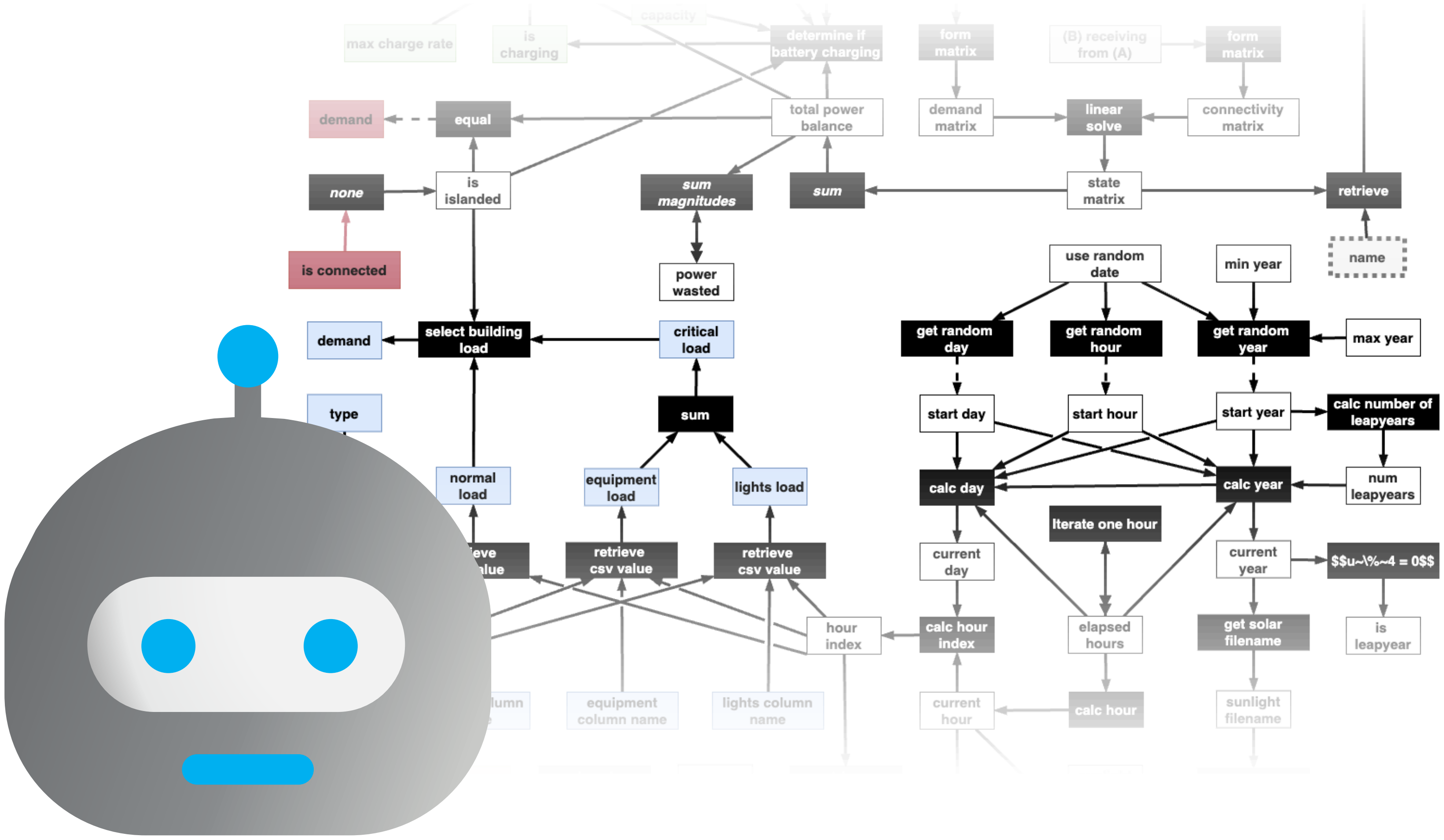
Database queries



Autonomously observes and simulates the real system

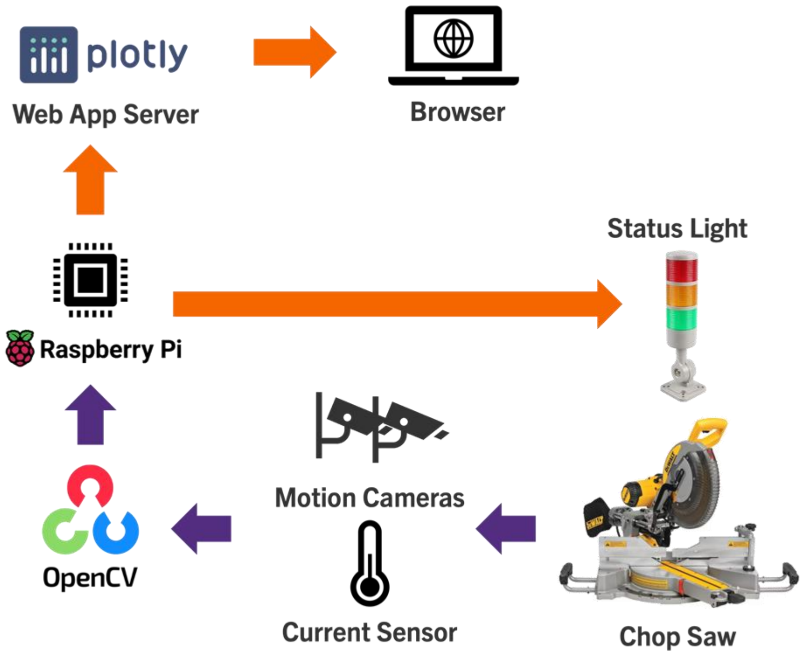


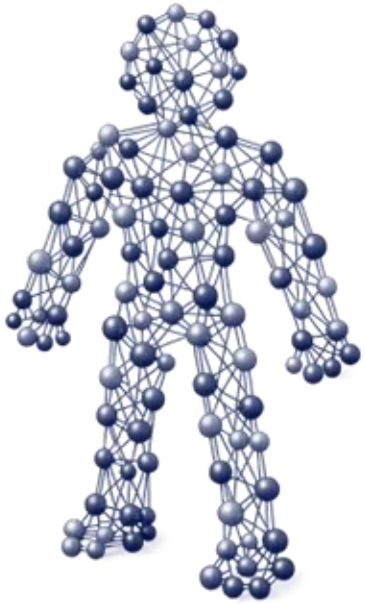
Reasoning is provided by automatic path composition, leading to deterministic AI



Needed for domains with complex information integration

Manufacturing PLM
CAD/CAE/CAM
ERP
IIoT





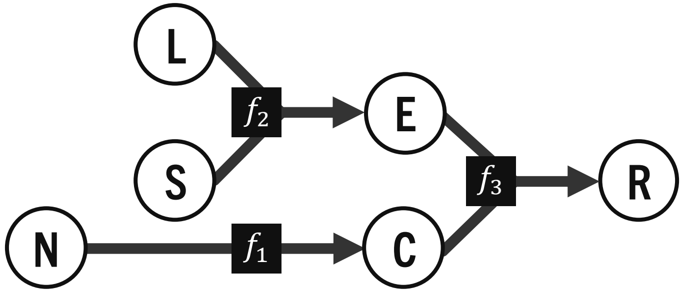
Personal Digital Twins
Medical records
Tracked biometric data
Environmental

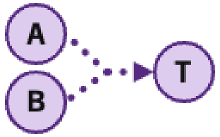
Therapeutic Diagnostics
Standardized treatment protocols
Doctor preferences
Available resources

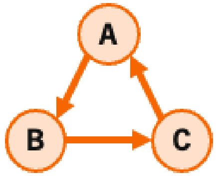
Oxford Medical Simulation

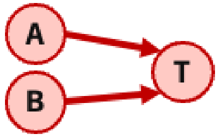
The structure of a CHG allows it to represent any system

Universal Modeling



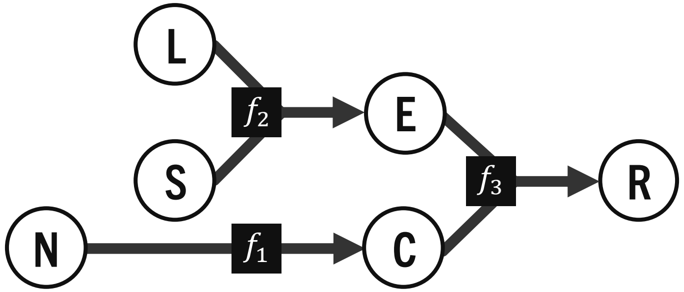


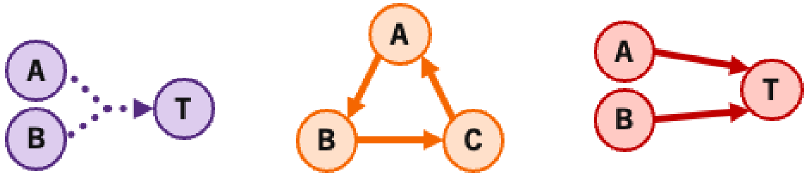




Information about a system can be discovered autonomously

Universal Modeling

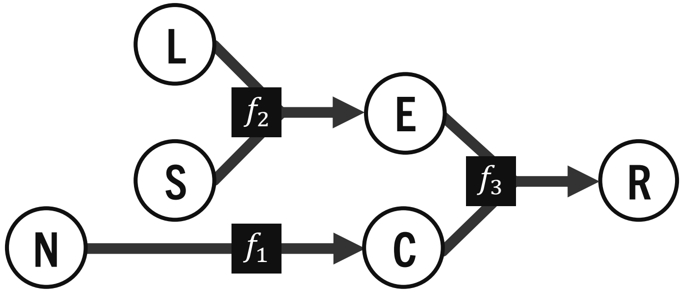


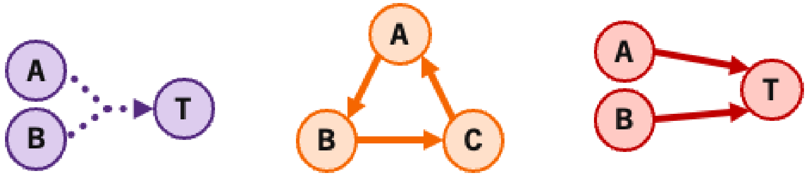


Declarative Simulation

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

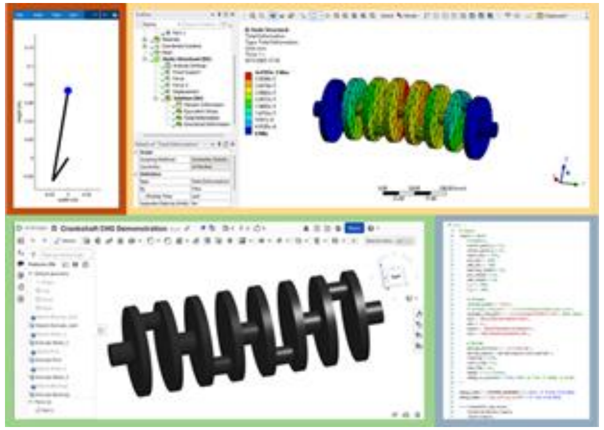
Universal Modeling

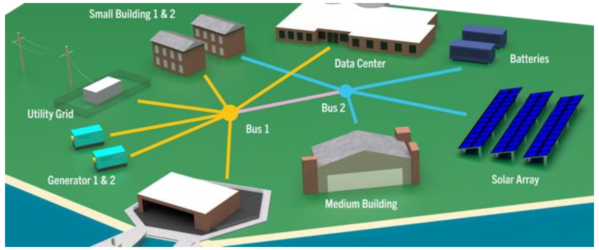




Declarative Simulation

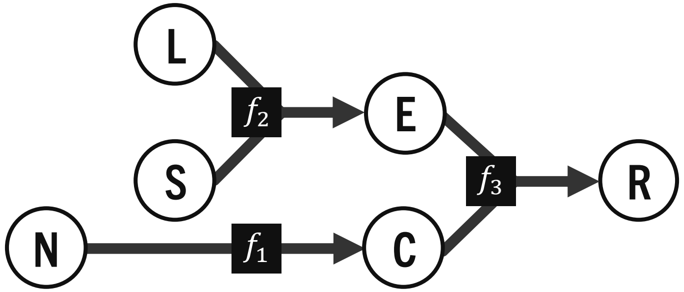
Executable Digital Threads

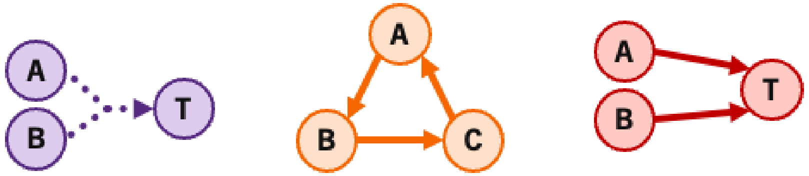




CHGs provide a universal language for declarative systems modeling

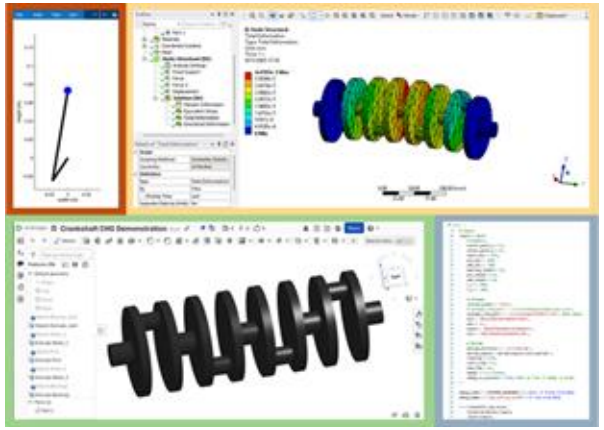
Universal Modeling

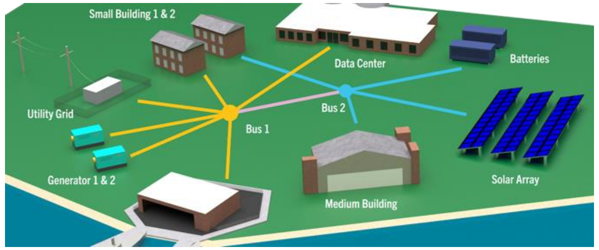




Declarative Simulation

Executable Digital Threads





John Morris
jhmrrs@clemson.edu