JSON Dataset of Simulated Building Heat Control for System of Systems Interoperability

Jacob Nilsson

March 21, 2022

Abstract

Machine learning based interoperability solutions are currently hard to test due to a lack of standardized datasets and testbeds. This technical report contains the details of an openly available simulation model and data set, which can be used by interoperability researchers to test, reproduce, and compare models and approaches.

1 Introduction

System of Systems (SoS) interoperability based on machine learning is currently in its infancy [1, 2]. Currently, a lack of open datasets and testbeds makes it hard to test and evaluate approaches. In this technical report, I present a dataset based on a heating simulation of a building. The data generated is not as complex as real data, but I hope to inspire more interoperability researchers to share their data openly as well.

1.1 SoS Interoperability

There is a push for industrial systems to share information at a greater extent to increase efficiency, reduce waste, and provide greater customer support, called the fourth industrial revolution [3]. Central to the fourth industrial revolution is the use of Service-Oriented Architectures (SOAs) and SoSs, whose techniques could in principle allow for data sharing between heterogeneous and autonomous systems. Such information sharing is called *interoperability* [4], which comes in many different forms [5].

The dataset presented in this report allows users to test *semantic* and *operational interoperability*. Semantic interoperability refers to systems' and

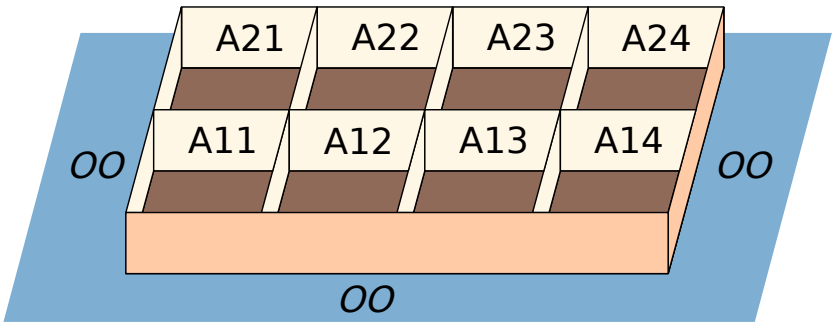


Figure 1: Room layout of simulation. The temperatures of rooms A11 – A24 are controlled by heating and cooling units that communicate using SenML-JSON. The outside temperature, OO, is taken from real temperature data.

operators' ability to interpret and share the meaning of data, and operational interoperability refers to systems' ability to cooperate towards goals individual subsystems cannot complete by themselves.

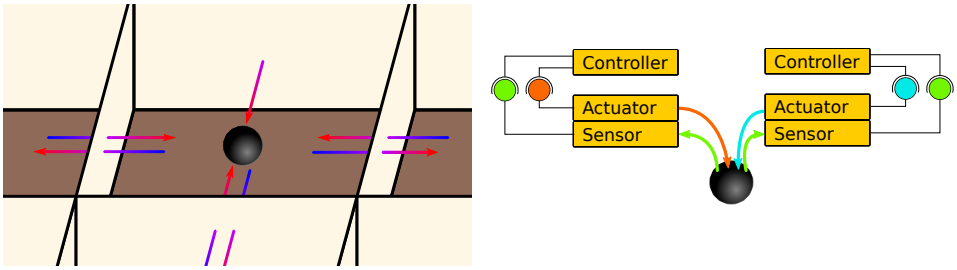
2 Simulation Environment

The data provided is generated from a thermodynamic simulation of a building. The building, shown in Figure 1, contains eight rooms whose temperature is controlled, and the outside "room" whose temperature is not controlled. Heat is only exchanged through the walls of the rooms, the ground and ceiling are perfectly heat insulated. Each room, except the outside, contains a heater system and cooler system. Each heater and cooler system in turn consists of a temperature sensor, PI controller, and actuator. These systems communicate through JSON-SenML messages[1], but the message formats are different between the heater and cooler systems.

2.1 Thermodynamic Simulation

The simulation environment consists of eight rooms organized as two rows of four, with heat exchanged between rooms that shares sides (Figure 1). Each room is also influenced by an outside temperature which is unaffected by the building temperature. The outside temperatures' effect on the rooms' temperatures is doubled for the corner rooms, as they share two sides with

[1]SenML IETF rfc: https://datatracker.ietf.org/doc/html/rfc9100



(a) (b)

Figure 2: Simulation diagram. The temperature in each room is point-like (black sphere) and exchanges heat with the neighboring rooms according to Equation (1), illustrated in panel (a). The temperature is also affected by the output of the heater and cooler systems according to Equation (2), illustrated in panel (b).

the outside. The rooms are simulated as point-like temperatures according to

$$T_t^r = T_{t-\Delta t}^r - \kappa\, \Delta t \sum_{n \in \mathcal{N}(r)} \Delta T_{t-\Delta t}^r(n), \tag{1}$$

where T_t^r denotes the temperature of room r at time t, κ the constant of heat transfer between rooms, Δt the simulation step time in seconds, $n \in \mathcal{N}(r)$ are the neighbors n of room r, and $\Delta T_{t-1}^r(n)$ the temperature difference between rooms t and n at time $t-1$. This temperature exchange is illustrated by the arrows in Figure 2a. The values of the constants κ, Δt, and the room heat capacity C can be found in Table 1. The Outside temperature is real data taken from Swedish Meteorologigal and Hydrological Institute (SMHI)[2] with permission.

2.2 Control Model

To regulate the rooms, they each have a heating regulator and cooling regulator, each controlled by two separate PI control loops with different constants, see the values of K_P and K_I in Table 1. The constants differ by orders of magnitude because the heating actuators expects the actuation value in watts, whereas the expects the actuation as a value between 0 and 1, which will be multiplied by the maximum power P_{max}, which is the same for both heating and cooling units. Further, the regulation is set up so

[2]https://www.smhi.se/en/about-smhi

Table 1: Values of simulation constants.

Property	Symbol	Value
Time step size	Δt	$10\,\mathrm{s}$
Room Heat capacity	C	$46 \times 10^3\,\mathrm{J/K}$
Room Heat transfer rate	κ	$10^{-4}\,\mathrm{W/(m \cdot K)}$
Proportional constant heating	K_P^{heat}	$40\,\mathrm{W/K}$
Proportional constant cooling	K_P^{cool}	$0.1\,\mathrm{K}^{-1}$
Integration constant cooling	K_I^{heat}	$1\,\mathrm{W/(s \cdot K)}$
Integration constant heating	K_I^{cool}	$0.01\,(\mathrm{s \cdot K})^{-1}$
Actuator maximum power output	P_{max}	$1500\,\mathrm{W}$

that the heating unit will have zero actuation when the cooling unit has a non-zero actuation, and vice versa. To achive some variability in the data, each rooms will randomly set a new actuation point, uniformly in the span of 280.15 to 310.15 K, at a random time according to a Poisson(3600 $[s]$) distribution from the last change. This means that on average every room sets a target temperature once every hour, but the exact time will be slightly different, introducing more variance in the data. The heat given to a room r by an actuator is governed by the following equation:

$$T_t^r = T_{t-\Delta t}^r + \frac{y_t^{r,\mathrm{cool}} + y_t^{r,\mathrm{heat}}}{C} \tag{2}$$

where $y_t^{r,unit}$ is the power output of the cooling and heating actuators in watts in room r at time t. To note here is that the the cooling actuation $y_t^{r,cool}$ is always negative, simulating a non-realistic heat sink.

2.3 Control Loops

The full control loop is executed at every time step and is illustrated in Figure 2b:

1. Store the previous temperatures T_{t-1}.
2. Generate temperature messages for each unit in each room (green arrows).
3. (a) Controllers requests the current temperature in their corresponding room (green circles).

(b) Sensors respond with the temperature messages.

4. Update control and generate controller messages.

   (a) If it is time, update the random setpoints before updating control.

5. Controllers send the temperature messages to the actuators (cyan and orange circles).

6. Actuators update the actuations y_t (cyan and orange arrows)

7. Room temperatures updated.

The last step combines Equation 1 and Equation 2 into the full temperature update equation:

$$T_t^r = T_{t-\Delta t}^r - \kappa\, \Delta t \left(\sum_{n \in \mathcal{N}(r)} \Delta T_{t-\Delta t}^r(n) - \frac{y_t^{r,\text{cool}} + y_t^{r,\text{heat}}}{C} \right) \tag{3}$$

3 Generated Dataset

The data comes in two **semicolon-separated (;)** csv files, `training.csv` and `test.csv`. The train/test split is not random; training data comes from the first 80% of simulated timesteps, and the test data is the last 20%. There is no specific validation dataset, the validation data should instead be randomly selected from the training data.

The simulation runs for as many time steps as there are outside temperature values available. The original SMHI data only samples once every hour, which we linearly interpolate to get one temperature sample every ten seconds. The data saved at each time step consists of 34 JSON messages (four per room and two temperature readings from the outside), 9 temperature values (one per room and outside), 8 setpoint values, and 8 actuator outputs. The data associated with each of those 34 JSON-messages is stored as a single row in the tables. This means that much data is duplicated, a chose made to make it easier to use the data.

The messages are all JSON-SenML, but the structure is different between the heating and cooling systems, and the actuator and sensor uses different labels. Four example messages can be found in Figure 3, where the messages from each unit represents the same information, encoded by the different colors.

3.1 File structure

Each data file has the following columns: timeline, message, room_name, system, unit, temperature, setpoint, actuation, previous_actuation_1, previous_actuation_2, previous_actuation_3.

timeline (float)

The timeline column is the time elapsed since the start of the simulation in seconds.

message (string)

Generated JSON message.

room_name (string)

Name of room where the data point was generated. One of 9 possible values: OO, A11, A12, A13, A14, A21, A22, A23, or A24.

	Temperature sensor	Controller

Heating system	`[{`
	`"n":"A13_temp_sensor",`
	`"t":60,`
	`"u":"K",`
	`"v":292.34...`
	`}]`
		`[{`
		`"n":"A13_heater",`
		`"t":60,`
		`"u":"W",`
		`"v":0`
		`}]`

Cooling system	`[`
	`{"bn":"temp_sensor","bt":60},`
	`{"u":"Cel","v":19.199...},`
	`{"u":"Lon","v":"2"},`
	`{"u":"Lat","v":"0"}`
	`]`
		`[`
		`{"bn":"cooler","bt":60},`
		`{"u":"/","v":0.254...},`
		`{"u":"Lon","v":"2"},`
		`{"u":"Lat","v":"0"}`
		`]`

Figure 3: Example messages from the simulation. The top row shows messages from the heating system, and the bottom row messages from the cooling system. The left column shows messages from the temperature sensors and the right column messages from the controller. The parts containing the same information are highlighted in red (location) and green (temperature).

system (string)

System, i.e., temperature sensor or controller, where the data point was generated. One of 2 possible values: temp_sensor or control.

unit (string)

Unit, i.e., heater or cooler, where the data point was generated. One of 2 possible values: heater or cooler.

temperature (float)

Temperature setpoint at the time the data point was generated.

actuation (float)

Actuator output at the time the data point was generated.

previous_actuation_1 (float)

Actuator output in the previous timestep.

previous_actuation_2 (float)

Actuator output two timesteps ago.

previous_actuation_3 (float)

Actuator output three timesteps ago.

References

[1] J. Nilsson and F. Sandin, "Semantic interoperability in industry 4.0: Survey of recent developments and outlook," in *2018 IEEE 16th international conference on industrial informatics (INDIN)*. IEEE, 2018, pp. 127–132.

[2] J. Nilsson, J. Delsing, and F. Sandin, "Autoencoder alignment approach to run-time interoperability for system of systems engineering," in *2020 IEEE 24th International Conference on Intelligent Engineering Systems (INES)*. IEEE, 2020, pp. 139–144.

[3] IERC, AC, "IoT semantic interoperability: Research challenges, best practices, recommendations and next steps," *European Commission Information Society and Media, Tech. Rep*, vol. 8, 2013.

[4] A. Geraci, F. Katki, L. McMonegal, B. Meyer, J. Lane, P. Wilson, J. Radatz, M. Yee, H. Porteous, and F. Springsteel, *IEEE standard computer dictionary*. IEEE Press, 1991.

[5] D. Gürdür and F. Asplund, "A systematic review to merge discourses: Interoperability, integration and cyber-physical systems," *Journal of Industrial information integration*, vol. 9, pp. 14–23, 2018.