Declarative Integration of CAD Software into Multi-Physics Simulation via Constraint Hypergraphs

John Morris, **Gregory Mocko**, Satchit Ramnath, John Wagner

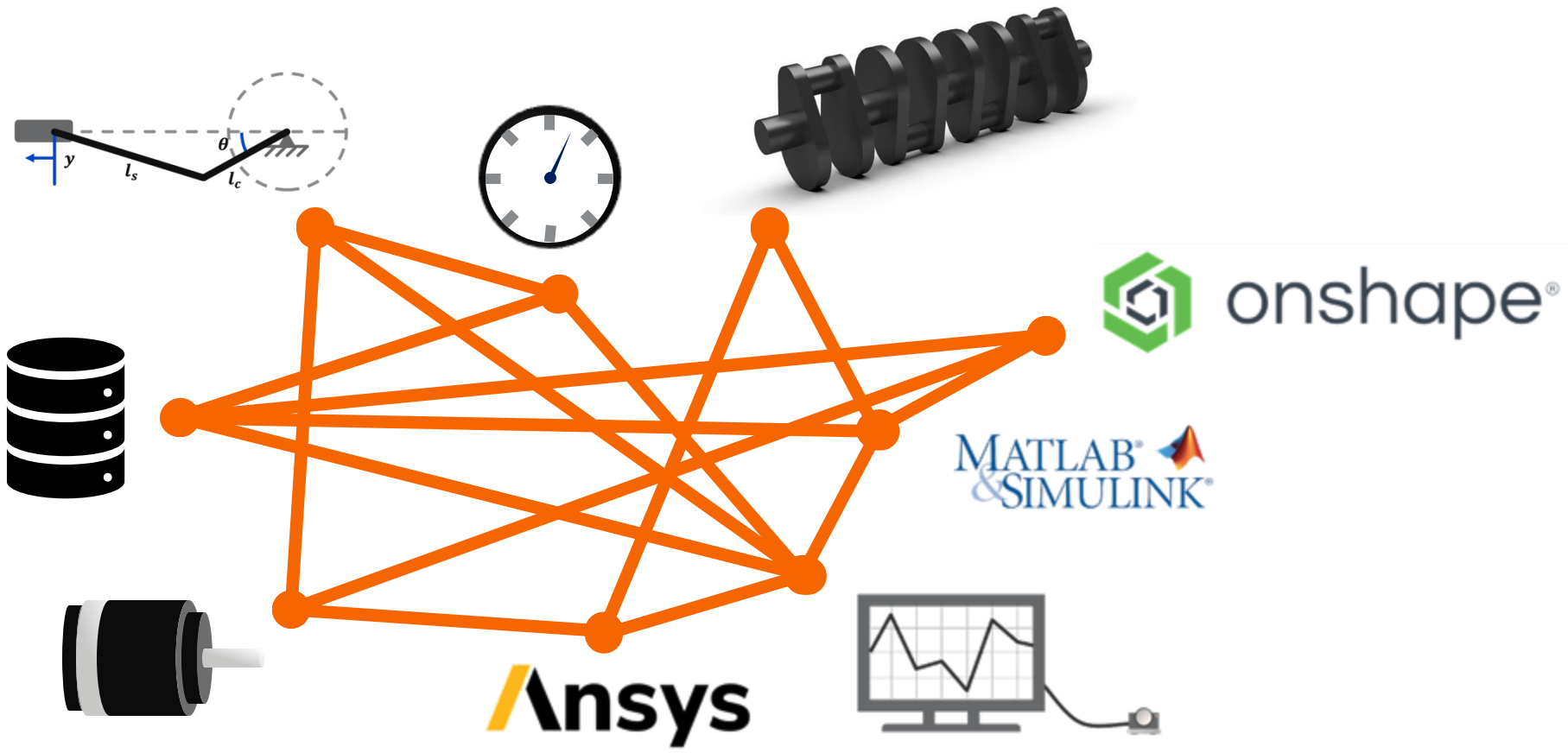
Dept. of Mechanical Engineering
Clemson University

PLMC

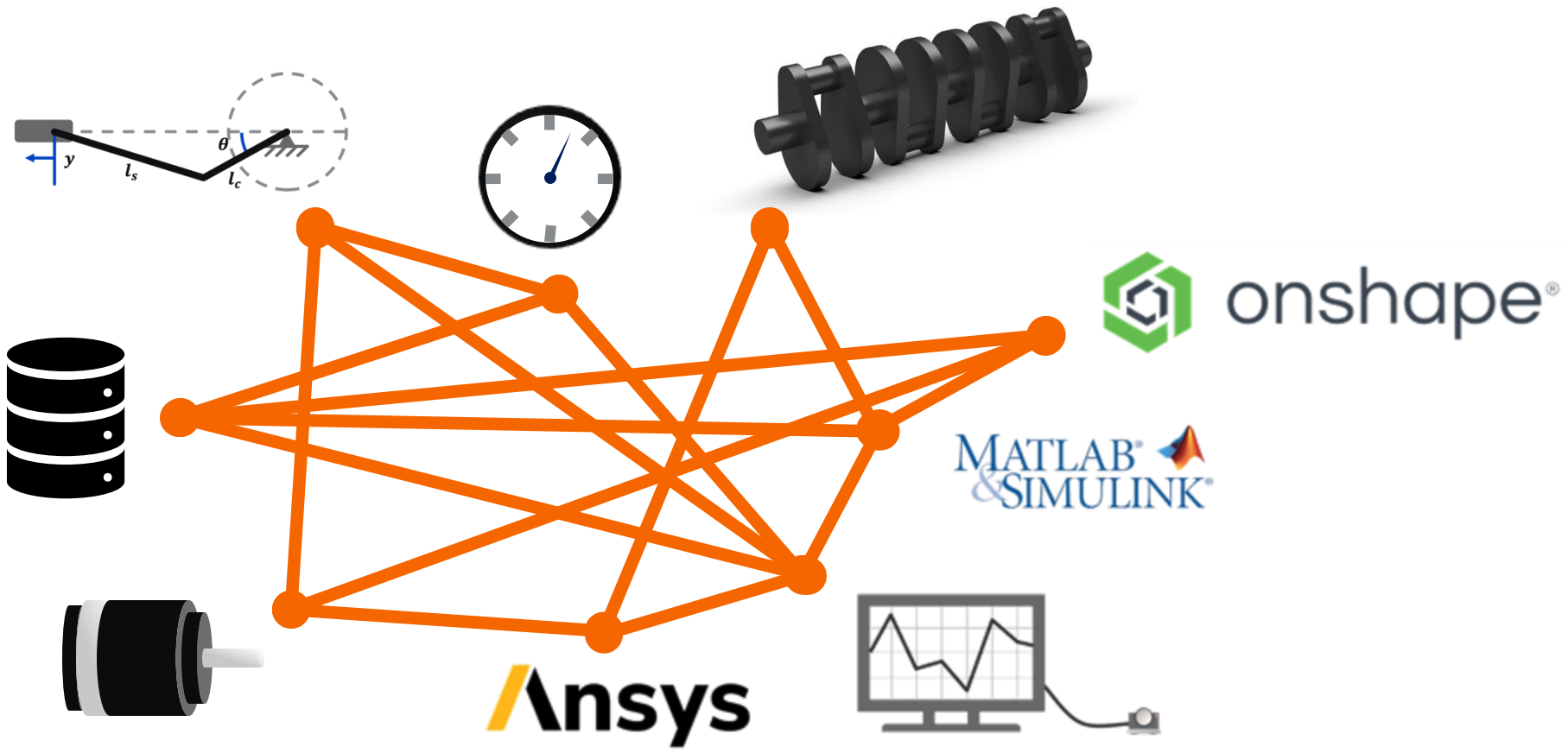
CEDAR
Clemson Engineering Design Applications and Research

CLEMSON

Complex systems are represented across different formalisms and software

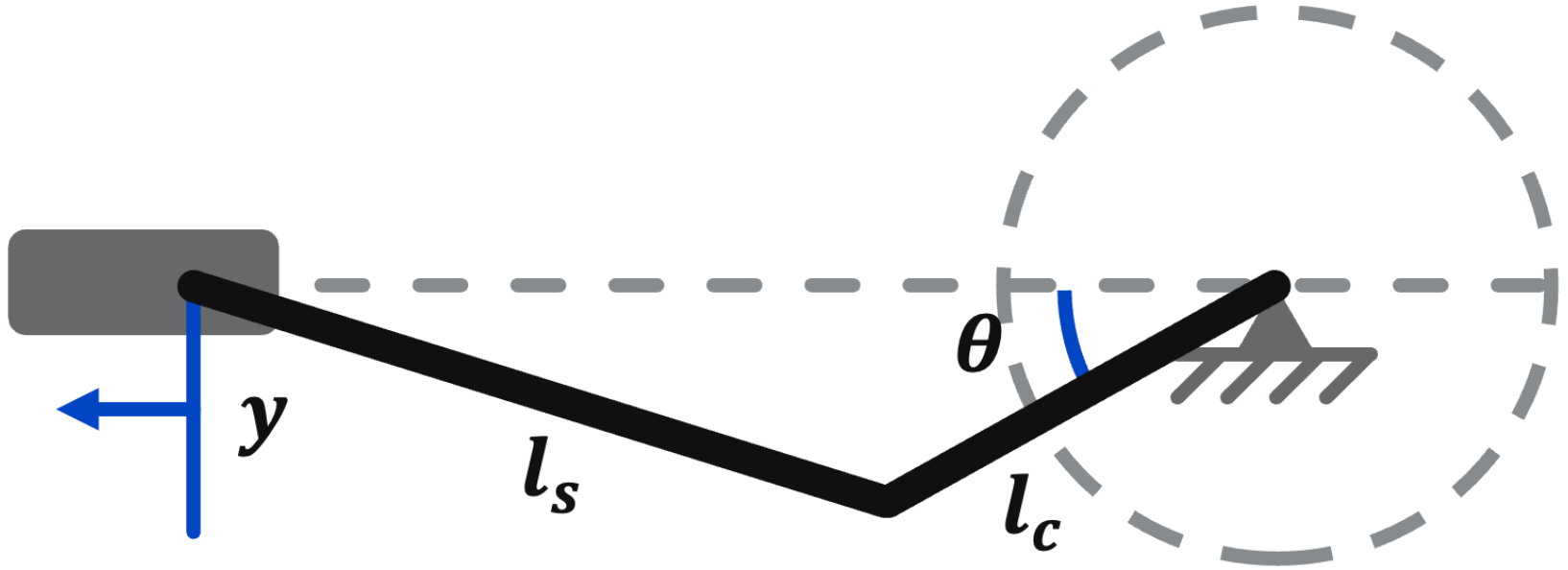


“A system is a collection of variables” —Ross Ashby



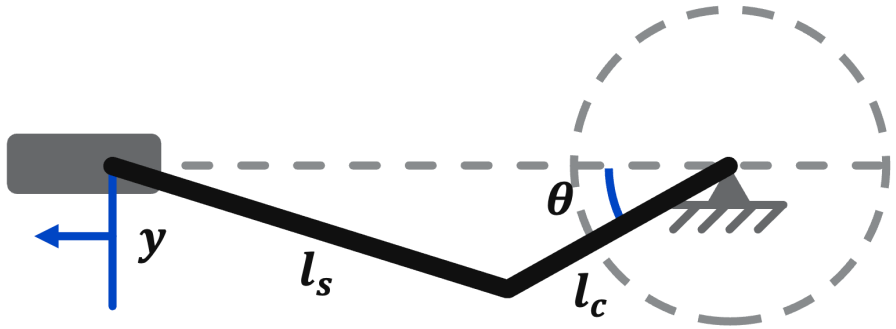
Ashby, W. R. (1956). An Introduction to Cybernetics. Chapman & Hall.

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



Ashby, W. R. (1956). An Introduction to Cybernetics. Chapman & Hall.

Behavioral models describe relationships between variables as functions



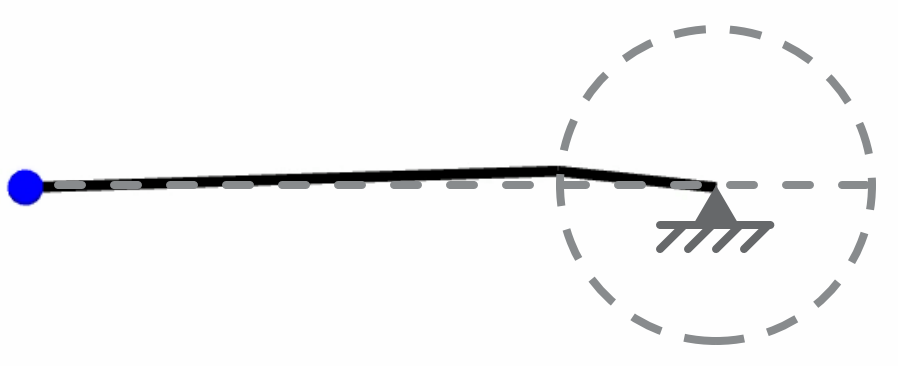
$$\theta = \cos^{-1}\left(\frac{y^2 + l_c^2 - l_s^2}{2yl_c}\right)$$

$$y = l_c \cos\theta + \sqrt{l_s^2 - l_c^2 \sin^2\theta}$$

$$\omega = \frac{d\theta}{dt}$$

$$\dot{y} = \frac{dy}{dt}$$

Simulation is the use of these functions to transform a set of inputs to a set of outputs



$$\theta = \cos^{-1}\left(\frac{y^2 + l_c^2 - l_s^2}{2yl_c}\right)$$

$$y = l_c \cos\theta + \sqrt{l_s^2 - l_c^2 \sin^2\theta}$$

$$\omega = \frac{d\theta}{dt}$$

$$\dot{y} = \frac{dy}{dt}$$

Unique sequences exist for every input/output pairing

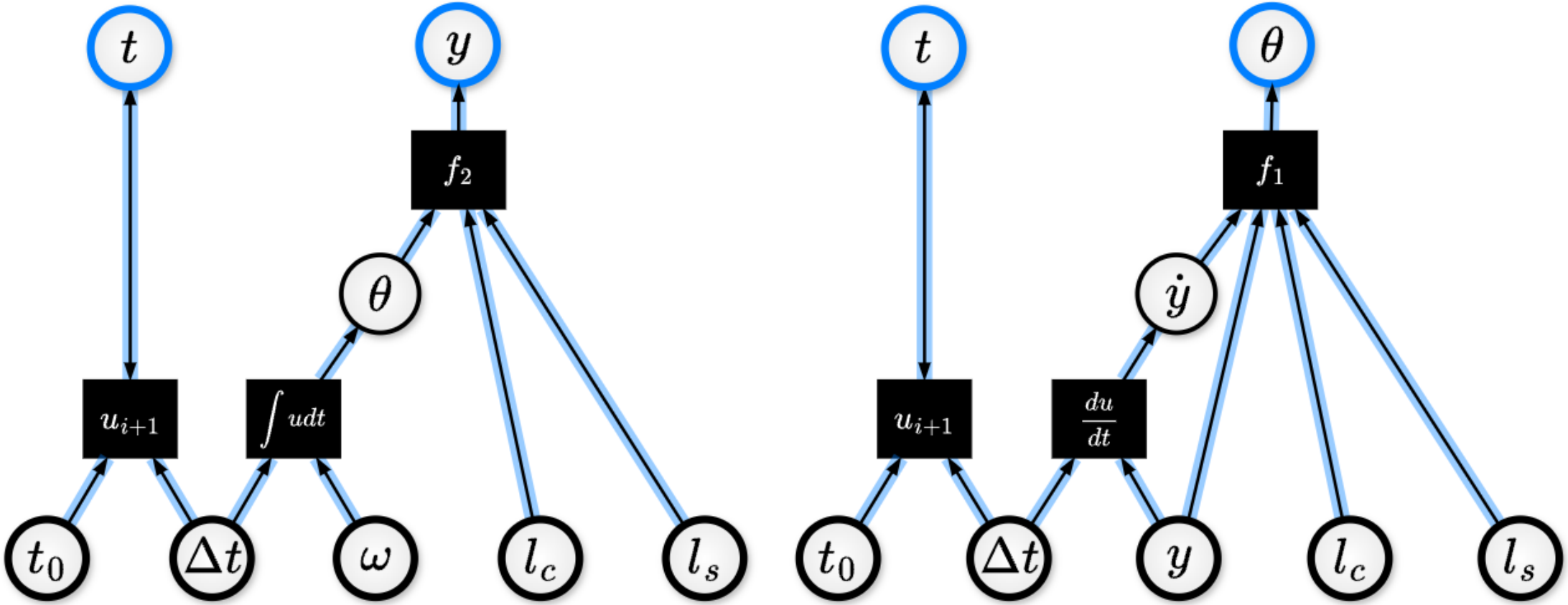
$\omega \rightarrow y$

```
% Inputs
l_c = 30;
l_s = 100;
timestep = 0.01;
time = 0:timestep:4;
omega = 2*pi;

% Simulation process
theta = zeros(size(time));
for i = 2:length(time)
    theta(i) = theta(i-1) + omega * timestep;
end

y = arrayfun(@(th) piston_height(th,l_c,l_s), theta);

function y = piston_height(th, l_c, l_s)
    y = l_c * cos(th) + sqrt(l_s^2 - l_c^2 * sin(th)^2);
end
```

$y \rightarrow \theta$

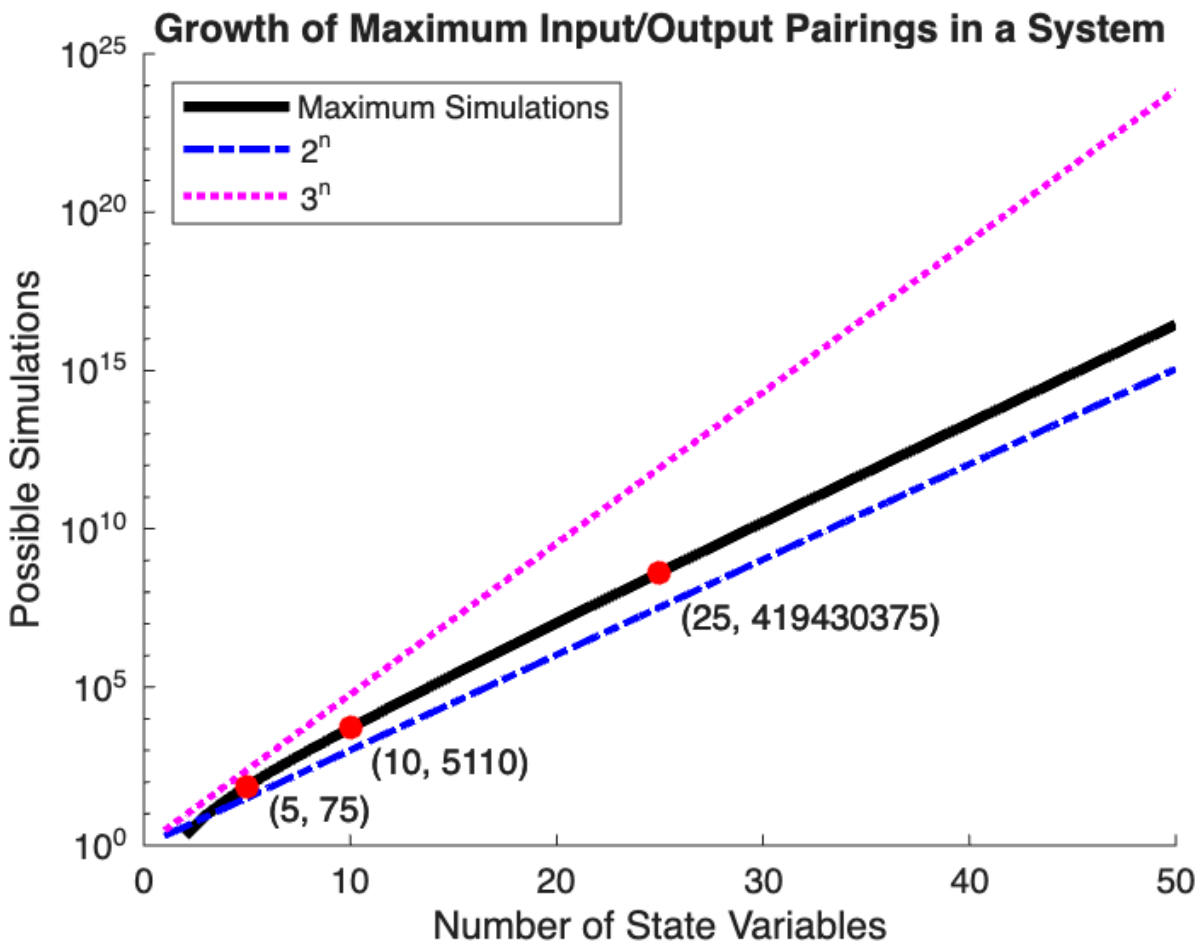
```
% Inputs
l_c = 30;
l_s = 100;
timestep = 0.01;
time = 0:timestep:4;
y = l_c*cos(time*10) + l_s;

% Simulation process
ydot = zeros(size(time));
for i = 2:length(time)
    ydot(i) = (y(i) - y(i-1)) / timestep;
end

th = arrayfun(@(y, ydot) crank_pos(y,ydot,l_c,l_s), y, ydot);

function th = crank_pos(y, ydot, l_c, l_s)
    th = acos((y^2 + l_c^2 - l_s^2) / (2*y*l_c));
    if ydot > 0 %Correct arccos domain
        th = -th;
    end
end
```

Simulation sequences can be represented as trees with inputs as leaves and outputs are roots

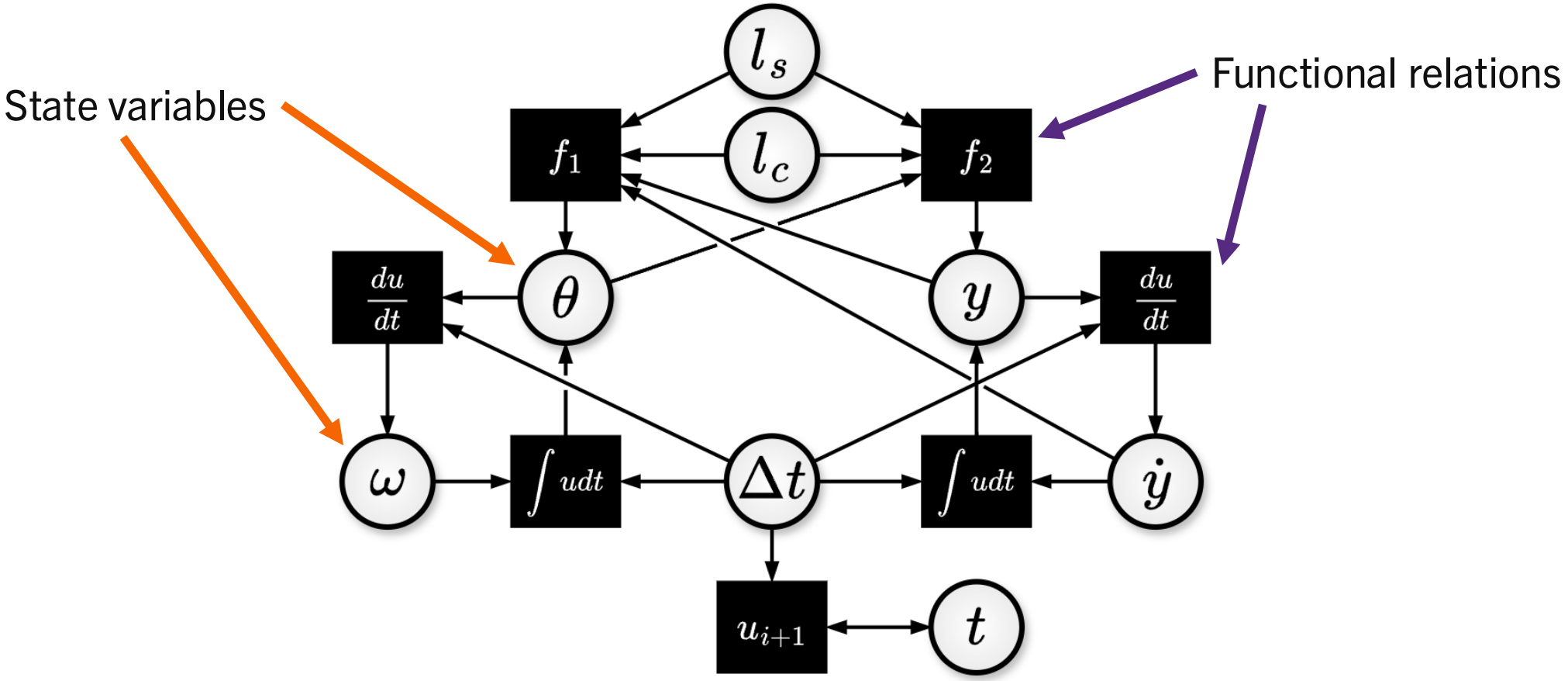


For a system of *n* variables, full simulation requires impractical number of sequences

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

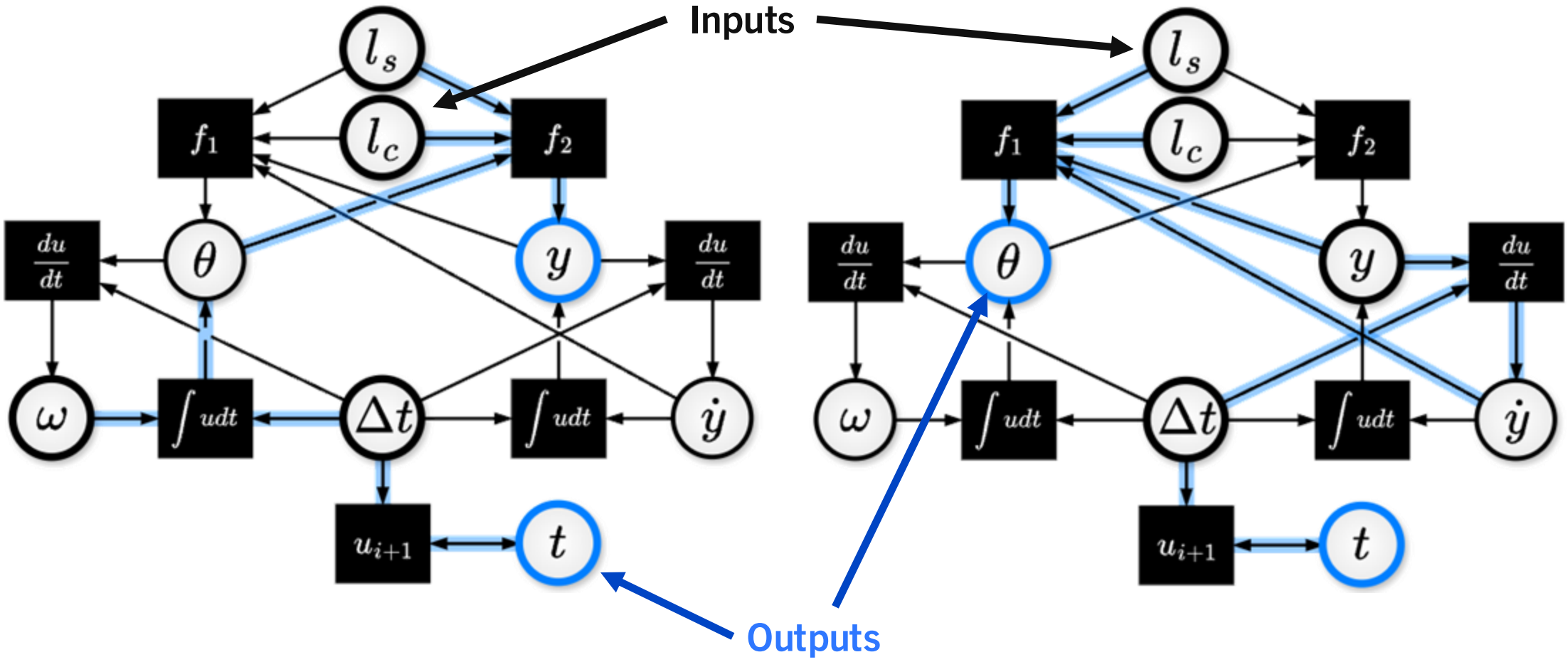


A constraint hypergraph gives declarative depiction of all simulation sequences

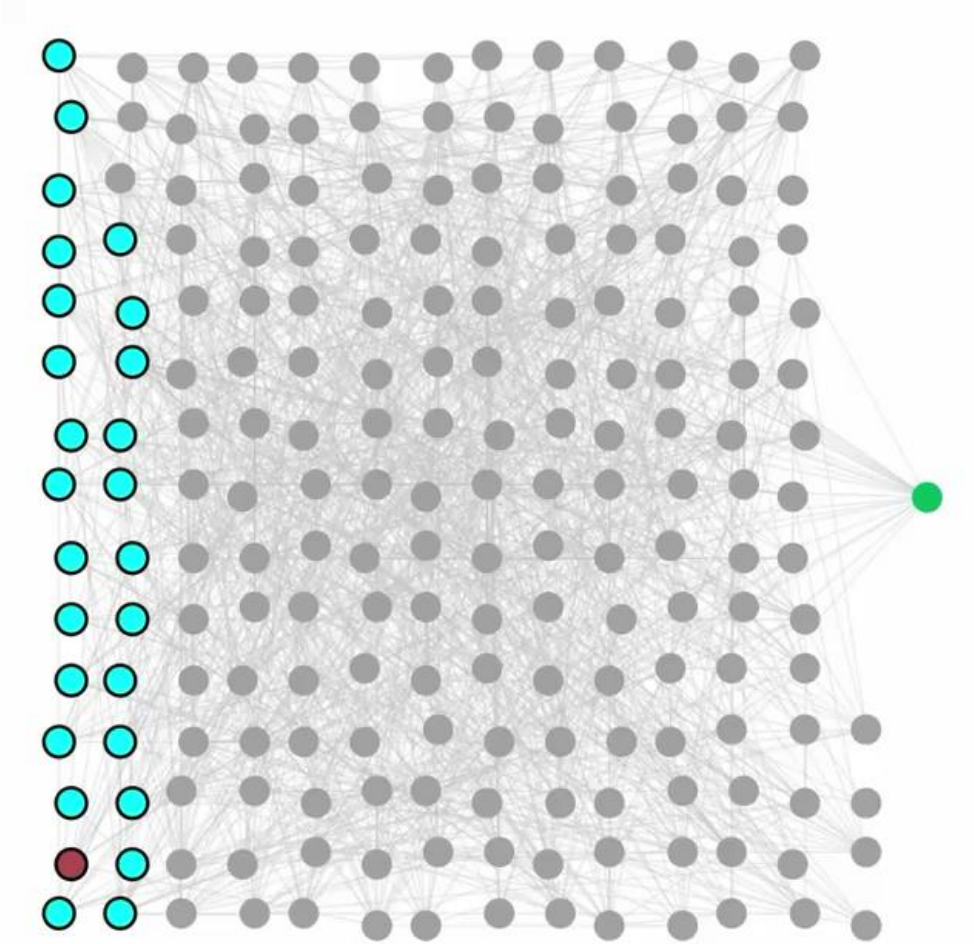


Morris, J., Mocko, G., & Wagner, J. (2025). Unified System Modeling and Simulation via Constraint Hypergraphs. *Journal of Computing and Information Science in Engineering.*

Unique simulation sequences are described by paths in the hypergraph



Simulation constructed using a Breadth-First Search of the graph



ConstraintHg

constrainthg 0.2.3

`pip install constrainthg`

Latest version

Released: Jun 25, 2025

Kernel for building and simulating constraint hypergraphs.

Navigation

- Project description
- Release history
- Download files

Verified details

These details have been verified by PyPI

Project description

ConstraintHg

DOI 10.5281/zenodo.15741546 | docs passing | tests 26/26 | release v0.2.3 | last commit july

ConstraintHg is a systems modeling kernel written in Python that enables general definition and universal simulation of any system. The kernel breaks a system down into the informational values (nodes) and functional relationships (hyperedges), providing robust simulation through pathfinding operations. This repository is under active

Process of forming an integrated constraint hypergraph model

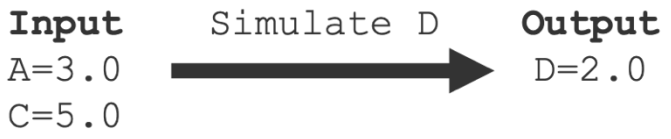
1. Identify system facts (nodes)
Parameters, application variables, API tokens, etc.

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

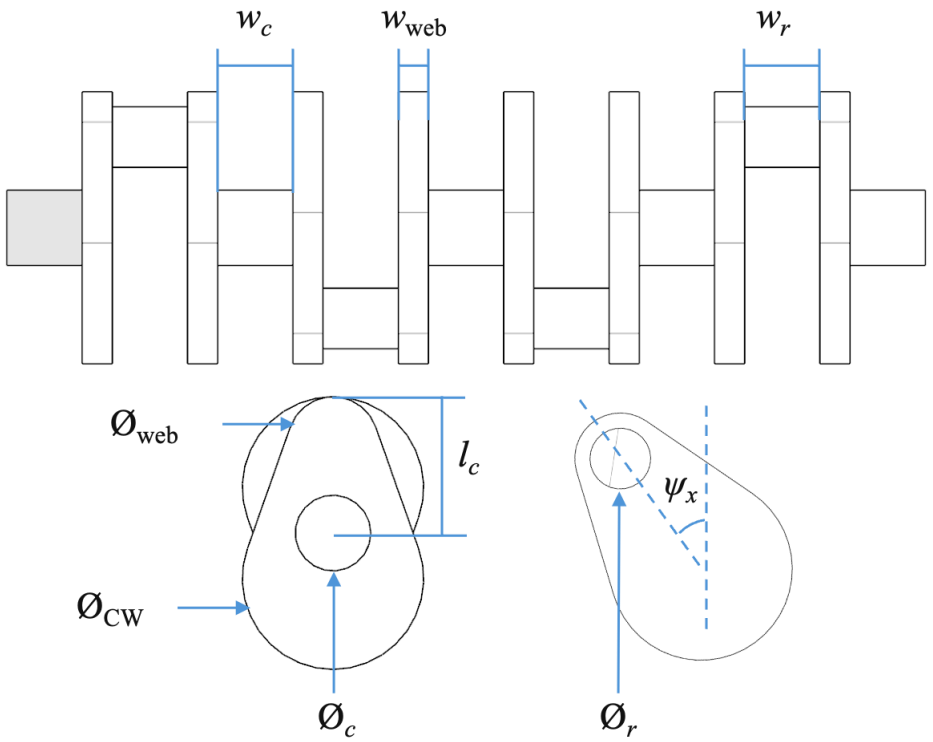
ConstraintHg

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

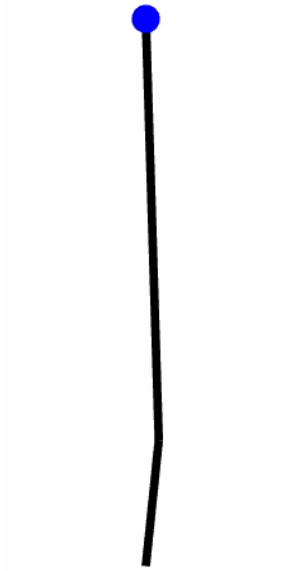


Constraint hypergraphs can describe simulation paths that go across calculating software

Geometric Specifications

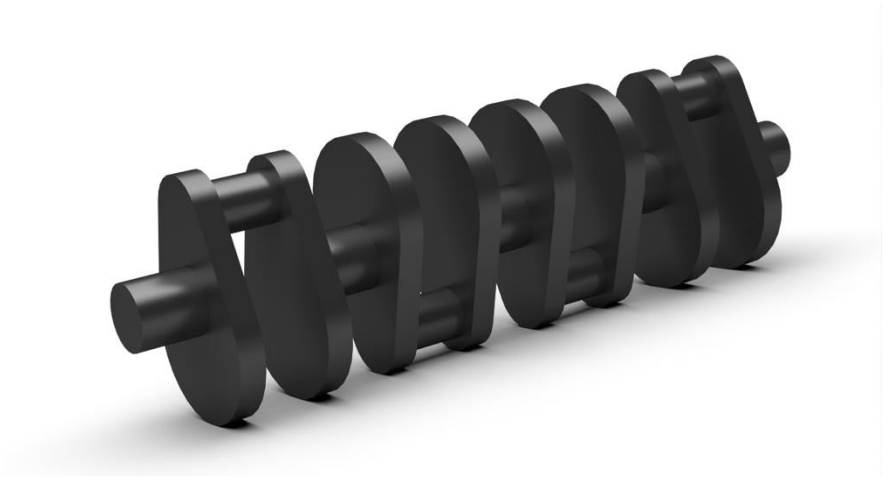


Kinematic Analysis



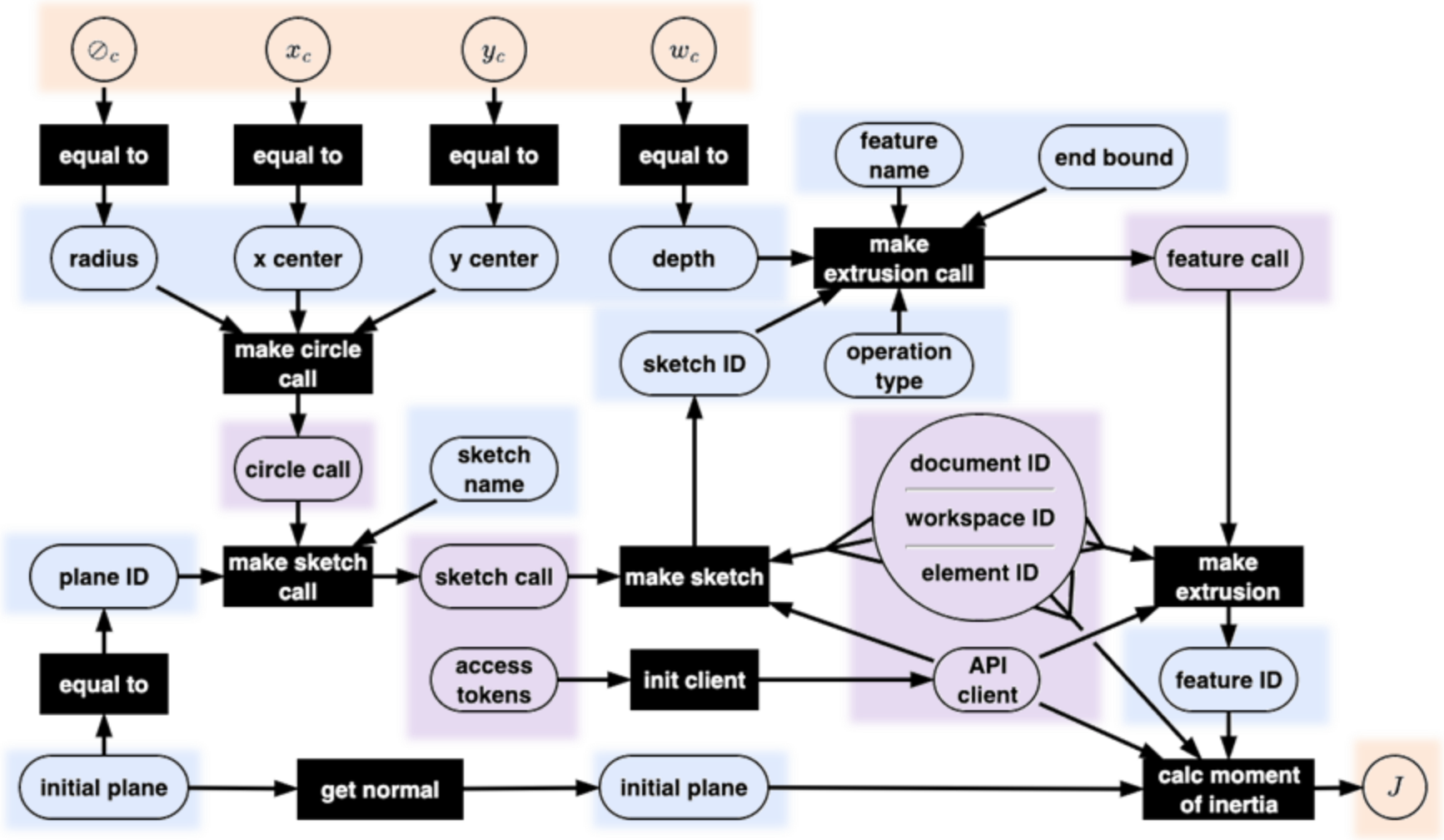
MATLAB®

Mass Properties



onshape®

The hypergraph declaratively integrates Onshape's functionalities with the greater system model



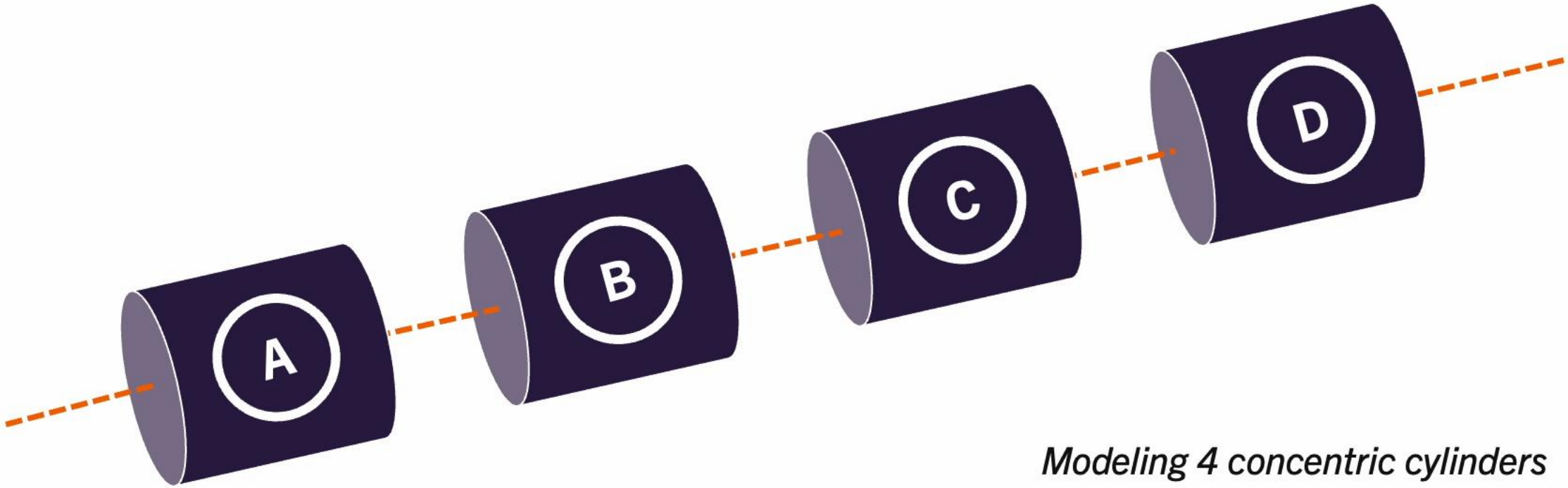
Legend

CAD functionalities

Physical parameters

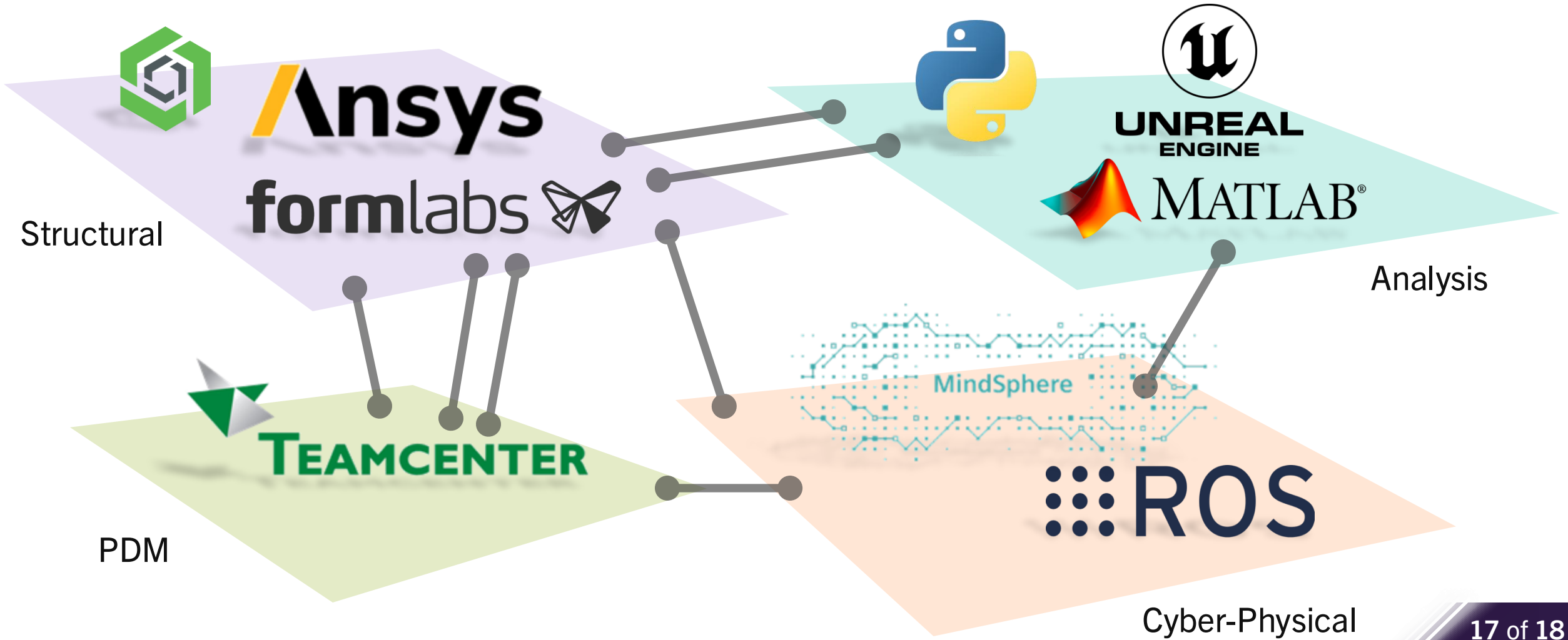
API specific

Declarative modeling allows autonomous reordering of information



Modeling 4 concentric cylinders

Future work: describe a foundation for model-based engineering using constraint hypergraphs



More information available at our documentation

Please reach out to jhmrrs@clemson.edu

PLMC

CEDAR
Clemson Engineering Design Applications and Research

CLEMSON