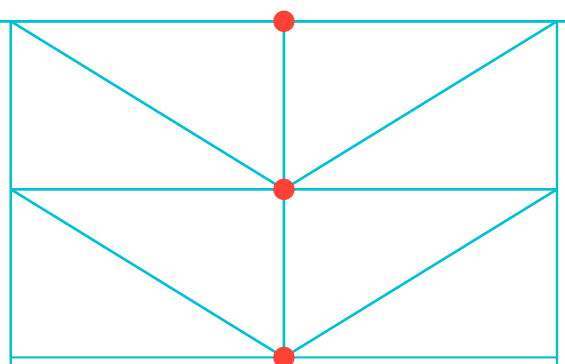
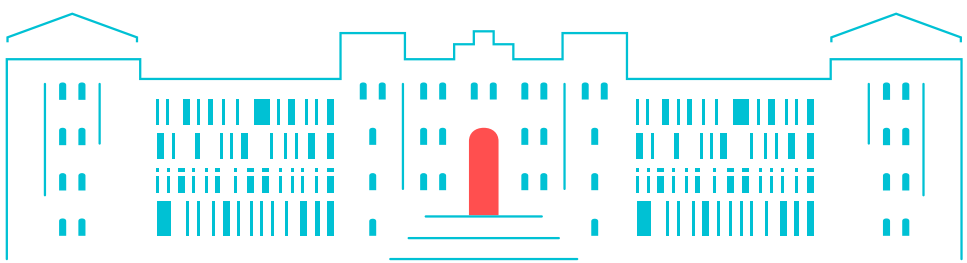
Tutorial: Data-Driven Modeling of Hybrid Automata within a Unified Model Learning Framework

TUHH
Hamburg
University of
Technology



2026-05-11



Maximilian Schmidt, Swantje Plambeck

CPS-IoT Week 2026

Schedule

TUHH

Lecture

Hands-on

Part I — Unified CPS Modeling with Flowcean

- Data acquisition
- Preprocessing
- Learning strategies
- Model
- Evaluation strategies & metrics

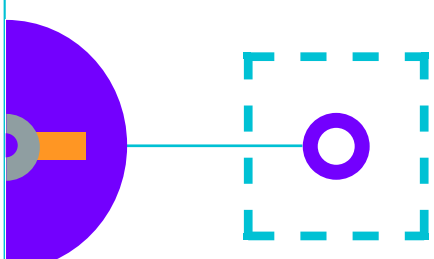
- Load and data
- Train models
- Evaluate accuracy and runtime
- Store and reload models

Part II — Data-Driven Identification of Hybrid Automata

- Motivation for hybrid models
- Physical flow functions
- Segmentation of time-series data
- Guard conditions and transition logic
- Iterative learning process

- Load datasets or generate data
- Create hybrid system simulators
- Learn hybrid automaton models
- Visualize model
- Compare performance

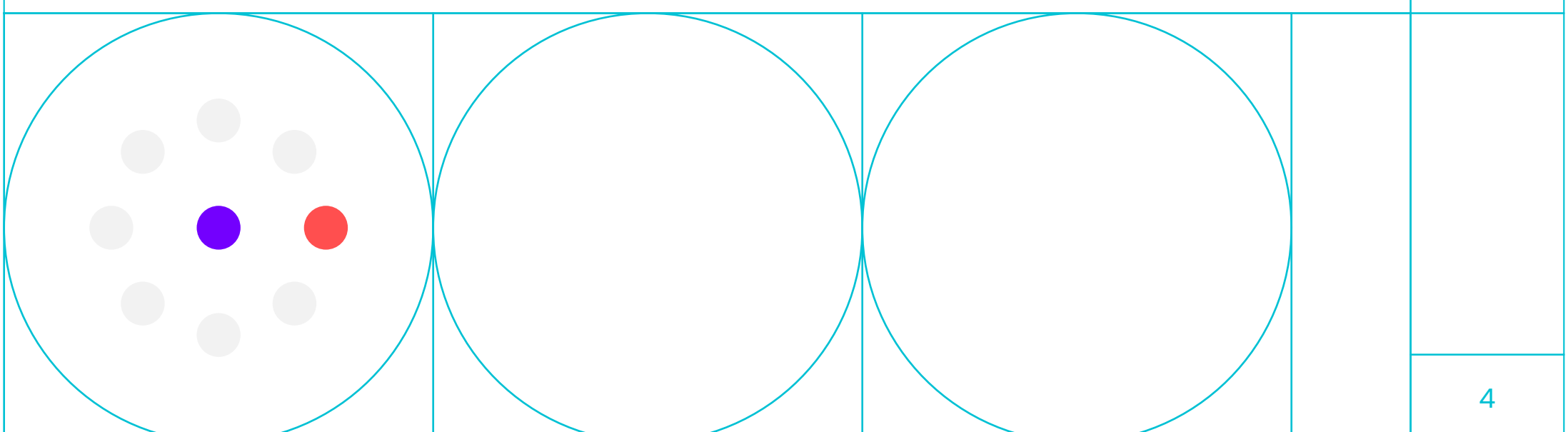
Agenda



1. Cyber-Physical Systems
2. Learning Pipeline
3. Unified Learning Framework
4. Hands-On
5. Motivation for Hybrid Automata
6. Simulation Models
7. Hands-On
8. Learning Hybrid Automata

TUHH

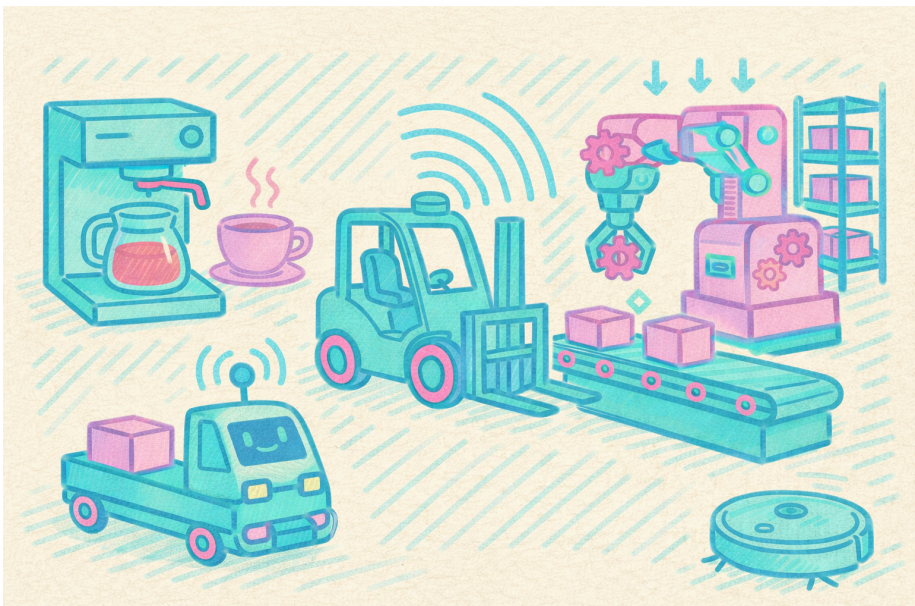
Cyber-Physical Systems



Cyber-Physical Systems

TUHH

- CPS systems are the backbone of modern industrial automation



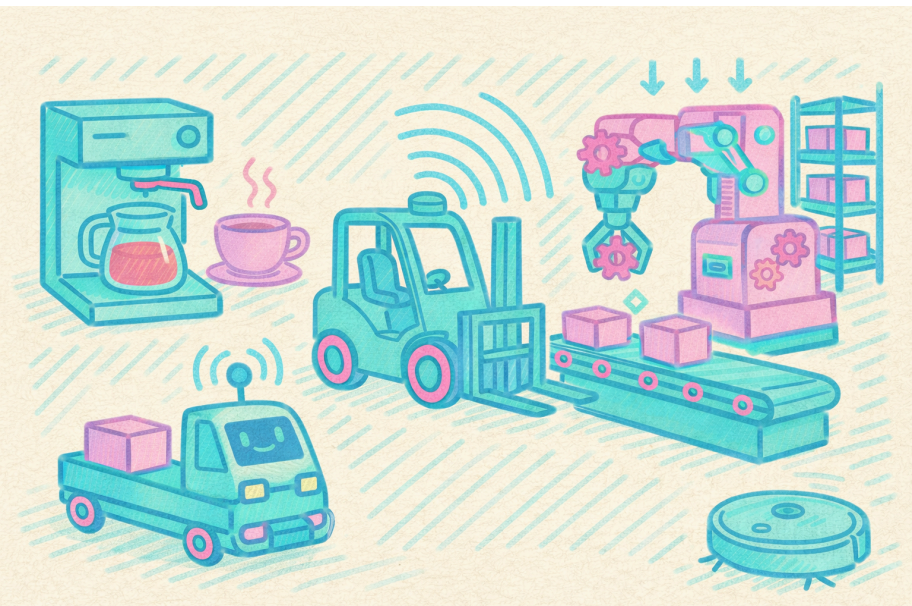
Cyber-Physical Systems

TUHH

- CPS systems are the backbone of modern industrial automation
- operational reality is challenging
  - complex dynamics
  - heterogeneous data

Cyber-Physical Systems

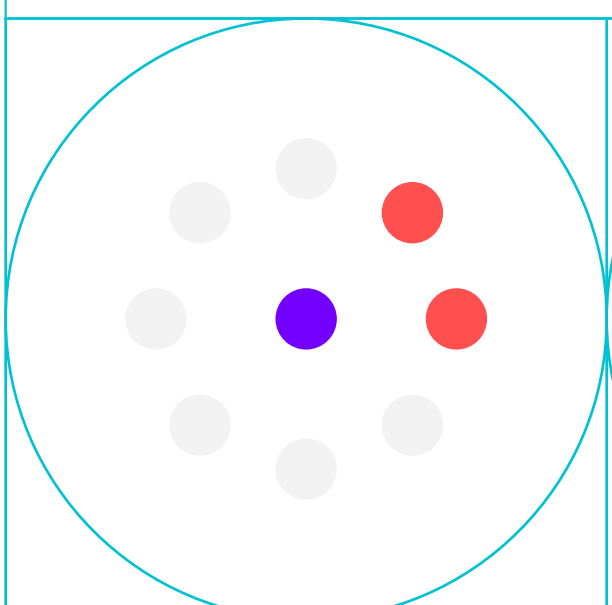
- CPS systems are the backbone of modern industrial automation
- operational reality is challenging
  - complex dynamics
  - heterogeneous data
- modeling and learning are crucial for understanding, optimizing, and controlling CPS



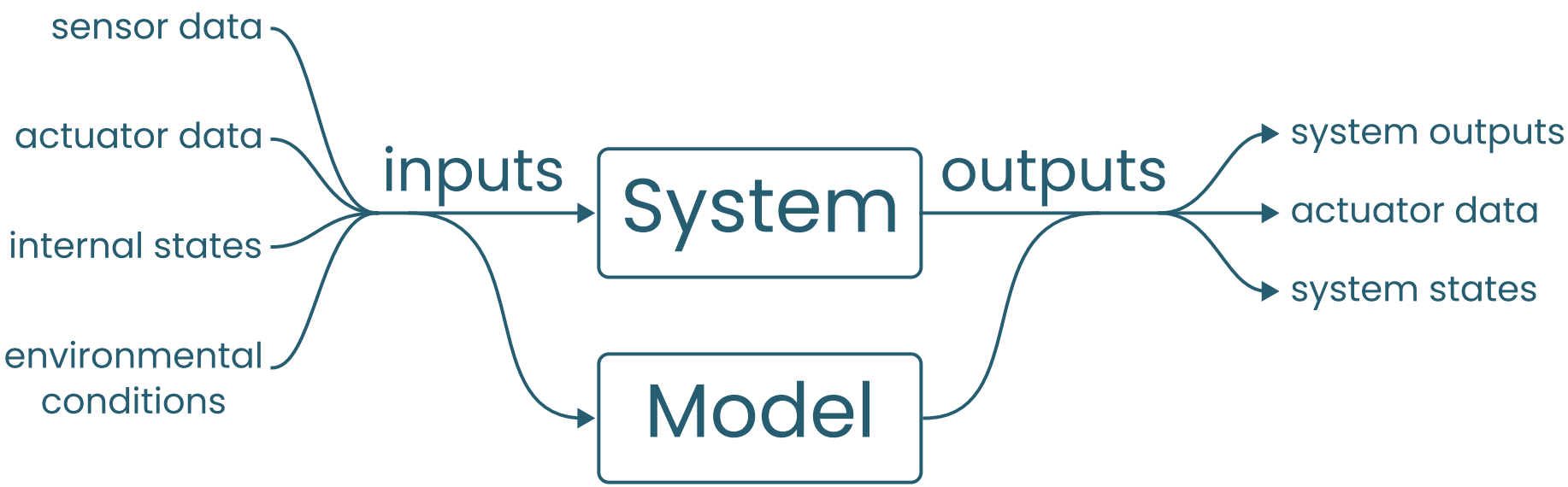
Cyber-Physical Systems

- CPS systems are the backbone of modern industrial automation
- operational reality is challenging
  - complex dynamics
  - heterogeneous data
- modeling and learning are crucial for understanding, optimizing, and controlling CPS
- data-driven modeling is increasingly important because of the complexity and variability of CPS

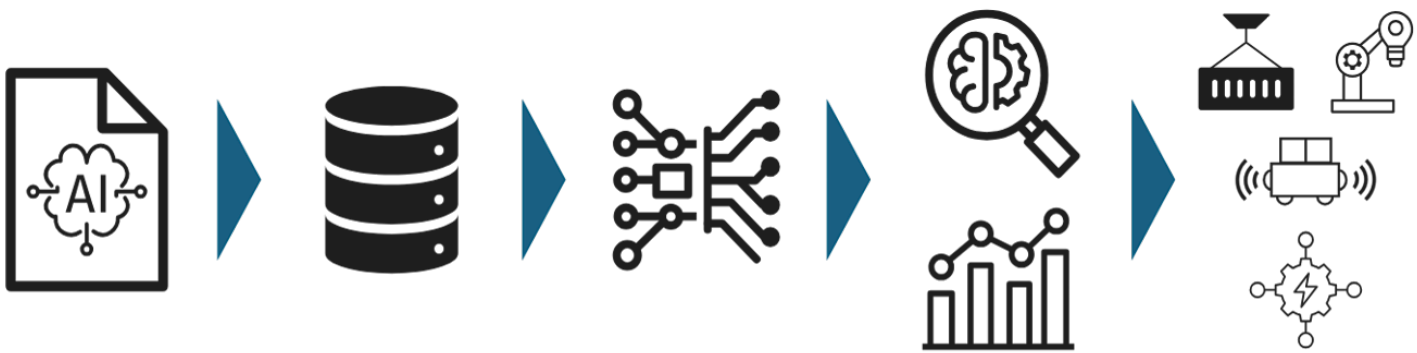
Learning Pipeline



System & Model

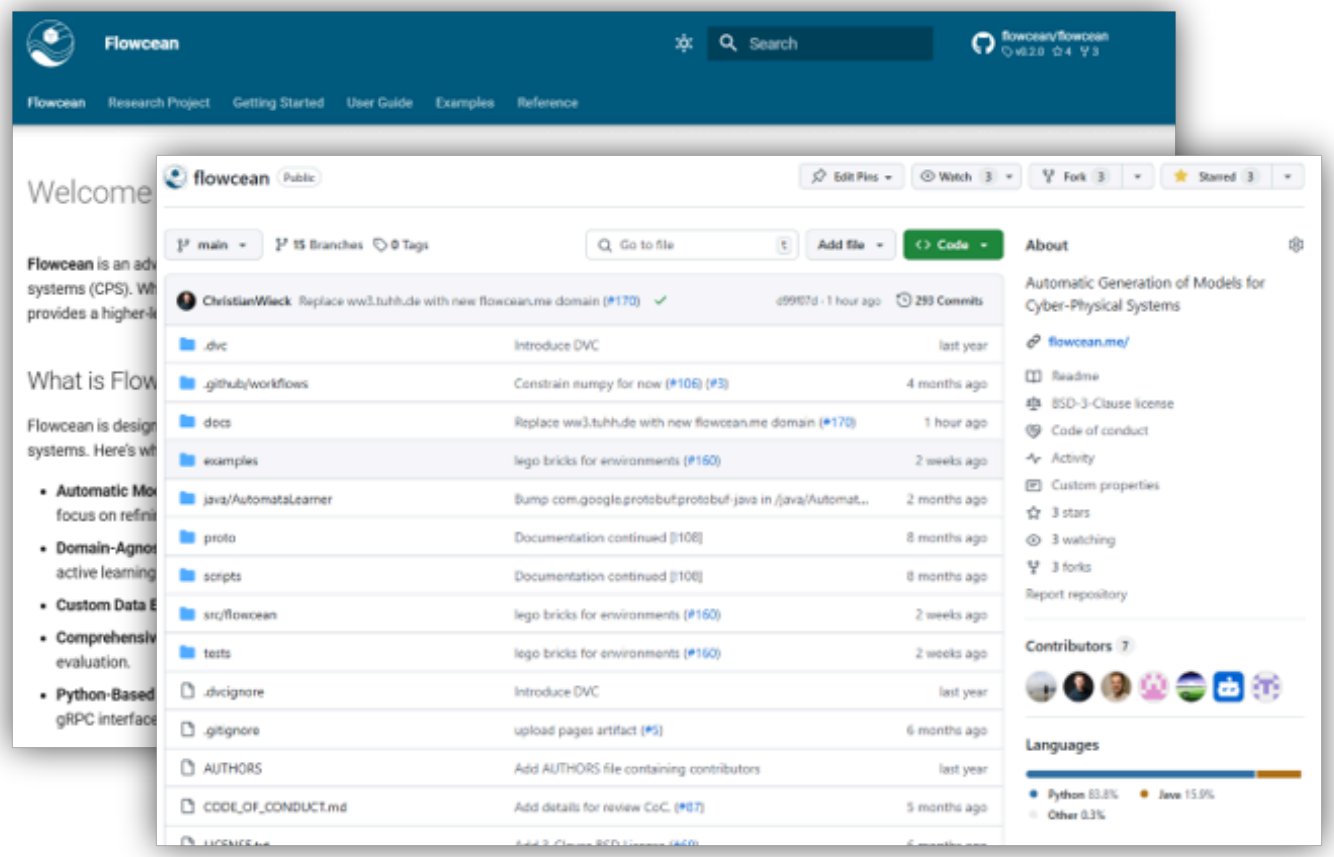


Modeling Pipeline

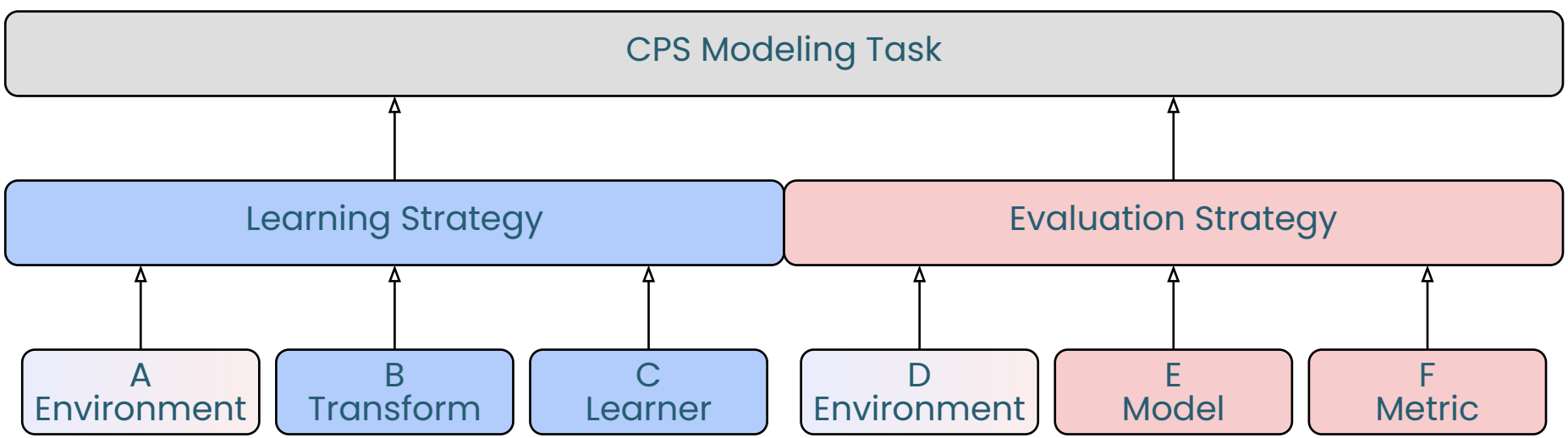


Flowcean Software

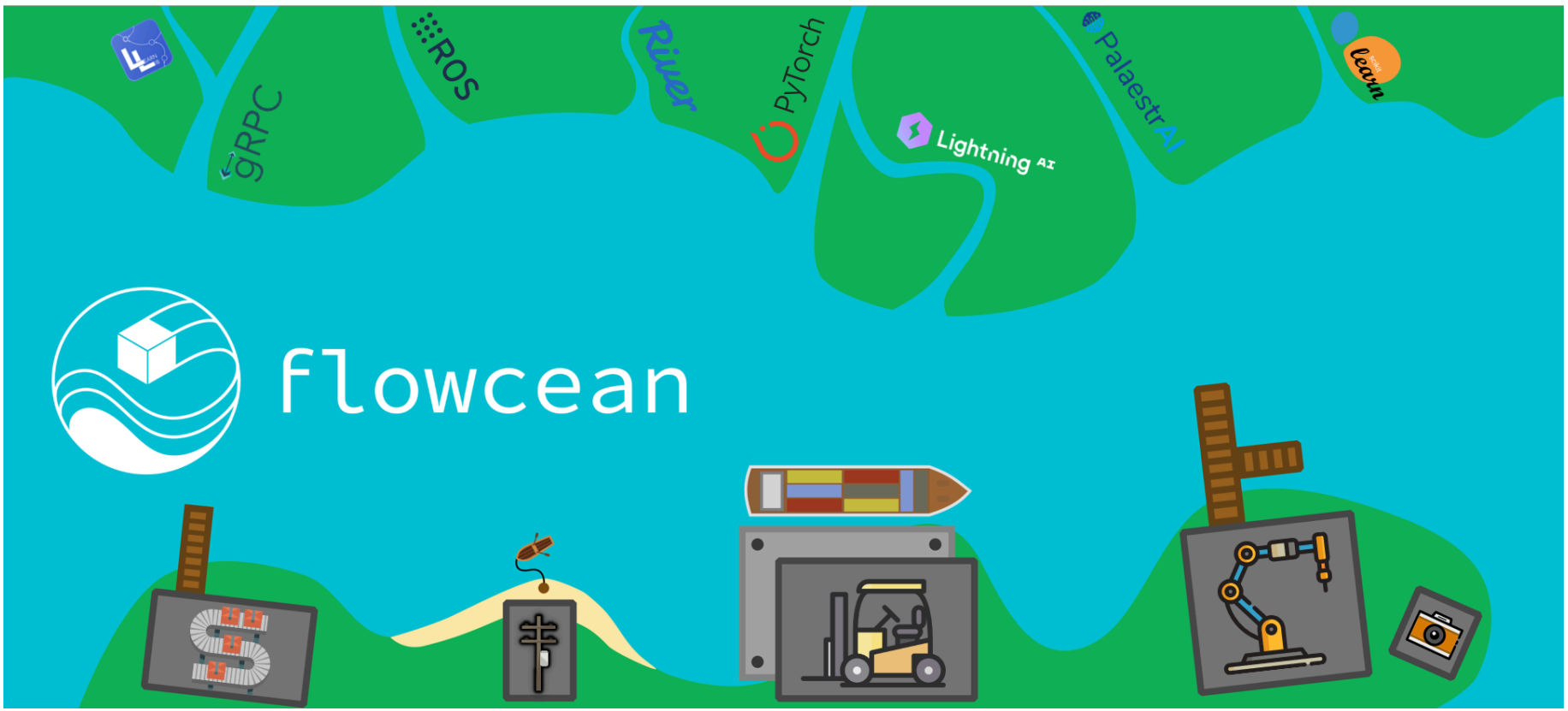
TUHH



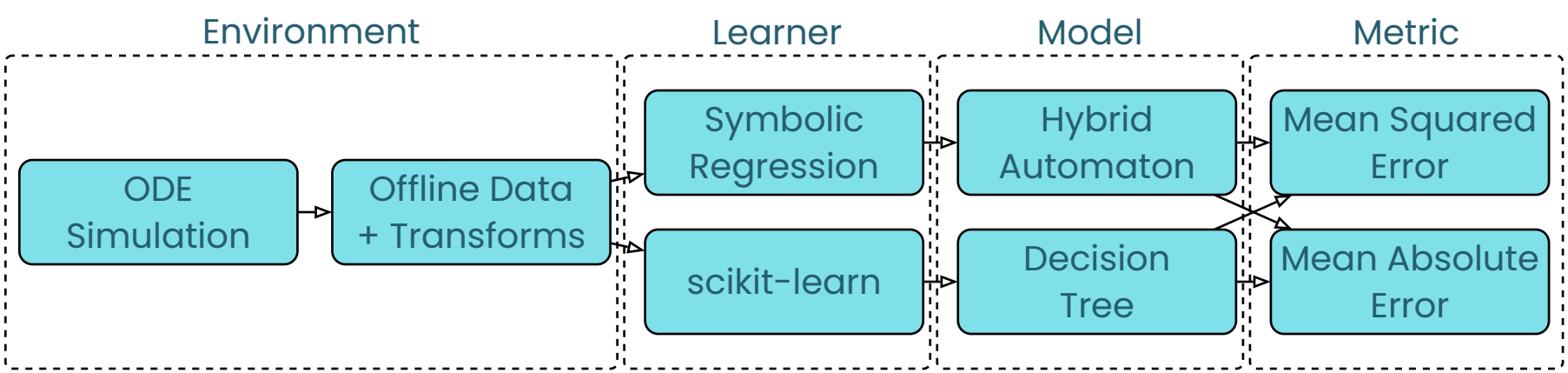
Learning Pipeline



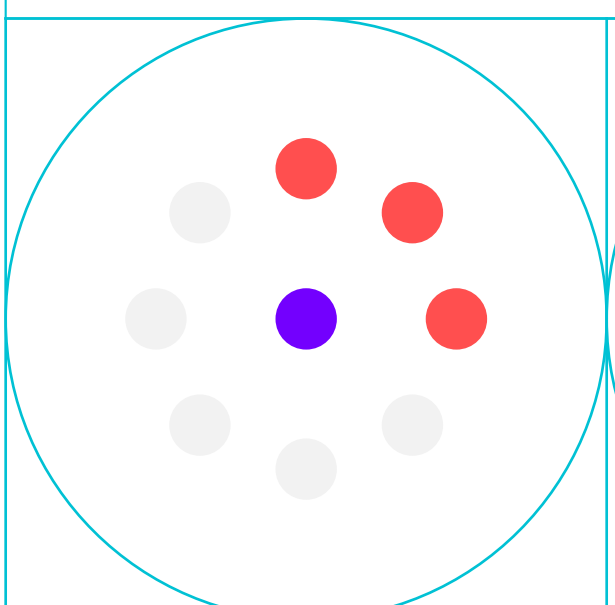
Modular Design



Modular Example



Unified Learning Framework



Data Sources & Preprocessing

TUHH

In Flowcean

- *environments* represent data sources
- *transforms* execute operations on data such as preprocessing steps

```
environment = DataFrame.from_csv(path)
(train, test) = TrainTestSplit(ratio=0.8, shuffle=False).split(environment)
window = SlidingWindow(window_size=4)
```

Model Learning

In Flowcean

- a *learner* is a learning algorithm
- the *model*, i.e., the result of the learning step differentiates from the learner
- a *learning strategy* describes how the learner interacts with the environment
  - e.g., offline learning, incremental learning, active learning

```
learner = RegressionTree(max_depth=5, random_state=42)
model = learn_offline(train, learner, INPUTS, OUTPUTS)
```

Evaluation

TUHH

In Flowcean

- an *evaluation strategy* describes how the model is assessed

```
report = evaluate_offline(
    model,
    test,
    INPUTS,
    OUTPUTS,
    [MeanAbsoluteError(), MeanSquaredError()],
)
```

Application

TUHH

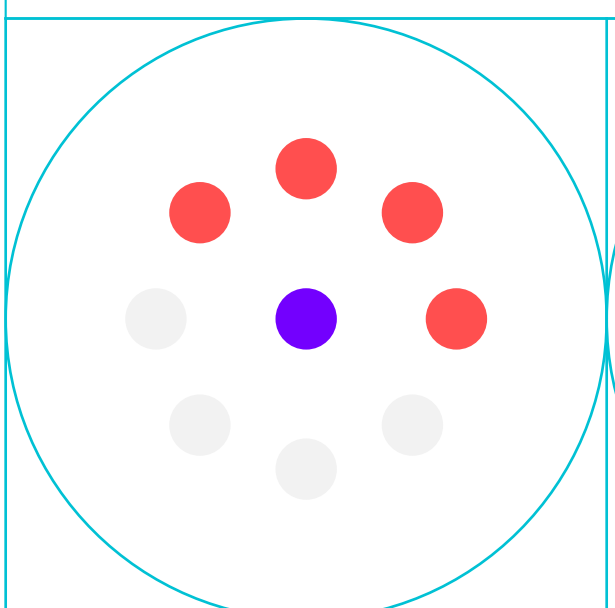
In Flowcean

- the learned model can be used for several applications such as
  - monitoring
  - testing
  - prediction

```
test_frame = test.observe()
predictions = model.predict(test_frame.select(INPUTS))
```

TUHH

Hands-On



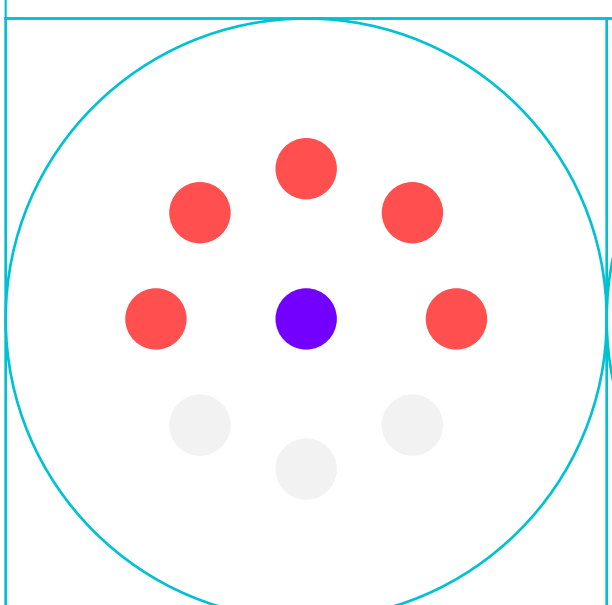
Hands On I

1. set up your working environment (if not yet done)
2. create your own Flowcean experiment based on the example

Flowcean has a modular design:

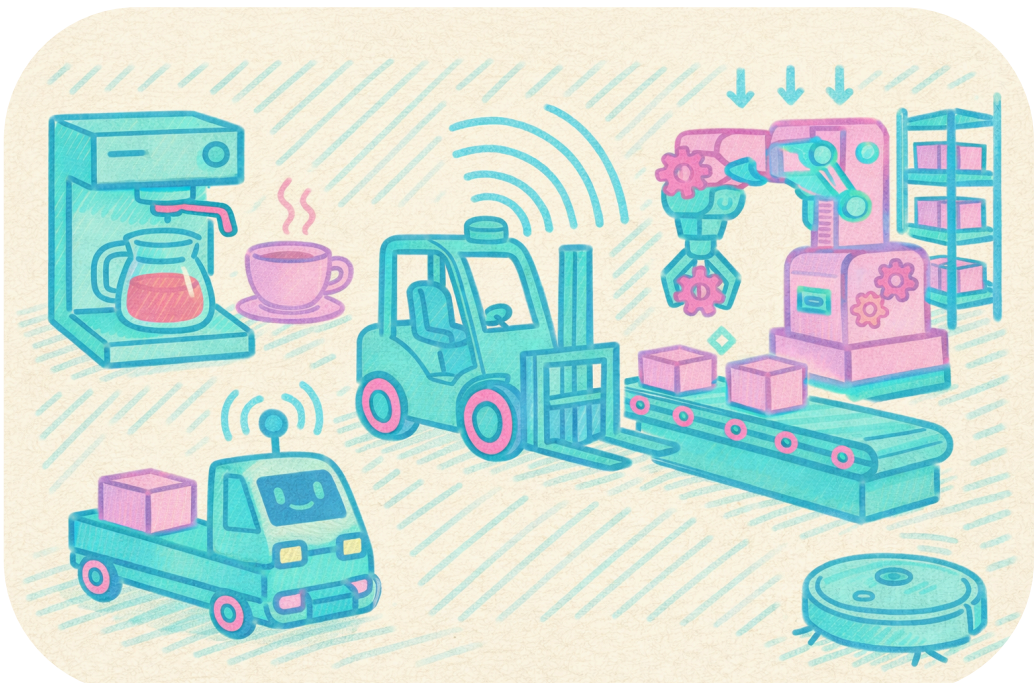
- easy replacement or adaption of transforms
- exchangable learners over diverse libraries (e.g., torch, sklearn)
- evaluation metrics and direct comparison of different models

Motivation for Hybrid Automata



Motivation

- Integration of **computation** with **physical** processes

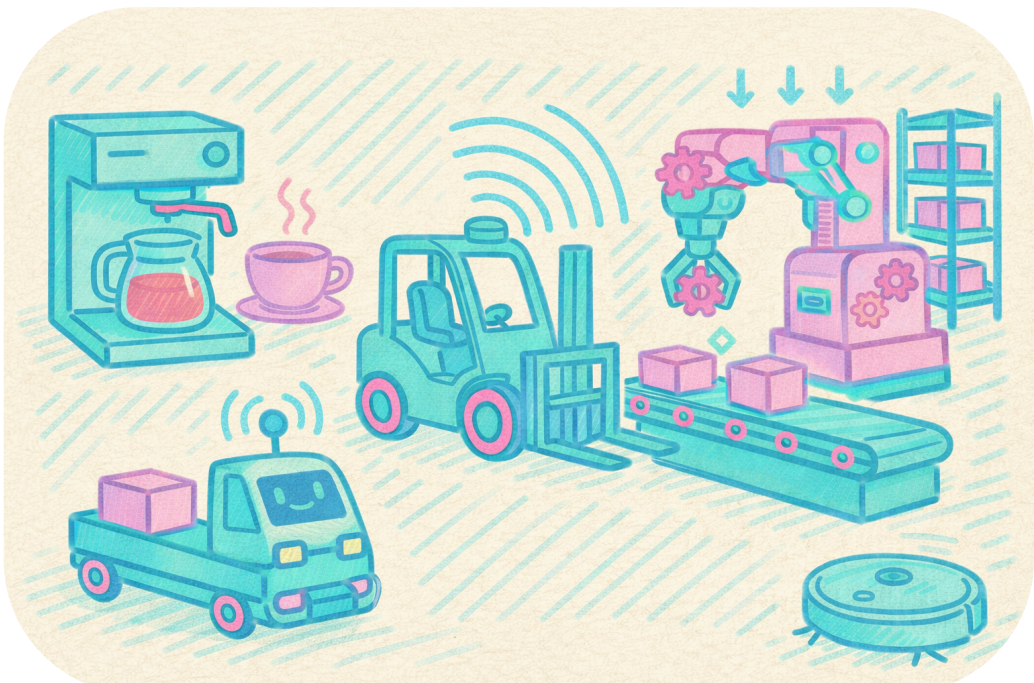


Motivation

- Integration of **computation** with **physical** processes
- Models serve **verification**, **control**, and **explanation**

Motivation

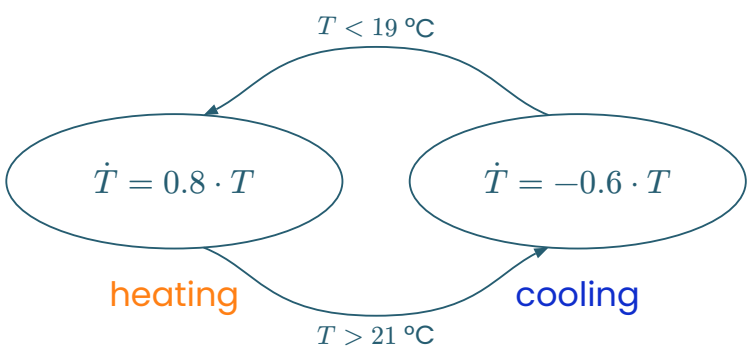
- Integration of **computation** with **physical** processes
- Models serve **verification**, **control**, and **explanation**
- **Time-consuming** manual modeling



Motivation

- Integration of **computation** with **physical** processes
- Models serve **verification**, **control**, and **explanation**
- **Time-consuming** manual modeling
- **Expert knowledge** required

Hybrid Automata



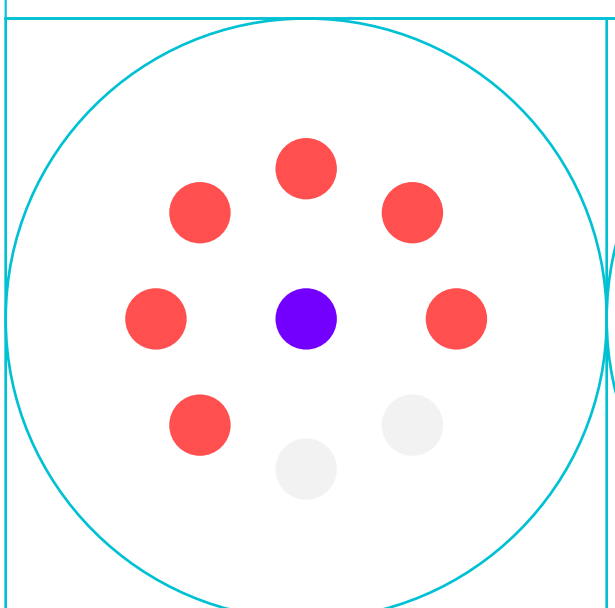
Structure

- combine **finite state machines** with **continuous dynamics**
- modes, flow functions, and discrete transitions

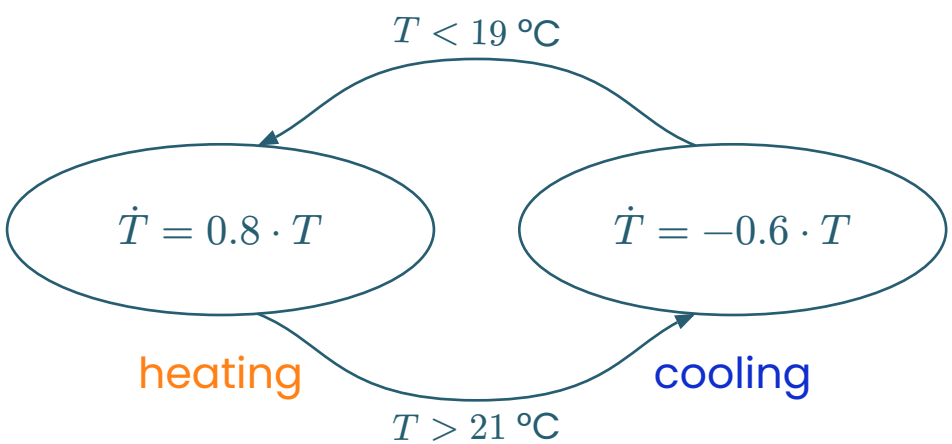
Properties / Advantages

- model systems with **both discrete and continuous** behavior
- capture complex behavior in **cyber-physical systems**

Simulation Models



Hybrid System Simulation

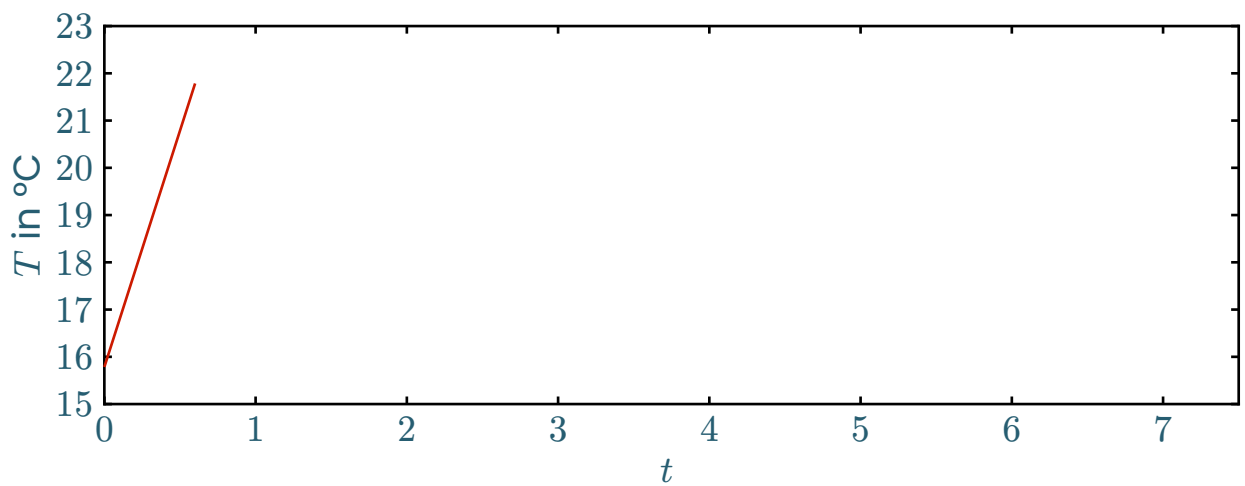


Hybrid System Simulation

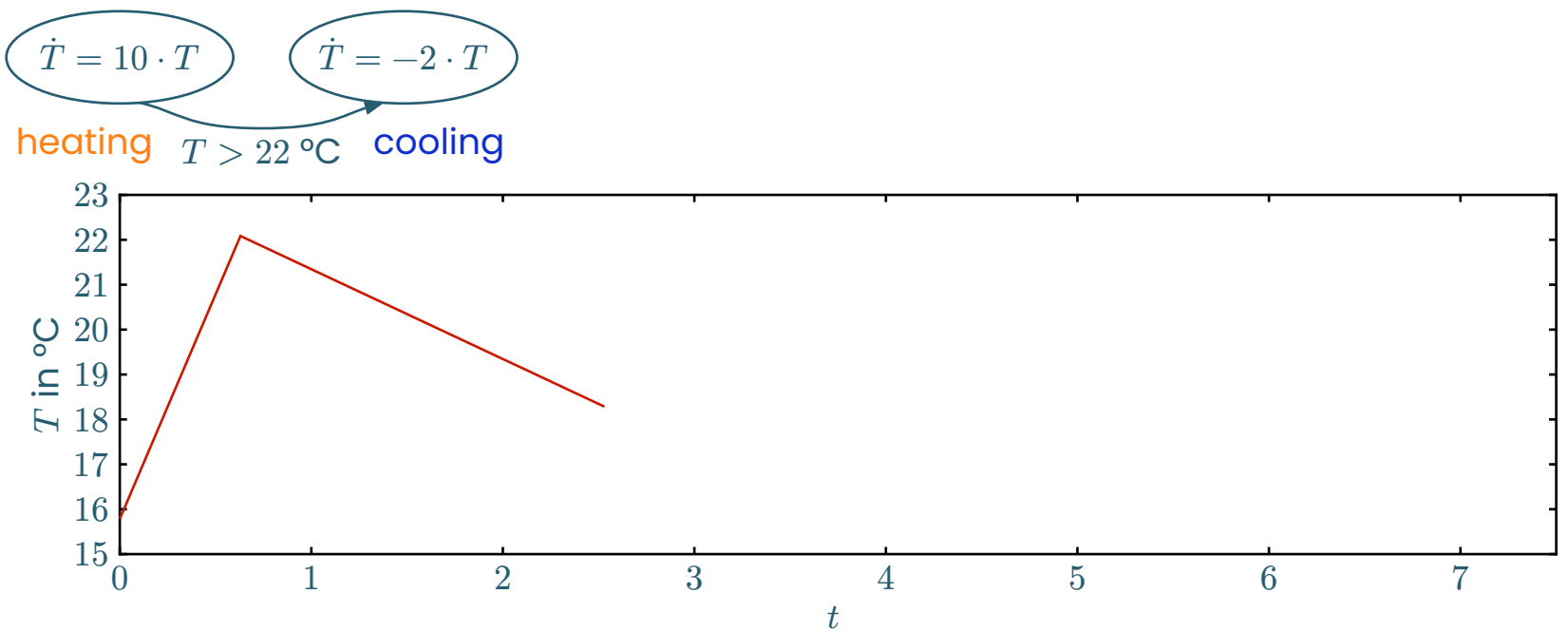
TUHH

$\dot{T} = 10 \cdot T$

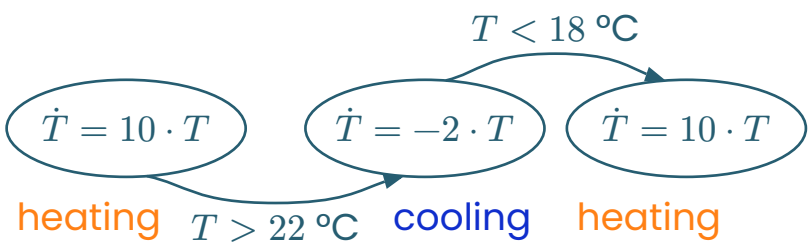
heating

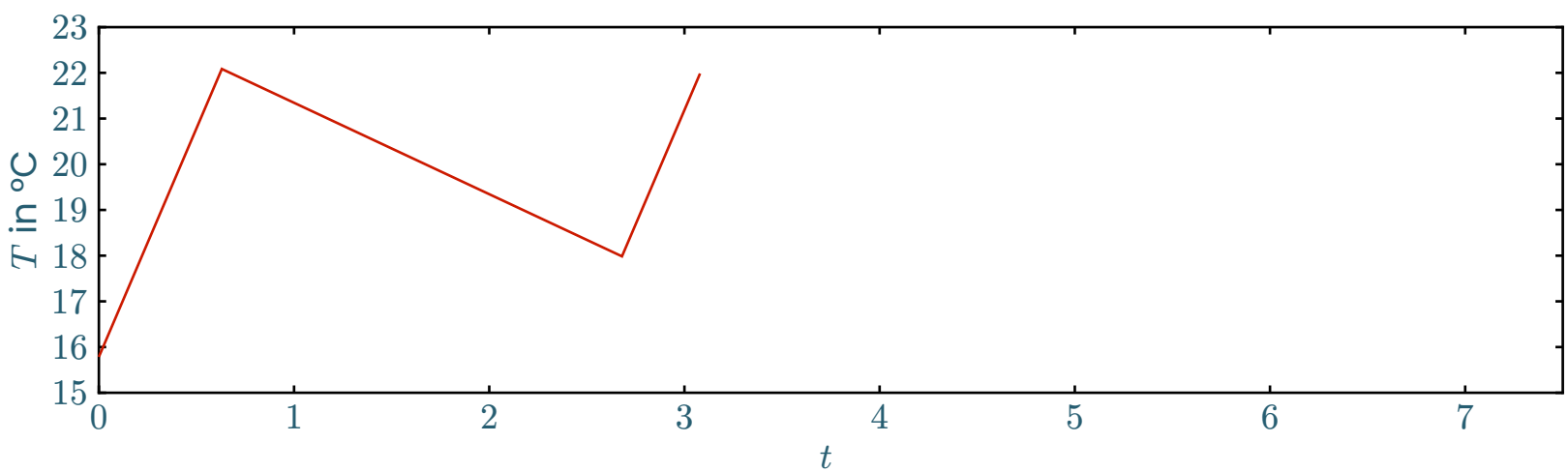


Hybrid System Simulation

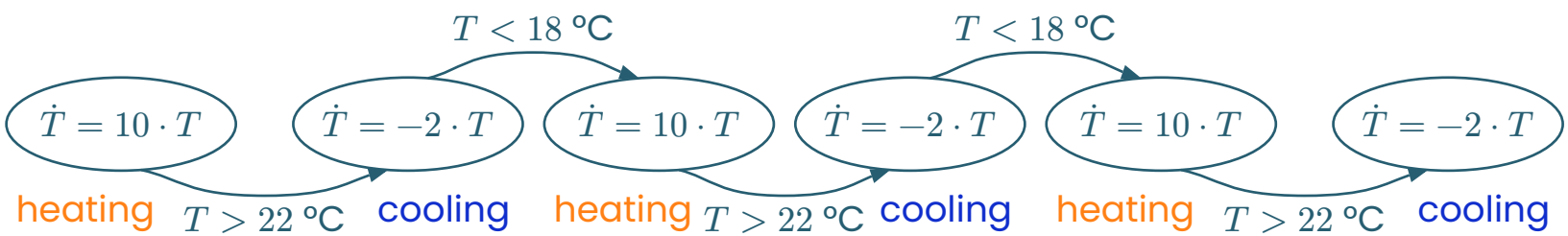


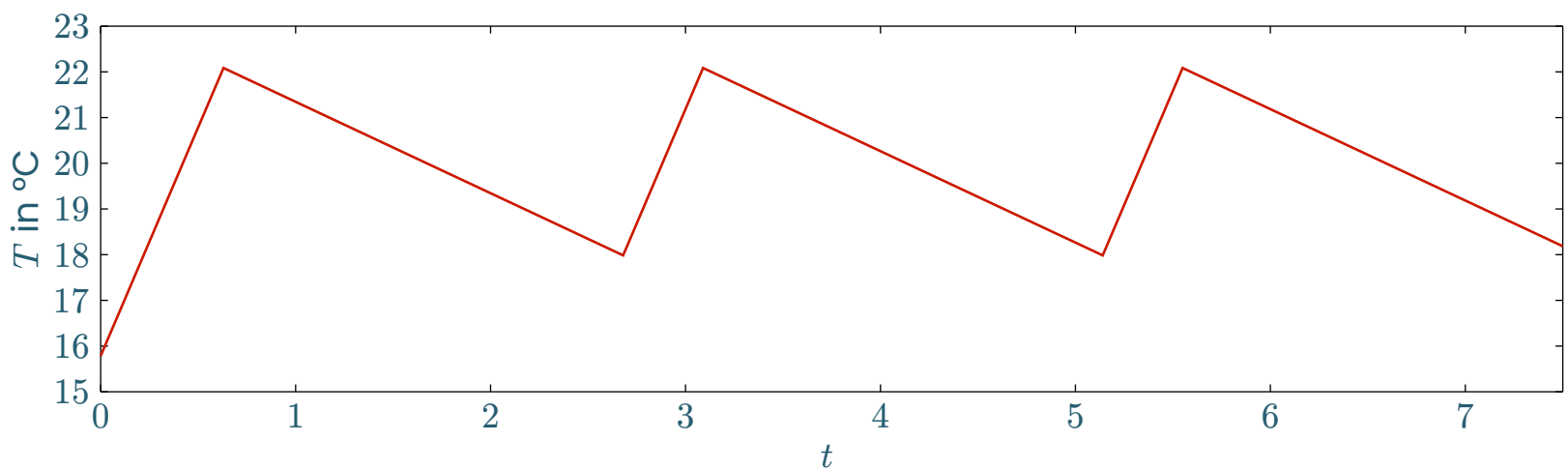
Hybrid System Simulation



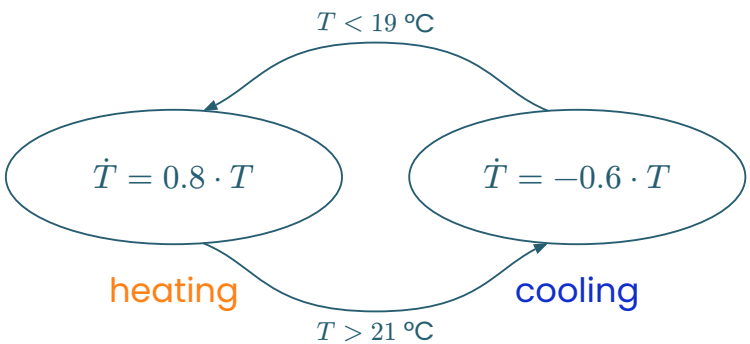


Hybrid System Simulation





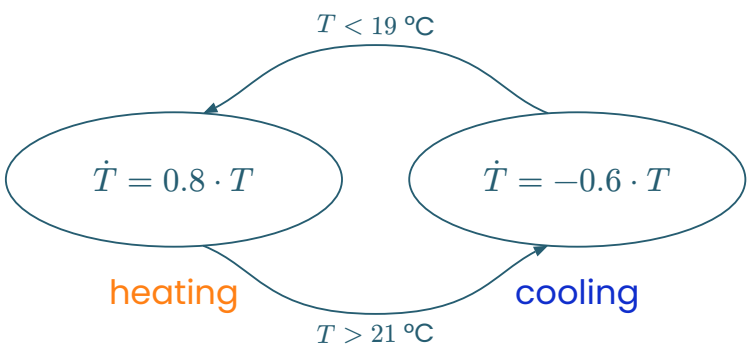
Define Continuous Dynamics




```python
def heating_flow() -> np.ndarray:
  """Warm the room at a constant rate."""
  return np.array([0.8])


def cooling_flow() -> np.ndarray:
  """Cool the room at a constant rate."""
  return np.array([-0.6])
```

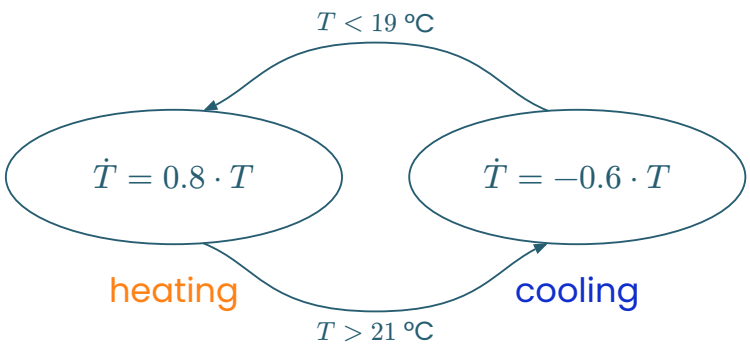
Define Locations




```
heating = Location(heating_flow, label="heating")

cooling = Location(cooling_flow, label="cooling")
```

Define Transition Conditions



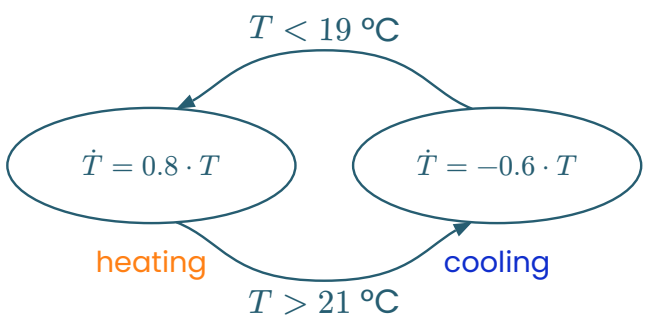

```python
def too_hot(state: np.ndarray) → float:
  """Switch from heating to cooling at the upper threshold."""
  return state[0] - 21.0


def too_cold(state: np.ndarray) → float:
  """Switch from cooling to heating at the lower threshold."""
  return state[0] - 19.0
```

Construct Hybrid System

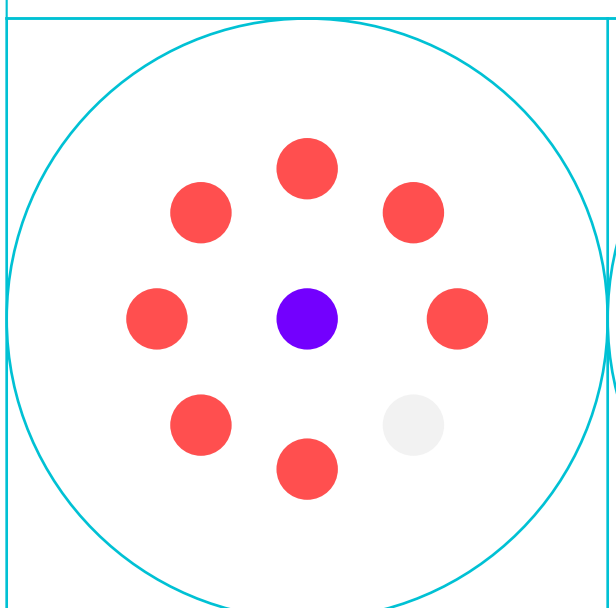
TUHH

```
thermostat = HybridSystem(
  locations=[heating, cooling],
  transitions=[
      Transition(
          source=heating,
          target=cooling,
          event=EventSurface(
              too_hot,
              direction=CrossingDirection.RISING,
              label="too_hot",
          ),
      ),
      Transition(
          source=cooling,
          target=heating,
          event=EventSurface(
              too_cold,
              direction=CrossingDirection.FALLING,
              label="too_cold",
          ),
      ),
  ],
  initial_location=heating,
  initial_state=np.array([19.0]),
)
```

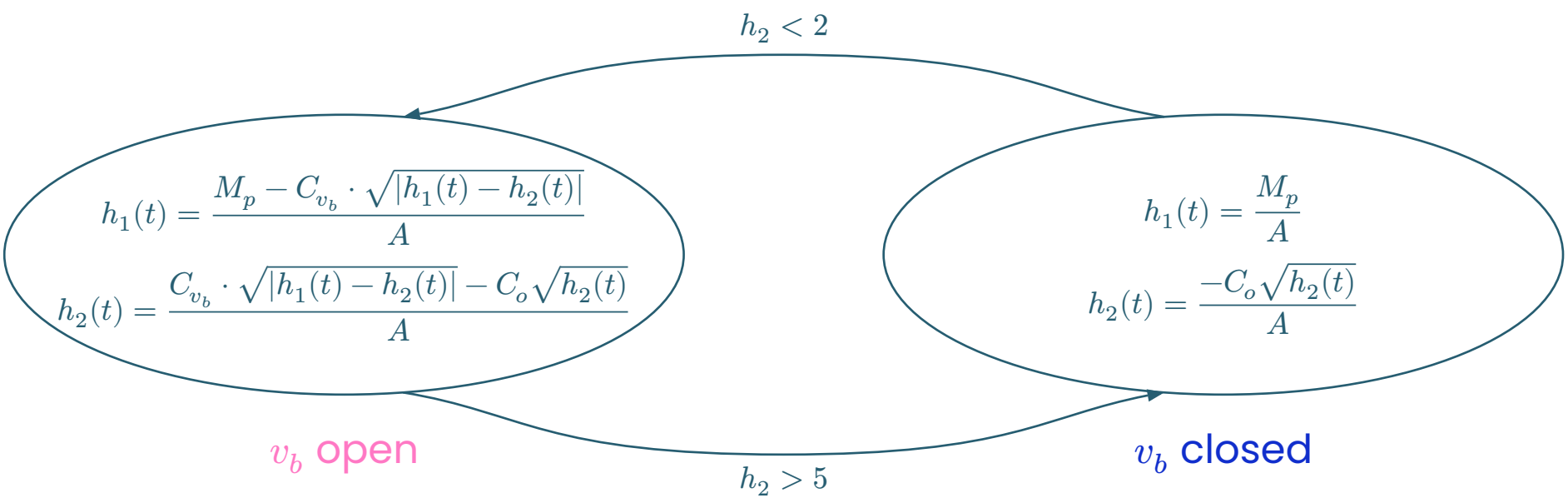


TUHH

Hands-On

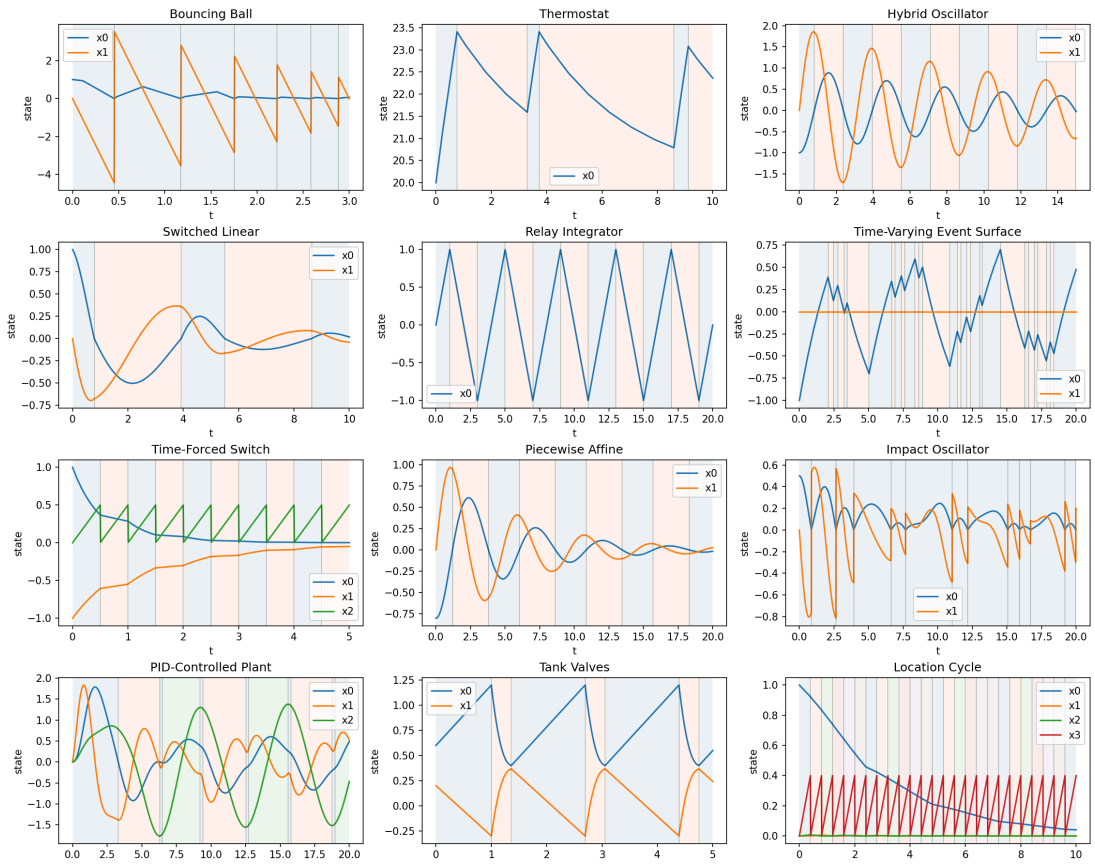


Implementation of our own hybrid system model

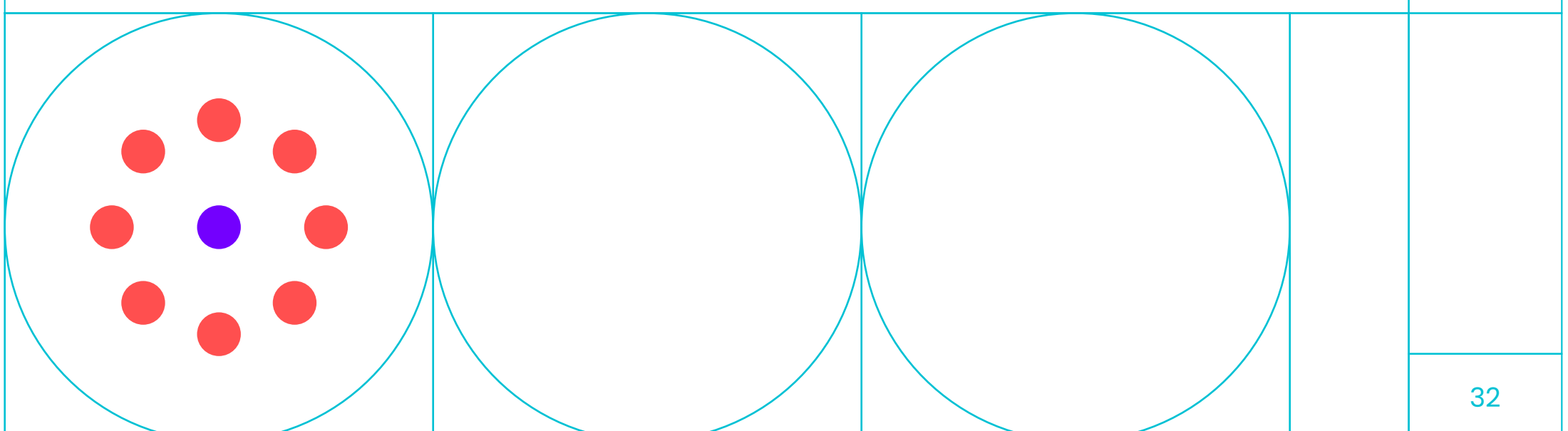


Initial values: $h_1 = 12, h_2 = 4$

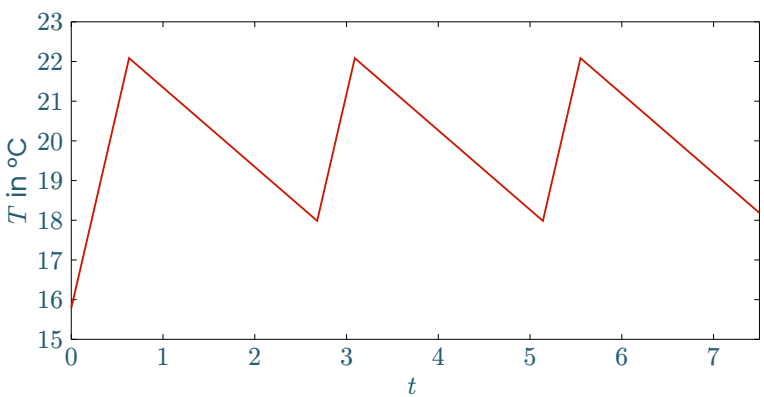
Benchmark Gallery

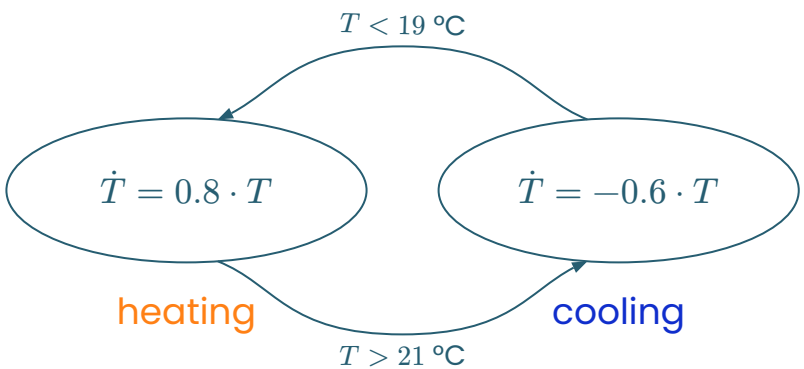


Learning Hybrid Automata

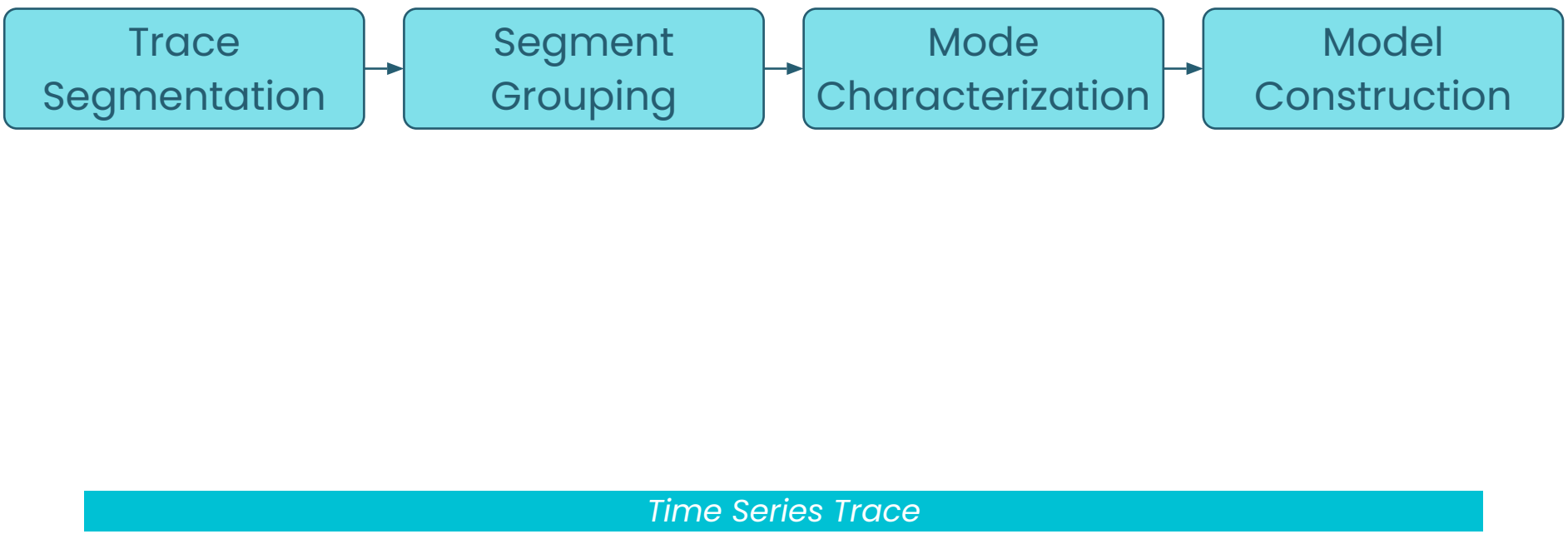


Hybrid System Identification — Problem Statement



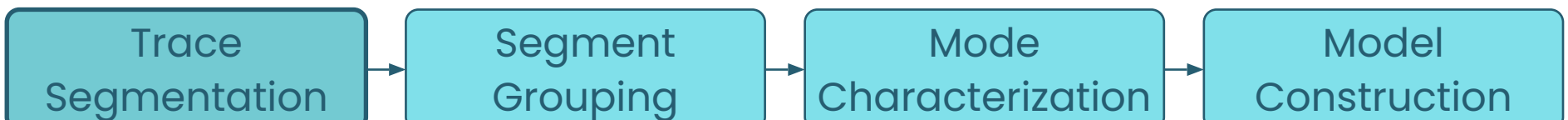


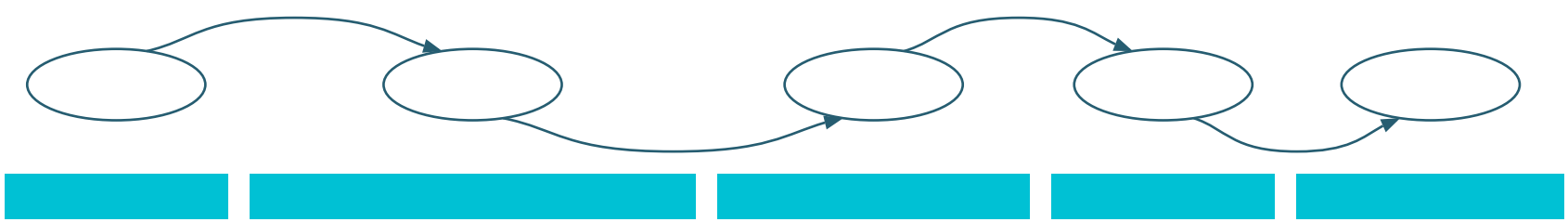
Hybrid System Identification — Traditional Approach



Hybrid System Identification — Traditional Approach

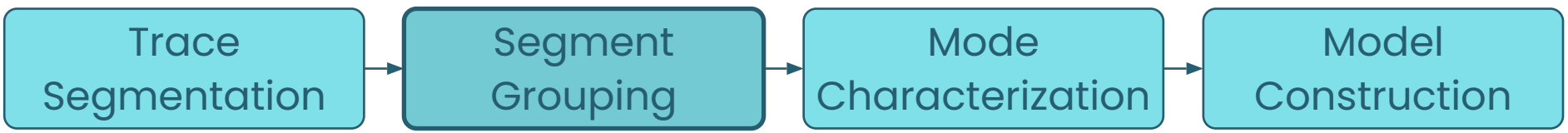
TUHH

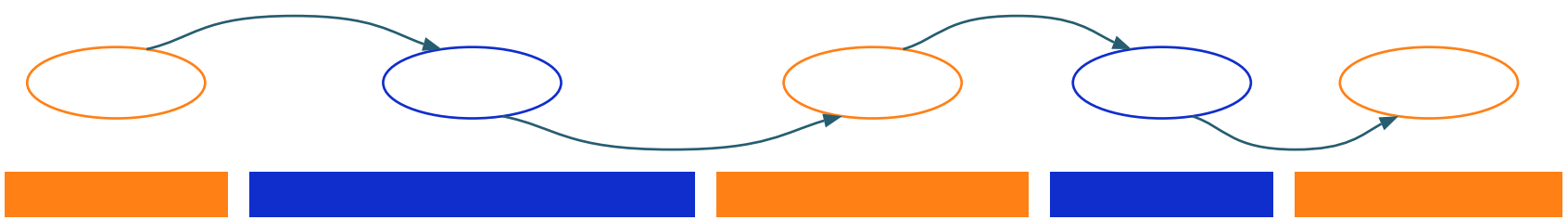




Hybrid System Identification — Traditional Approach

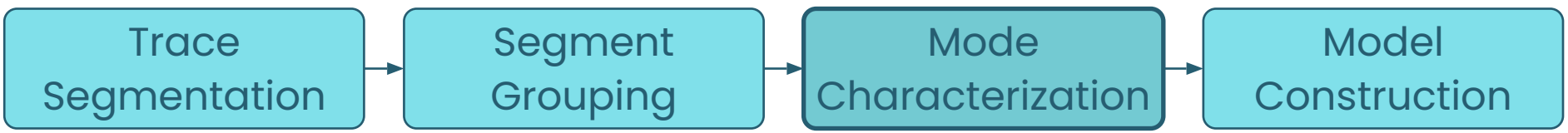
TUHH

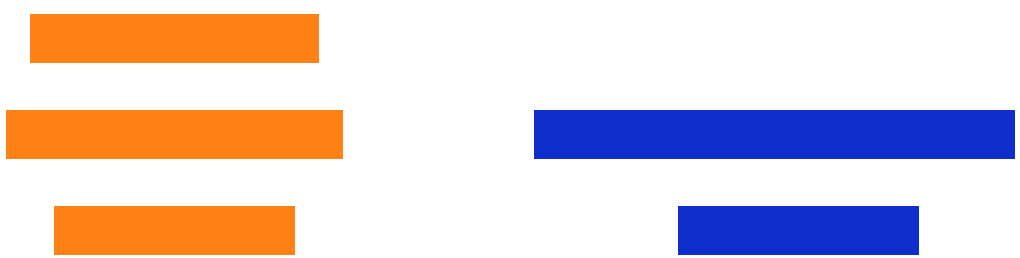




Hybrid System Identification — Traditional Approach

TUHH



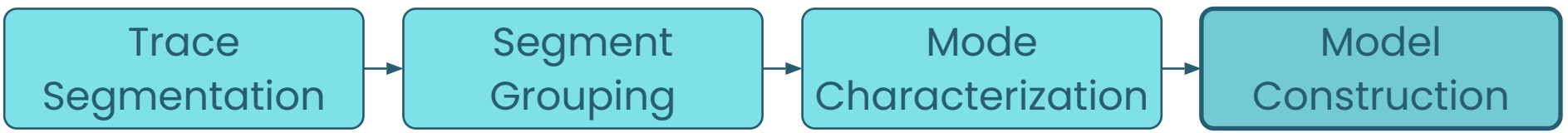


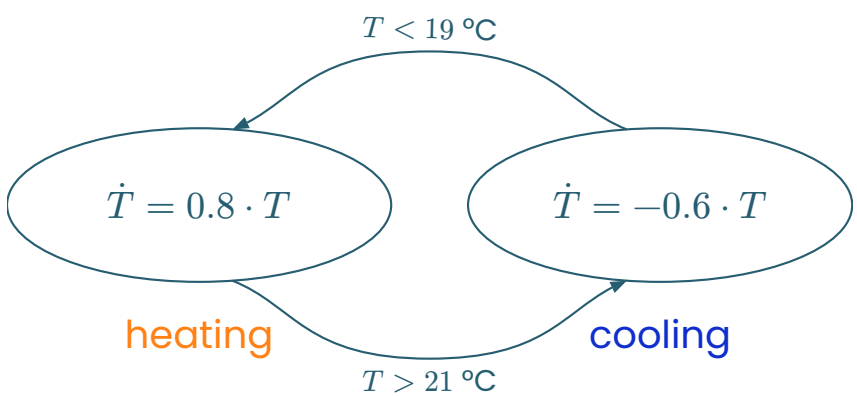
$\dot{T} = 10 \cdot T$

$\dot{T} = -2 \cdot T$

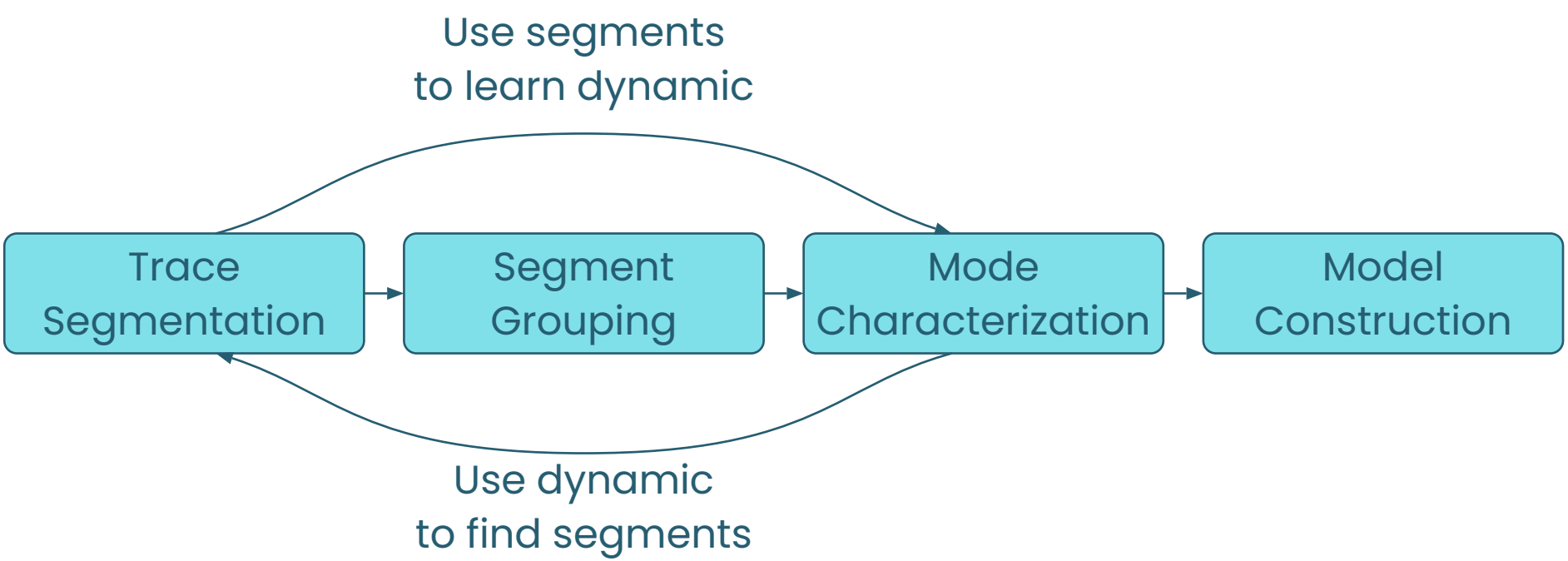
Hybrid System Identification — Traditional Approach

TUHH



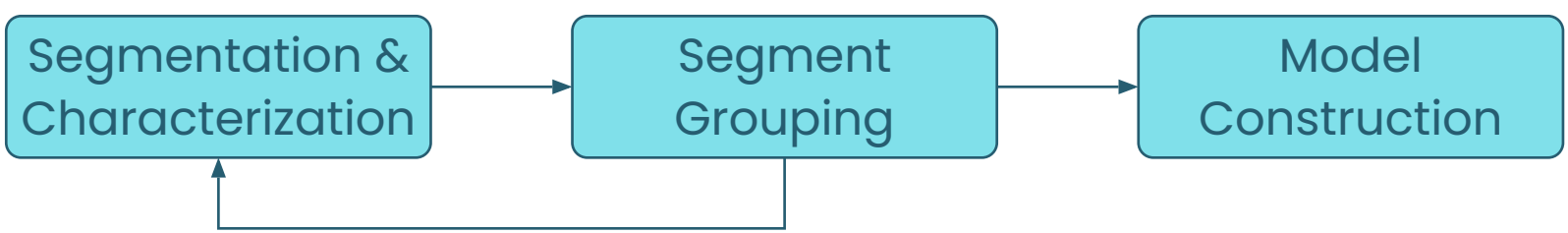


Dynamics-Based Hybrid System Identification

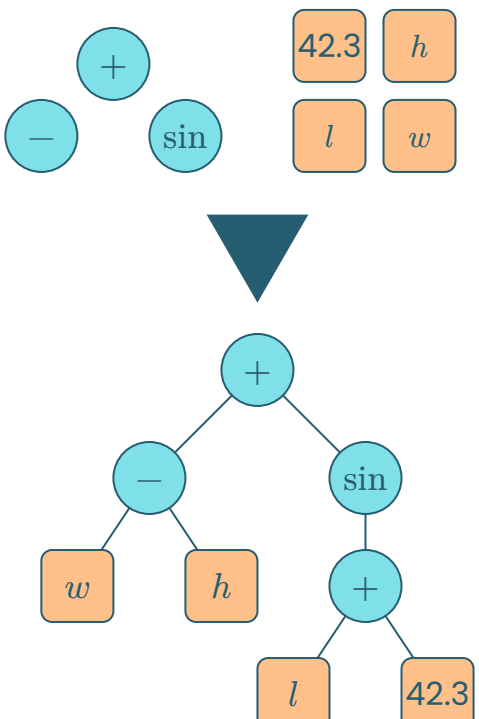


HyDRA

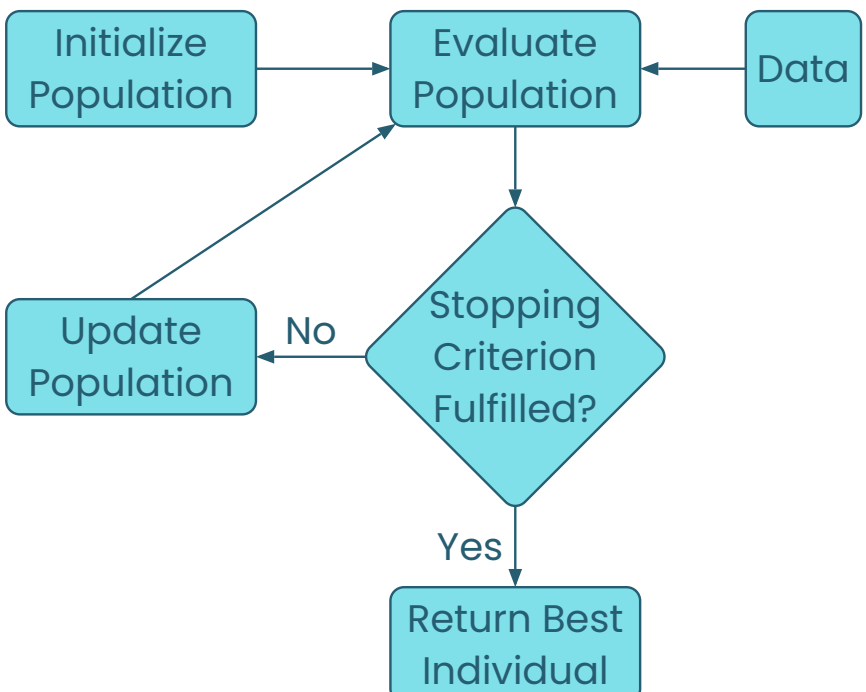
Hybrid systems via Dynamics and Regression-based Analysis



Candidate Expression

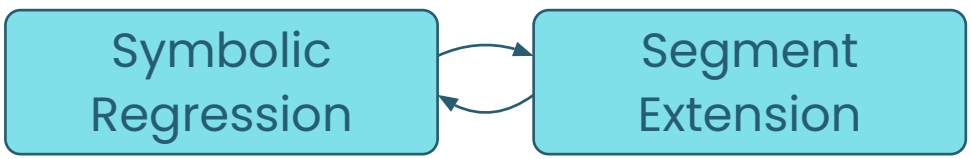


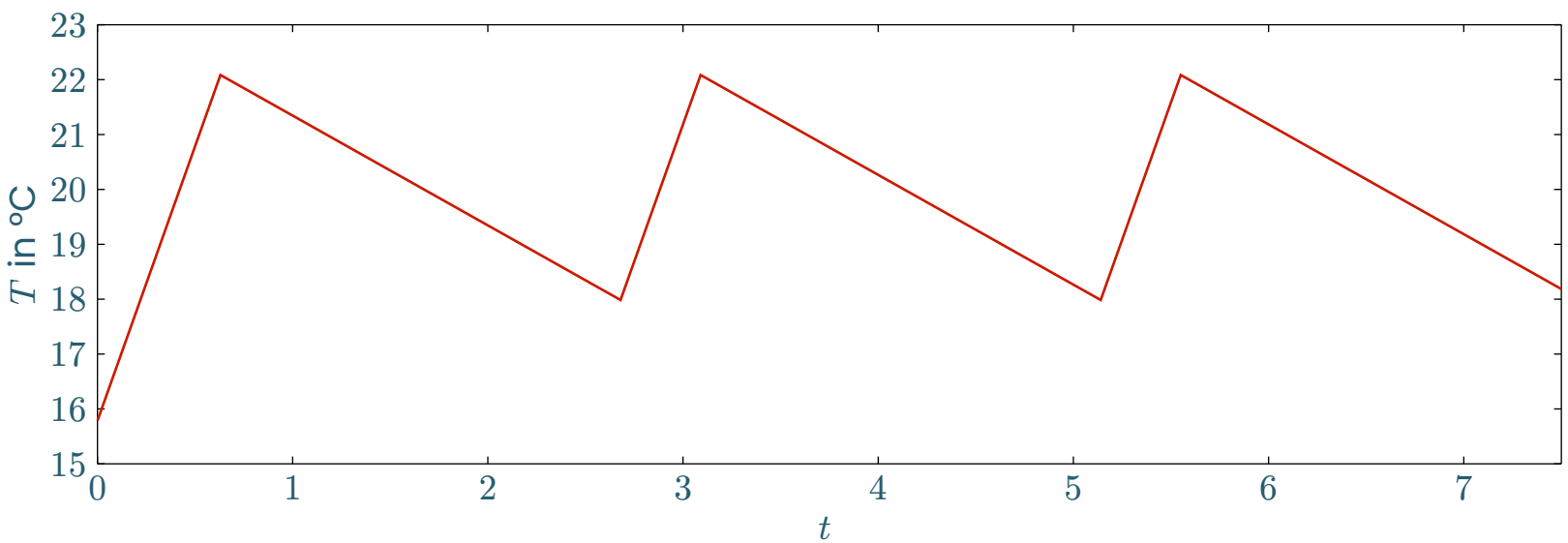
Learning Process



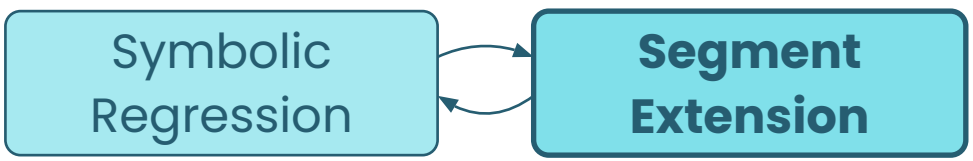
Dynamics-Based Hybrid System Identification

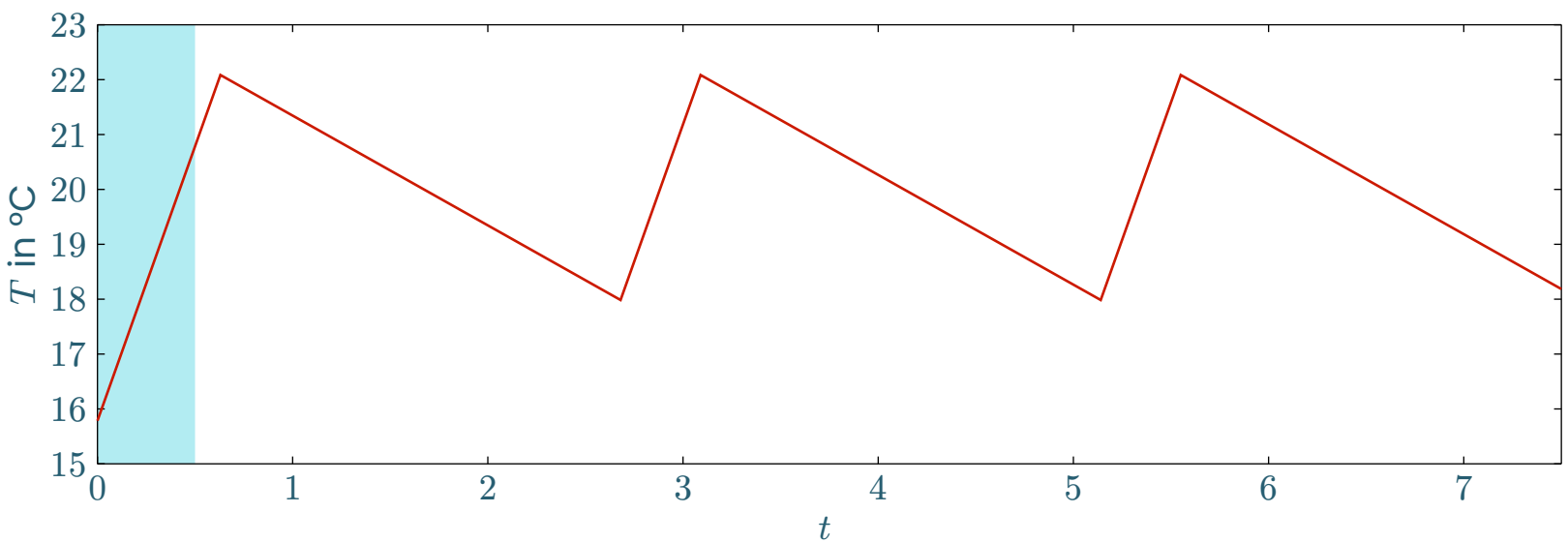
TUHH



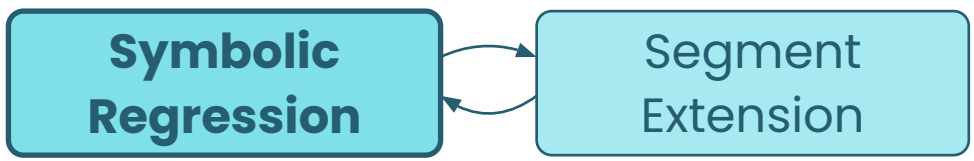


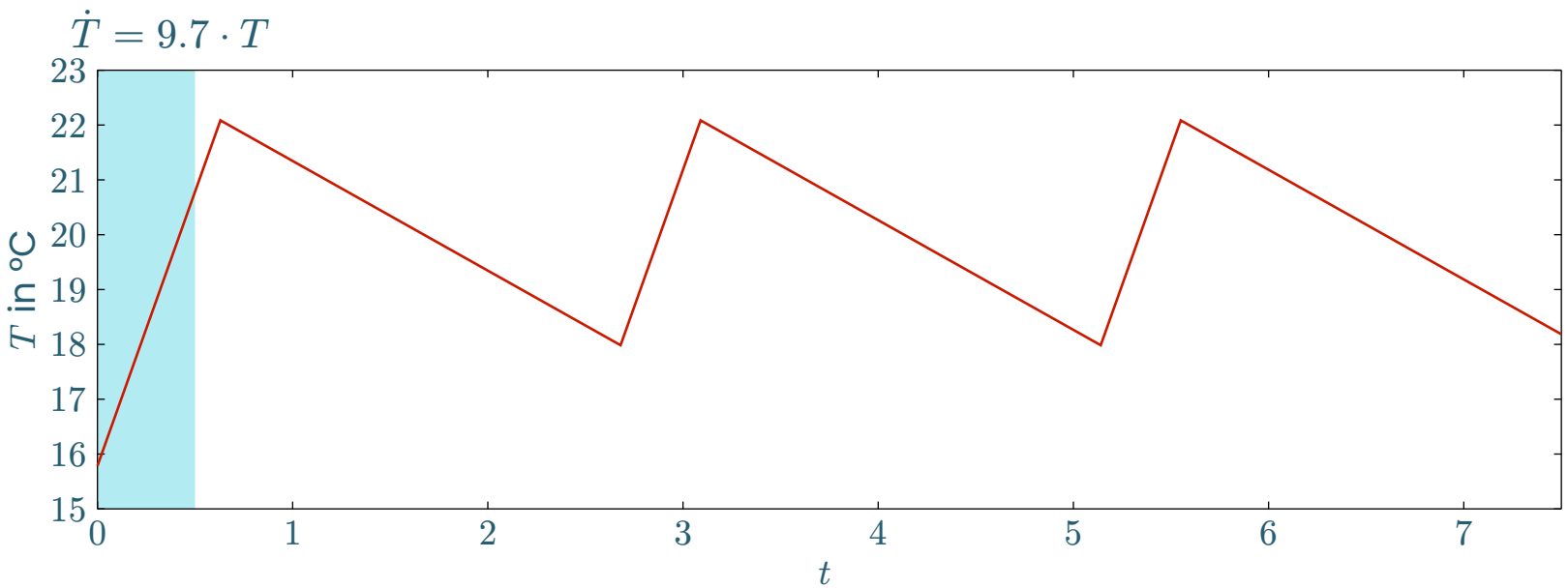
Dynamics-Based Hybrid System Identification



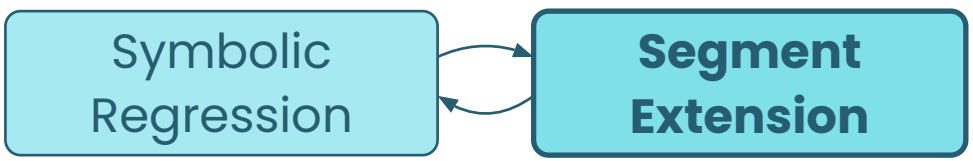


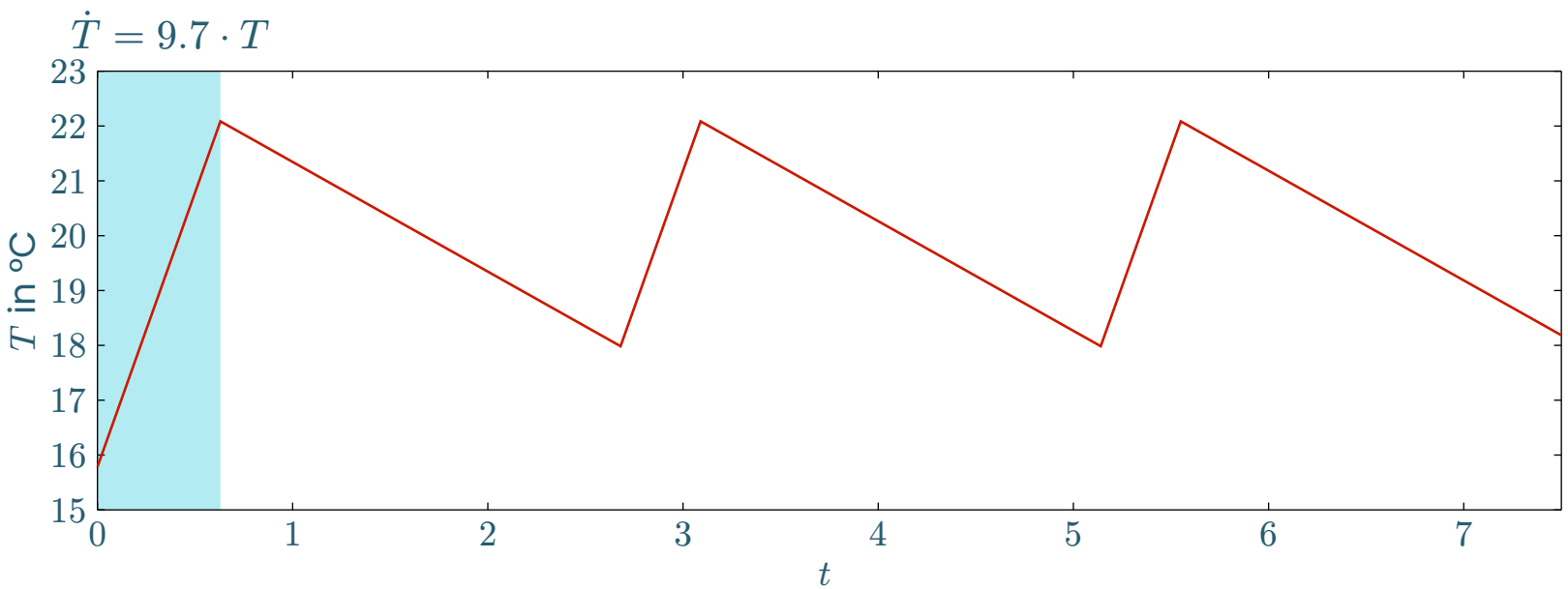
Dynamics-Based Hybrid System Identification



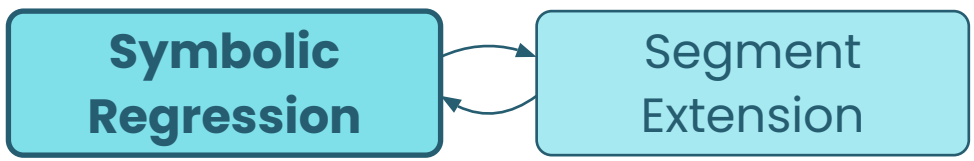


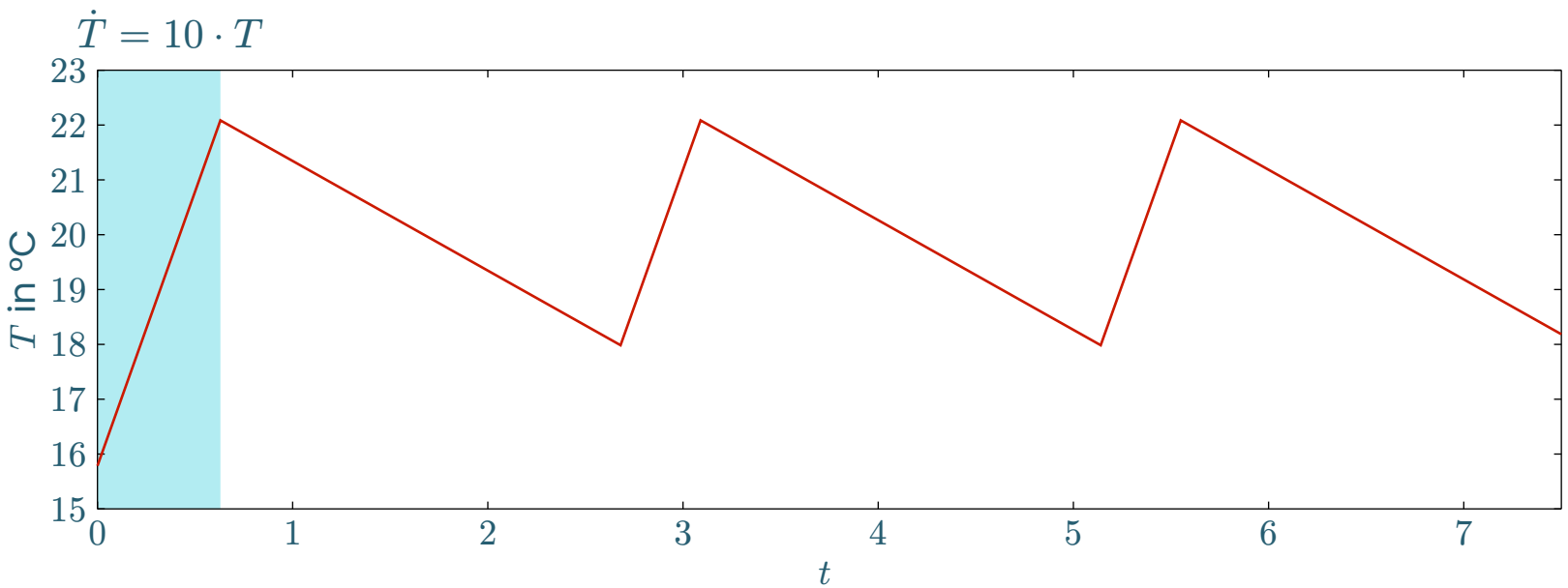
Dynamics-Based Hybrid System Identification



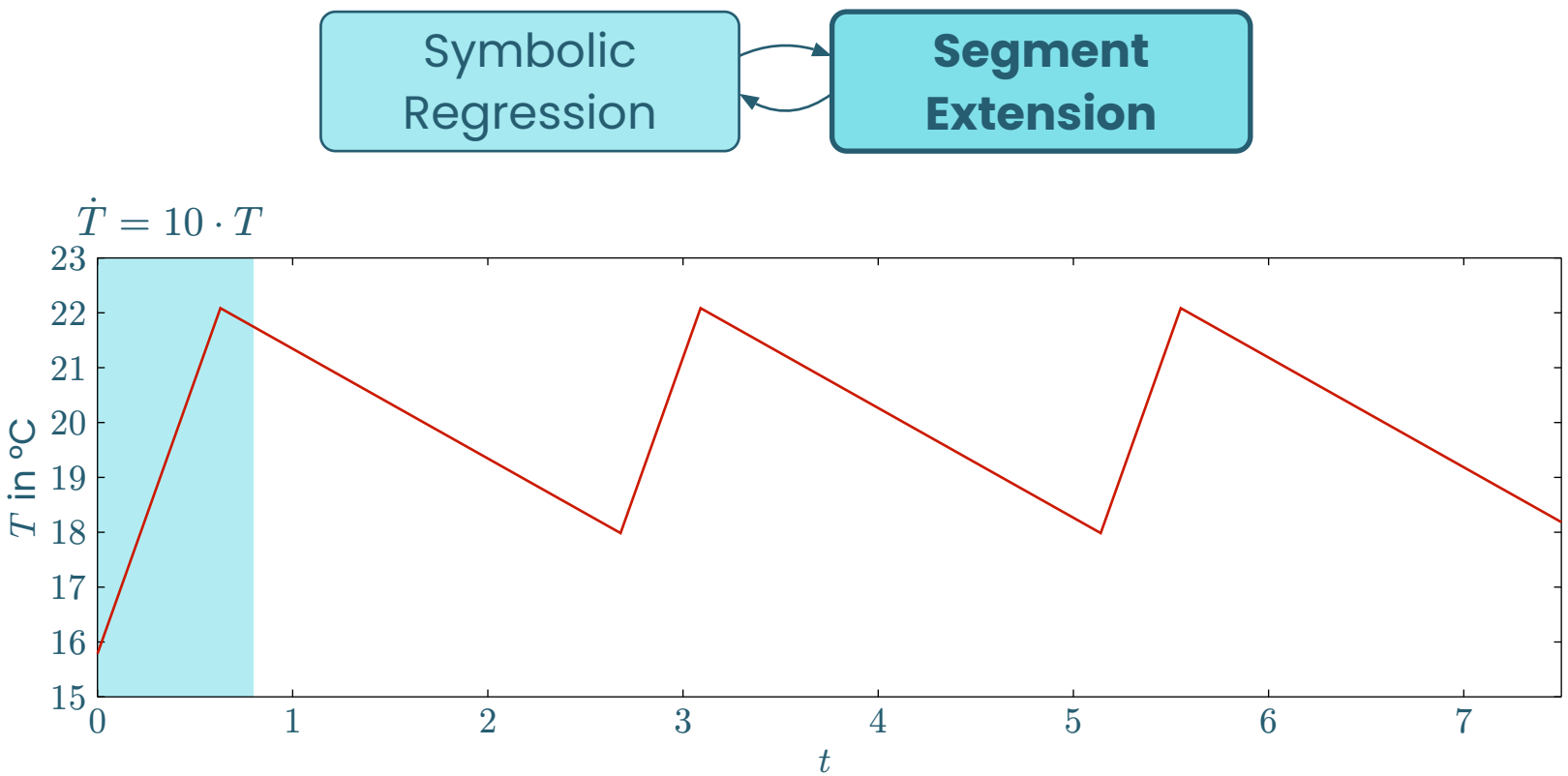


Dynamics-Based Hybrid System Identification

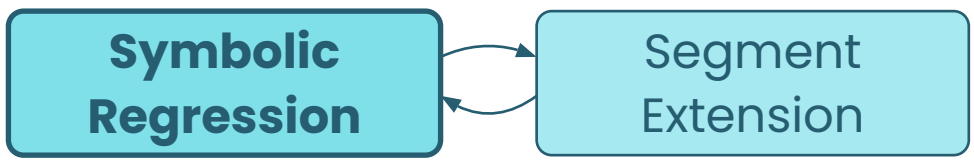


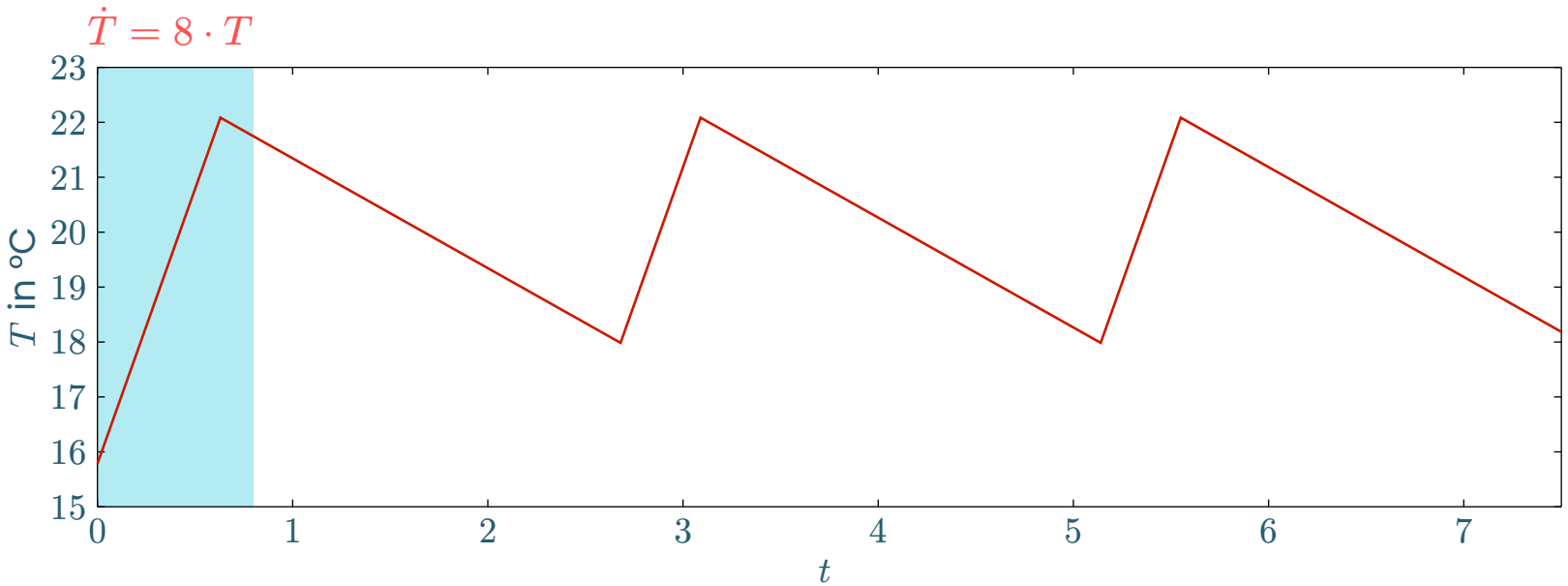


Dynamics-Based Hybrid System Identification

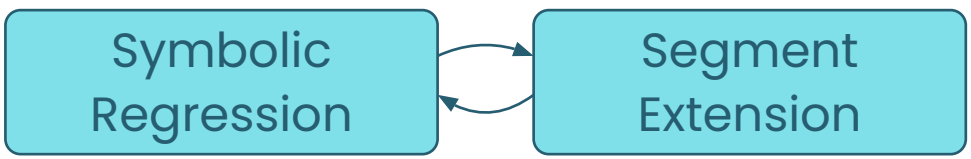


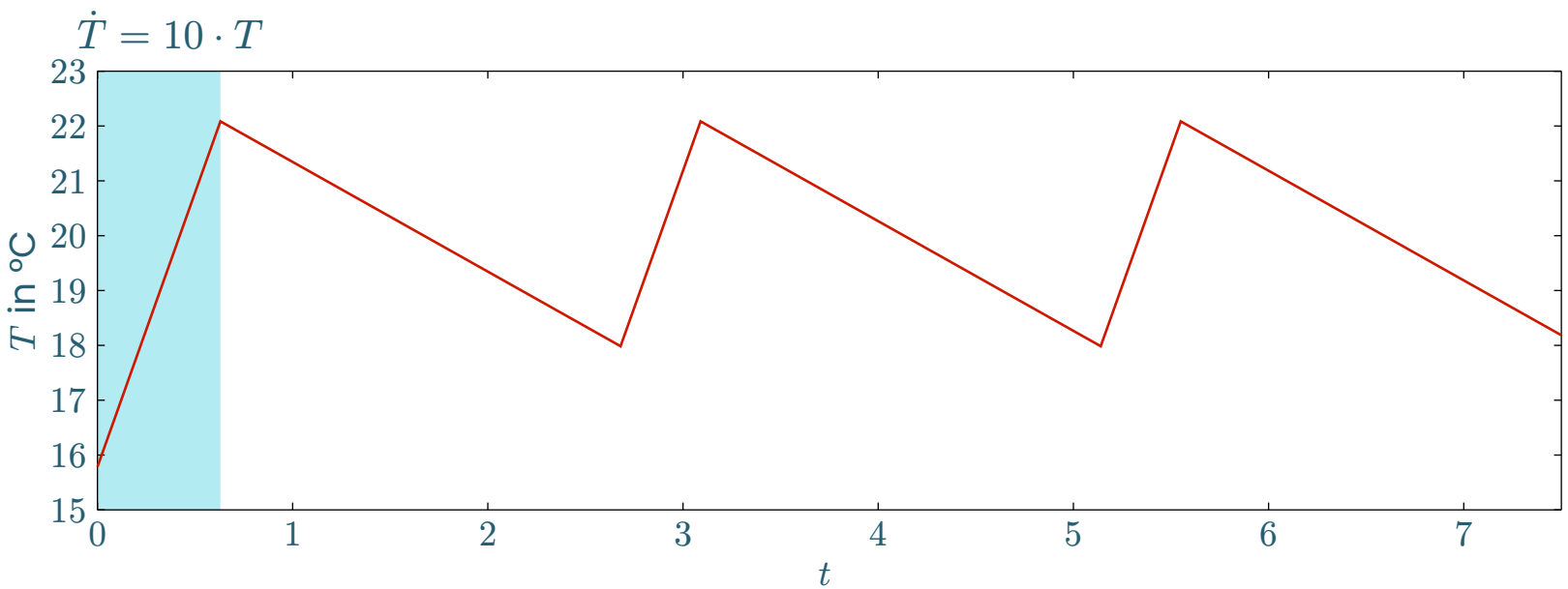
Dynamics-Based Hybrid System Identification



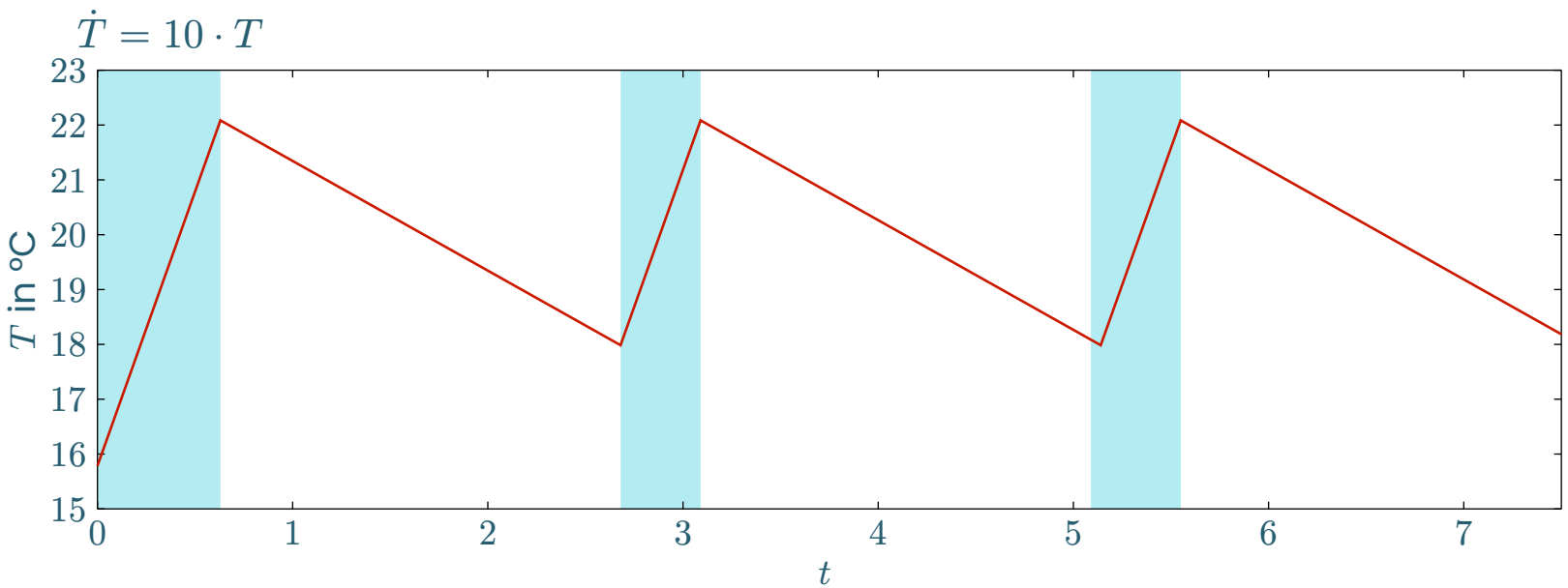


Dynamics-Based Hybrid System Identification

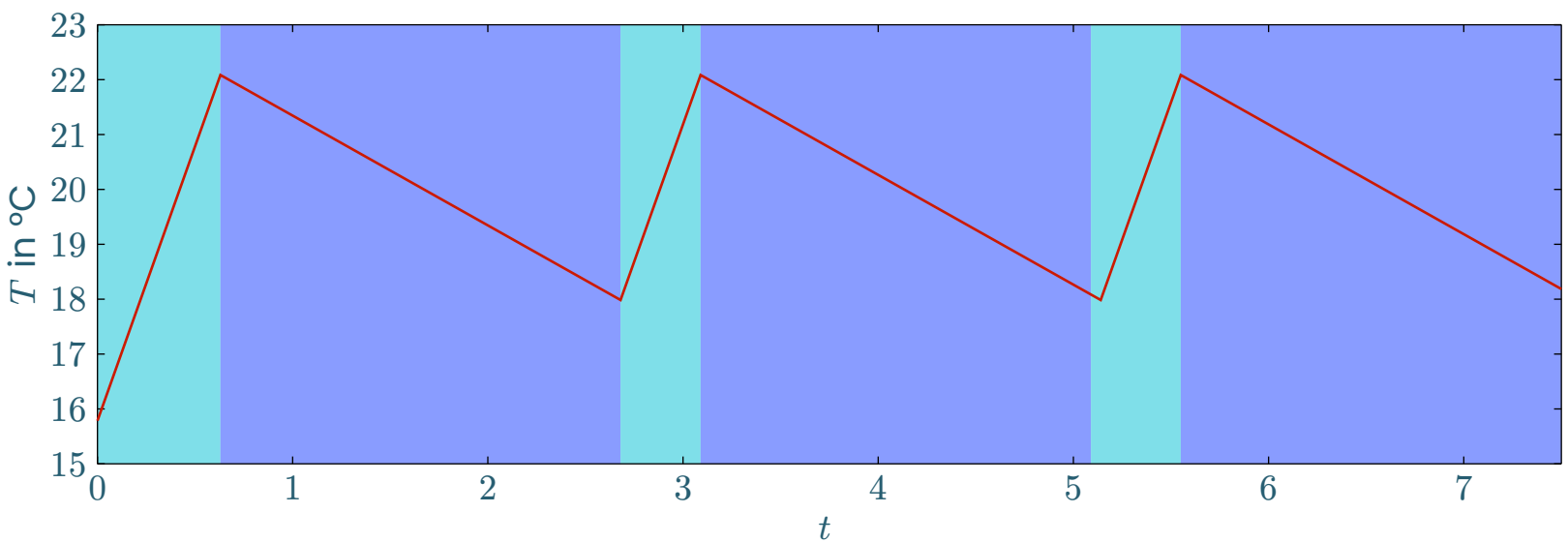




Segment Grouping



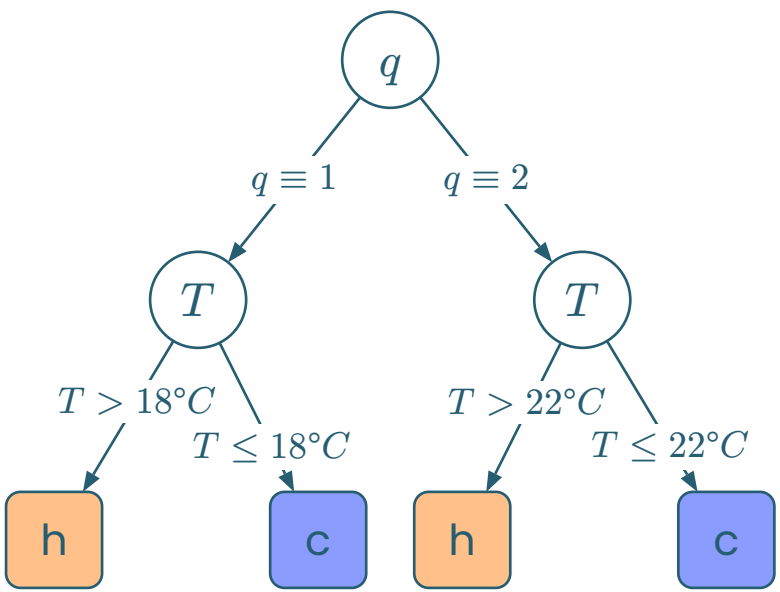
Segment Grouping



Hybrid Decision Tree

TUHH

Decision Tree



Flow Functions

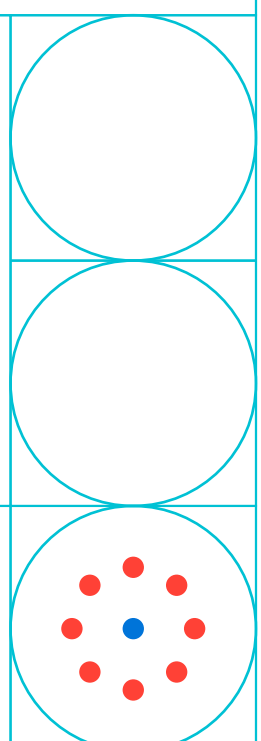
h	$\dot{T} = 10 \cdot T$
c	$\dot{T} = -2 \cdot T$

Thank You!
Questions?

Maximilian Schmidt & Swantje Plambeck
firstname.lastname@tuhh.de

Institute of Embedded Systems
Hamburg University of Technology (TUHH)
Am Schwarzenberg-Campus 3
21073 Hamburg

tuhh.de



TUHH
Hamburg
University of
Technology