Examining Feasibility of STL for Enforcing Safe Building Control

Jacob Brown
Software and Information Systems
University of North Carolina
Charlotte
Charlotte, North Carolina, USA
jbrow533@charlotte.edu

Gabe Fierro
Computer Science
Colorado School of Mines
Golden, Colorado, USA
gtfierro@mines.edu

Meera Sridhar
Software and Information Systems
University of North Carolina
Charlotte
Charlotte, North Carolina, USA
msridhar@charlotte.edu

Abstract

This poster explores Signal Temporal Logic (STL) as a lightweight formalism for specifying and monitoring building control requirements over simulated traces. We implement a single zone thermostat simulation with supervisory controls, PI-style heating and cooling, supply air temperature reset, and more. The simulation produces time series traces that are checked using RTAMT against STL specifications covering comfort bounds, occupied recovery, and more. This preliminary prototype demonstrates that STL can express and monitor safety and correctness properties for building control systems from execution traces alone, supporting a longer-term vision of proactive, data-driven verification for opaque IoT deployments.

CCS Concepts

• **Software and its engineering** → **Formal methods**; • **Computer systems organization** → *Embedded and cyber-physical systems.*

ACM Reference Format:
Jacob Brown, Gabe Fierro, and Meera Sridhar. 2026. Examining Feasibility of STL for Enforcing Safe Building Control. In *ACM Sustainability Week 2026 (ACM Sustainability Week Companion '26), June 22–25, 2026, Banff, AB, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3765611.3815354

1 Introduction

IoT deployments in commercial buildings are becoming larger, more heterogeneous, and more distributed. Sensing, actuation, and decision logic are now spread across many devices, often from different vendors and with limited visibility into their internal control behavior. At the same time, building control sequences are already complex enough to admit subtle bugs, unsafe interactions, and degraded performance even when each individual component appears to operate correctly [5]. This makes it difficult to reason about whether a deployment will remain safe as configurations change, devices are updated, and control decisions interact through a shared physical environment.

Most existing approaches for building controls address these issues reactively. Fault detection and diagnostic tools, along with rule-based checks such as ConStrain [2], are valuable for identifying problems after they emerge in operational data. However, they do not directly answer the preventative question: can we establish ahead of time that a controller and its surrounding environment will avoid undesirable states?

Formal methods such as *model checking* [1] offer a complementary perspective. Rather than waiting for faults to manifest, they allow one to represent controller behavior and system assumptions mathematically and check whether safety properties and invariants can be violated.

This poster is a first step in exploring how that perspective can be adapted to opaque IoT control systems in buildings. In real-world settings, we typically do not have white-box access to the control logic embedded inside deployed devices, making source-level analysis or binary instrumentation impractical. We also cannot assume a complete model of the environment in which those devices operate.

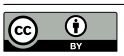
Our long-term goal is to show how learned behavioral models and learned environmental assumptions in tandem with formal verification such as model checking can support proactive verification for real-world building IoT systems. This poster describes our first step: we build a lightweight simulation prototype for a thermostat-driven zone, encode control features such as demand limiting, standby behavior, PI-style response, and supply-air-temperature reset, and then monitor the resulting execution traces against formally-encoded safety and correctness properties.

2 Background

Model checking is a formal verification method for assessing whether a system conforms to predefined safety or security properties. Given a formal model of the system and a property specified in a declarative language, a model checker systematically explores all reachable execution paths permitted by the system's behavior. The outcome is either a proof that the property is satisfied or a counterexample illustrating how the property can be violated.

More recent work has explored the extension of model checking to *cyber-physical systems* (CPS) [3]. Signal Temporal Logic (STL) [4] is an extension of the traditional Linear Temporal Logic (LTL) [1] designed for specifying properties over real-valued signals that change over time. This makes it ideal for building systems, where quantities such as supervisory inputs, thermostat logic, occupancy, weather, and thermal response can be expressed directly as temporal requirements over signals like temperature.

In this implementation we use STL in a simulation based setting. We generate traces from a lightweight thermostat-zone model and then use the RTAMT Python library [6] to evaluate those traces against STL specifications.



This work is licensed under a Creative Commons Attribution 4.0 International License.
ACM Sustainability Week Companion '26, Banff, AB, Canada
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2199-1/26/06
https://doi.org/10.1145/3765611.3815354

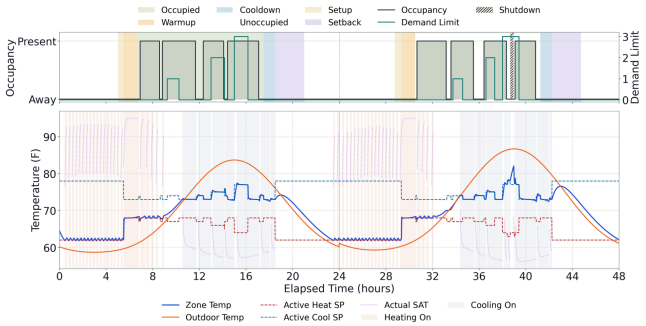


Figure 1: Simulated system trace.

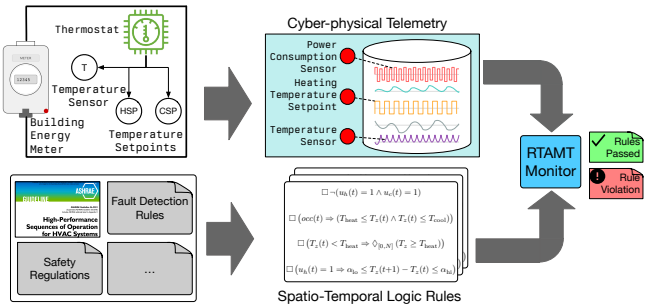


Figure 2: We encode safety and correctness constraints as STL rules which are checked against simulated traces.

3 Technical Approach

As the first step, we build a lightweight simulation of a single thermostat-controlled zone that incorporates mode-based setpoints, cooling resets, supply air temperature, etc. Running the simulation produces a trace of the simulated system state at each step, as seen in Fig. 1. Here, the top part shows the supervisory context: the occupancy line shows true occupancy, the colored bands show the current zone group mode, and the demand limit level. The bottom part shows the thermal response; the x-axis is the elapsed time and the y-axis is the temperature, including the zone temperature, the outdoor temperature, active heating and cooling setpoints, active SAT, and heating/cooling operation as colored bands.

We currently define 12 STL specifications ranging from a simple zone temperature check to more involved bounded time recovery properties. The zone temperature check requires the zone temperature to remain between the active heating and cooling setpoints:

$$heating\_setpoint \leq zone\_temperature \leq cooling\_setpoint$$

In the implementation, we allow a 2 degree Fahrenheit tolerance around this band and encode the condition as a derived signal called the *zone band margin*. We then write the STL monitor as:

$$\mathbf{G}(zone\_band\_margin \geq 0),$$

where $\mathbf{G}$ denotes that the condition must hold globally, or at every time step in the trace.

As an example of a more complex specification, we check *occupied recovery*. During occupied mode, the zone must either already be inside the allowed band or return to that band within 30 minutes:

$$\mathbf{G}\begin{pmatrix} \neg occupied \vee (zone\_band\_margin \geq 0) \\ \vee\, \mathbf{F}_{[1,30]}(zone\_band\_margin \geq 0) \end{pmatrix}$$

where $\mathbf{F}_{[1,30]}$ denotes that the condition must become true sometime within the subsequent 1 to 30 minutes.

Fig. 2 summarizes the end-to-end verification pipeline. Time series inputs provide outdoor temperature and supervisory state. The thermostat-zone model outputs a simulation trace. The STL monitor then evaluates the trace against the 12 specifications and reports pass/fail results with robustness values. In our simulation, all 12 specifications pass with positive robustness margins.

4 Conclusion

This prototype demonstrates that STL can express and monitor real-world safety and correctness properties for building control systems using only execution traces, without requiring white-box access to controller logic. This supports our longer-term goal of proactive, data-driven verification for opaque IoT deployments. Future work will ground the physical model in learned thermal dynamics from real building data, extend to multi-zone environments, and broaden the specification set. Ultimately, we plan to move beyond trace monitoring with RTAMT toward full model checking over learned behavioral models.

Acknowledgments

Work supported in part by University of North Carolina System Research Opportunities Initiative Award; and by the National Alliance for Water Innovation (NAWI), funded by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy (EERE), Industry Technology Office, under DE-FOA-0001905.

References

[1] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. MIT Press.

[2] Yan Chen, Michael Wetter, Xuechen Lei, Jeremy Lerond, Anand K. Prakash, Yun Joon Jung, Paul Ehrlich, and Draguna Vrabie. 2023. Control Performance Verification – The Hidden Opportunity of Ensuring High Performance of Building Control System. In *2023 Building Simulation Conference*. doi:10.26868/25222708.2023.1660

[3] Edward A. Lee. 2015. *Principles of Cyber-Physical Systems*. MIT Press.

[4] Oded Maler and Dejan Nickovic. 2004. Monitoring Temporal Properties of Continuous Signals. In *Formal Modeling and Analysis of Timed Systems (Lecture Notes in Computer Science, Vol. 3253)*. Springer, 152–166.

[5] Michael Wetter, Paul Ehrlich, Antoine Gautier, Milica Grahovac, Philip Haves, Jianjun Hu, Anand Prakash, Dave Robin, and Kun Zhang. 2022. OpenBuildingControl: Digitizing the control delivery from building energy modeling to specification, implementation and formal verification. *Energy* 238 (Jan. 2022), 121501. doi:10.1016/j.energy.2021.121501

[6] Tomoya Yamaguchi, Bardh Hoxha, and Dejan Nickovic. 2024. RTAMT – Runtime Robustness Monitors with Application to CPS and Robotics. *International Journal on Software Tools for Technology Transfer* 26 (2024), 79–99. doi:10.1007/s10009-023-00720-3