From Cold Start to Critical: Formal Synthesis of Autonomous Hybrid Controllers

PI: Dane A. Sabo
dane.sabo@pitt.edu

Advisor: Dr. Daniel G. Cole
dgcole@pitt.edu

Track: PhD Mechanical Engineering

Saturday 14th March, 2026

The goal of this research is to develop a methodology for creating autonomous control systems with event-driven control laws that have guarantees of safe and correct behavior.

Strong, direct opening. Sets scope immediately.

Nuclear power relies on extensively trained operators who follow detailed written procedures to manage reactor control. Based on these procedures and operators' interpretation of plant conditions, operators make critical decisions about when to switch between control objectives.

Consider: "operators" appears 3x in two sentences. Maybe: "Based on these procedures and their interpretation of plant conditions, they make critical decisions..."

But, reliance on human operators has created an economic challenge for next-generation nuclear power plants.

"But, reliance" — the comma after "But" is unusual. Either drop it or restructure: "However, this reliance..." or "This reliance, however, has created..."

Small modular reactors face significantly higher per-megawatt staffing costs than conventional plants. Autonomous control systems are needed that can safely manage complex operational sequences with the same assurance as human-operated systems, but without constant supervision.

"are needed that can" — passive. Try: "Autonomous control systems must safely manage..."

To address this need, we will combine formal methods from computer science with control theory to build hybrid control systems that are correct by construction. Hybrid systems use discrete logic to switch between continuous control modes, similar to how operators change control strategies. Existing formal methods generate provably correct switching logic but cannot handle continuous dynamics during transitions, while traditional control theory verifies continuous behavior but lacks tools for proving discrete switching correctness. We will bridge this gap through a three-stage methodology. First, we will translate written operating procedures into temporal logic specifications using NASA's Formal Requirements Elicitation Tool (FRET), which structures requirements into scope, condition, component, timing, and response elements. This structured approach enables realizability checking to identify conflicts and ambiguities in procedures before implementation. Second, we will synthesize discrete mode switching logic using reactive synthesis to generate deterministic automata that are provably correct by construction. Third, we will develop continuous controllers for each discrete mode using standard control theory and reachability analysis. We will classify continuous modes based on their transition objectives, and then employ assume-guarantee contracts and barrier certificates to prove that mode transitions occur safely and as defined by the deterministic automata. This compositional approach enables local verification of continuous modes without requiring global trajectory analysis across the entire

Clear statement of approach.

Nice parallel structure showing the gap.

hybrid system. We will demonstrate this on an Emerson Ovation control system.

This paragraph is dense. Consider breaking after the three stages, then a new paragraph for the compositional verification point and Emerson demo.

This approach will demonstrate autonomous control can be used for complex nuclear power operations while maintaining safety guarantees.

"can be used for" — weak. Try: "...will demonstrate that autonomous control can manage complex nuclear power operations while maintaining safety guarantees." Or even stronger: "...enables autonomous management of complex nuclear power operations with safety guarantees."

If this research is successful, we will be able to do the following:

1. *Synthesize written procedures into verified control logic.* We will develop a methodology for converting written operating procedures into formal specifications. These specifications will be synthesized into discrete control logic using reactive synthesis tools. Control engineers will be able to generate mode-switching controllers from regulatory procedures with little formal methods expertise, reducing barriers to high-assurance control systems.

   Good practical framing — emphasizes accessibility.

2. *Verify continuous control behavior across mode transitions.* We will develop methods using reachability analysis to ensure continuous control modes satisfy discrete transition requirements. Engineers will be able to design continuous controllers using standard practices while ensuring system correctness and proving mode transitions occur safely at the right times.
3. *Demonstrate autonomous reactor startup control with safety guarantees.* We will implement this methodology on a small modular reactor simulation using industry-standard control hardware. Control engineers will be able to implement high-assurance autonomous controls on industrial platforms they already use, enabling users to achieve autonomy without retraining costs or developing new equipment.

   Strong industrial grounding — the "platforms they already use" point is compelling for adoption.

Contents

1 Goals and Outcomes

The goal of this research is to develop a methodology for creating autonomous hybrid control systems with mathematical guarantees of safe and correct behavior.

Clear thesis statement. Gets right to it.

Nuclear power plants require the highest levels of control system reliability, where failures can result in significant economic losses, service interruptions, or radiological release. Currently, nuclear plant operations rely on extensively trained human operators who follow detailed written procedures and strict regulatory requirements to manage reactor control. These operators make critical decisions about when to switch between different control modes based on their interpretation of plant conditions and procedural guidance. This reliance on human operators prevents autonomous control capabilities and creates a fundamental economic challenge for next-generation reactor designs.

Stakes established immediately — good hook.

The "and" here joins two distinct issues (autonomy barrier + economics). Consider making the causal link explicit: "This reliance on human operators not only prevents autonomous control capabilities but also creates..." or split into two sentences.

Small modular reactors, in particular, face per-megawatt staffing costs far exceeding those of conventional plants and threaten their economic viability.

What is needed is a method to create autonomous control systems that safely manage complex operational sequences with the same assurance as human-operated systems, but without constant human supervision.

"What is needed is" — Gopen would call this a weak topic position. The sentence buries the subject. Try: "Autonomous control systems must safely manage complex operational sequences..." Puts the actor in the topic position.

To address this need, we will combine formal methods with control theory to build hybrid control systems that are correct by construction. Hybrid systems use discrete logic to switch between continuous control modes, mirroring how operators change control strategies. Existing formal methods can generate provably correct switching logic from written requirements, but they cannot handle the continuous dynamics that occur during transitions between modes. Meanwhile, traditional control theory can verify continuous behavior but lacks tools for proving correctness of discrete switching decisions. By synthesizing discrete mode transitions directly from written operating procedures and verifying continuous behavior between transitions, we can create hybrid control systems with end-to-end correctness guarantees. If existing procedures can be formalized into logical specifications and continuous dynamics verified against transition requirements, then autonomous controllers can be built that are provably free from design defects. This approach will enable autonomous control in nuclear power plants while maintaining the high safety standards required by the industry.

Excellent setup of the gap — shows why neither approach alone is sufficient.

Hypothesis is clear and testable.

This work is conducted within the University of Pittsburgh Cyber Energy Center, which provides access to industry collaboration and Emerson control hardware, ensuring that developed solutions align with practical implementation requirements.

This qualifications paragraph feels orphaned here. It's important context but reads as an afterthought. Consider integrating it into the approach paragraph ("...demonstrated on Emerson hardware through our partnership with the Cyber Energy Center") or moving to a "Why This Will Succeed" framing later.

If this research is successful, we will be able to do the following:

1. **Translate written procedures into verified control logic.** We will develop a methodology for converting existing written operating procedures into formal specifications that can be automatically synthesized into discrete control logic. This process will use structured intermediate representations to bridge natural language procedures and mathematical logic. Control system engineers will generate verified mode-switching controllers directly from regulatory procedures without formal methods expertise, lowering the barrier to high-assurance control systems.
2. **Verify continuous control behavior across mode transitions.** We will establish methods for analyzing continuous control modes to ensure they satisfy discrete transition requirements. Using classical control theory for linear systems and reachability analysis for nonlinear dynamics, we will verify that each continuous mode safely reaches its intended transitions. Engineers will design continuous controllers using standard practices while iterating to ensure broader system correctness, proving that mode transitions occur safely and at the correct times.
3. **Demonstrate autonomous reactor startup control with safety guarantees.** We will apply this methodology to develop an autonomous controller for nuclear reactor startup procedures, implementing it on a small modular reactor simulation using industry-standard control hardware. This demonstration will prove correctness across multiple coordinated control modes from cold shutdown through criticality to power operation. We will demonstrate that autonomous hybrid control can be realized in the nuclear industry with current equipment, establishing a path toward reduced operator staffing while maintaining safety.

"cold shutdown through criticality to power operation" — concrete and impressive scope.

The innovation in this work is unifying discrete synthesis with continuous verification to enable end-to-end correctness guarantees for hybrid systems. If successful, control engineers will create autonomous controllers from existing procedures with mathematical proof of correct behavior. High-assurance autonomous control will become practical for safety-critical applications. This capability is essential for the economic viability of next-

Clear "what's new" statement.

generation nuclear power. Small modular reactors offer a promising solution to growing energy demands, but their success depends on reducing per-megawatt operating costs through increased autonomy. This research will provide the tools to achieve that autonomy while maintaining the exceptional safety record the nuclear industry requires.

Strong closing — ties technical work to real-world impact and economic necessity.

2 State of the Art and Limits of Current Practice

The principal aim of this research is to create autonomous reactor control systems that are tractably safe. To understand what is being automated, we must first understand how nuclear reactors are operated today. This section examines reactor operators and the operating procedures we aim to leverage, then investigates limitations of human-based operation, and concludes with current formal methods approaches to reactor control systems.

Good roadmap — tells reader exactly what's coming.

2.1 Current Reactor Procedures and Operation

Nuclear plant procedures exist in a hierarchy: normal operating procedures for routine operations, abnormal operating procedures for off-normal conditions, Emergency Operating Procedures (EOPs) for design-basis accidents, Severe Accident Management Guidelines (SAMGs) for beyond-design-basis events, and Extensive Damage Mitigation Guidelines (EDMGs) for catastrophic damage scenarios. These procedures must comply with 10 CFR 50.34(b)(6)(ii) and are developed using guidance from NUREG-0899 [**?**, **?**], but their development relies fundamentally on expert judgment and simulator validation rather than formal verification. Procedures undergo technical evaluation, simulator validation testing, and biennial review as part of operator requalification under 10 CFR 55.59 [**?**]. Despite this rigor, procedures fundamentally lack formal verification of key safety properties. No mathematical proof exists that procedures cover all possible plant states, that required actions can be completed within available timeframes, or that transitions between procedure sets maintain safety invariants.

This paragraph is doing a lot. Consider splitting: first paragraph on the hierarchy and compliance, second on the lack of formal verification. The "No mathematical proof exists..." sentence is powerful and deserves emphasis.

LIMITATION: *Procedures lack formal verification of correctness and completeness.* Current procedure development relies on expert judgment and simulator validation. No mathematical proof exists that procedures cover all possible plant states, that required actions can be completed within available timeframes, or that transitions between procedure sets maintain safety invariants. Paper-based procedures cannot ensure correct application, and even computer-based procedure systems lack the formal guarantees that automated reasoning could provide.

This repeats the "No mathematical proof exists..." sentence almost verbatim from the paragraph above. Either cut it from the paragraph or from the LIMITATION box.

Nuclear plants operate with multiple control modes: automatic control, where the reactor control system maintains target parameters through continuous reactivity adjustment; manual control, where operators directly manipulate the reactor; and various intermediate modes. In typical pressurized water reactor operation, the reactor control system automatically maintains a floating average temperature and compensates for power demand changes through reactivity feedback loops alone. Safety systems, by contrast, operate with implemented automation. Reactor Protection Systems trip automatically on safety signals with millisecond response times, and engineered safety features actuate automatically on accident signals without operator action required.

The division between automated and human-controlled functions reveals the fundamental challenge of hybrid control. Highly automated systems handle reactor protection—automatic trips on safety parameters, emergency core cooling actuation, containment isolation, and basic process control [?, ?]. Human operators, however, retain control of strategic decision-making: power level changes, startup/shutdown sequences, mode transitions, and procedure implementation.

This is the key insight — the hybrid nature is already there, just not formally verified.

2.2 Human Factors in Nuclear Accidents

Current-generation nuclear power plants employ over 3,600 active NRC-licensed reactor operators in the United States [?]. These operators divide into Reactor Operators (ROs), who manipulate reactor controls, and Senior Reactor Operators (SROs), who direct plant operations and serve as shift supervisors [?]. Staffing typically requires at least two ROs and one SRO for current-generation units [?]. Becoming a reactor operator requires several years of training.

The persistent role of human error in nuclear safety incidents—despite decades of improvements in training and procedures—provides the most compelling motivation for formal automated control with mathematical safety guarantees. Operators hold legal authority under 10 CFR Part 55 to make critical decisions, including departing from normal regulations during emergencies. The Three Mile Island (TMI) accident demonstrated how a combination of personnel error, design deficiencies, and component failures led to partial meltdown when operators misread confusing and contradictory readings and shut off the emergency water system [?]. The President's Commission on TMI identified a fundamental ambiguity: placing responsibility for safe power plant operations on the licensee without formal verification that operators can fulfill this responsibility does not guarantee safety. This tension between operational flexibility and safety assurance remains unresolved: the person responsible for reactor safety is often the root cause of failures.

Strong thesis for this subsection.

"the person responsible for reactor safety is often the root cause of failures" — devastating summary. Very effective.

Multiple independent analyses converge on a striking statistic: 70–80% of nuclear power plant events are attributed to human error, versus approximately 20% to equipment failures [?]. More significantly, the root cause of all severe accidents at nuclear power plants—Three Mile Island, Chernobyl, and Fukushima Daiichi—has been identified as poor safety management and safety culture: primarily human factors [?]. A detailed analysis of 190 events at Chinese nuclear power plants from 2007–2020 [?] found that 53% of events involved active errors, while 92% were associated with latent errors—organizational and systemic weaknesses that create conditions for failure.

Strong empirical grounding. The Chinese plant data is a nice addition — shows this isn't just a Western regulatory perspective.

LIMITATION: *Human factors impose fundamental reliability limits that cannot be overcome through training alone.* The persistent human error contribution despite four decades of improvements demonstrates that these limitations are fundamental rather than a remediable part of human-driven control.

Well-stated. The "four decades" point drives it home.

2.3 Formal Methods

2.3.1 HARDENS

The High Assurance Rigorous Digital Engineering for Nuclear Safety (HARDENS) project represents the most advanced application of formal methods to nuclear reactor control systems to date [**?**].

HARDENS aimed to address a fundamental dilemma: existing U.S. nuclear control rooms rely on analog technologies from the 1950s–60s. This technology is obsolete compared to modern control systems and incurs significant risk and cost. The NRC contracted Galois, a formal methods firm, to demonstrate that Model-Based Systems Engineering and formal methods could design, verify, and implement a complex protection system meeting regulatory criteria at a fraction of typical cost. The project delivered a Reactor Trip System (RTS) implementation with full traceability from NRC Request for Proposals and IEEE standards through formal architecture specifications to verified software.

HARDENS employed formal methods tools and techniques across the verification hierarchy. High-level specifications used Lando, SysMLv2, and FRET (NASA Formal Requirements Elicitation Tool) to capture stakeholder requirements, domain engineering, certification requirements, and safety requirements. Requirements were analyzed for consistency, completeness, and realizability using SAT and SMT solvers. Executable formal models used Cryptol to create a behavioral model of the entire RTS, including all subsystems, components, and limited digital twin models of sensors, actuators, and compute infrastructure. Automatic code synthesis generated verifiable C implementations and SystemVerilog hardware implementations directly from Cryptol models—eliminating the traditional gap between specification and implementation where errors commonly arise.

Good technical depth on HARDENS toolchain.

Despite its accomplishments, HARDENS has a fundamental limitation directly relevant to hybrid control synthesis: the project addressed only discrete digital control logic without modeling or verifying continuous reactor dynamics. The Reactor Trip System specification and verification covered discrete state transitions (trip/no-trip decisions), digital sensor input processing through discrete logic, and discrete actuation outputs (reactor trip commands). The project did not address continuous dynamics of nuclear reactor physics. Real reactor safety depends on the interaction between continuous processes—temperature, pressure, neutron flux—evolving in response to discrete control decisions. HARDENS verified the discrete controller in isolation but not the closed-loop hybrid system behavior.

Clear articulation of the gap your work fills.

LIMITATION: *HARDENS addressed discrete control logic without continuous dynamics or hybrid system verification.* Verifying discrete control logic alone provides no guarantee that the closed-loop system exhibits desired continuous behavior such as stability, convergence to setpoints, or maintained safety margins.

HARDENS produced a demonstrator system at Technology Readiness

Level 2–3 (analytical proof of concept with laboratory breadboard validation) rather than a deployment-ready system validated through extended operational testing. The NRC Final Report explicitly notes [**?**] that all material is considered in development, not a finalized product, and that "The demonstration of its technical soundness was to be at a level consistent with satisfaction of the current regulatory criteria, although with no explicit demonstration of how regulatory requirements are met." The project did not include deployment in actual nuclear facilities, testing with real reactor systems under operational conditions, side-by-side validation with operational analog RTS systems, systematic failure mode testing (radiation effects, electromagnetic interference, temperature extremes), NRC licensing review, or human factors validation with licensed operators in realistic control room scenarios.

LIMITATION: *HARDENS achieved TRL 2–3 without experimental validation.* While formal verification provides mathematical correctness guarantees for the implemented discrete logic, the gap between formal verification and actual system deployment involves myriad practical considerations: integration with legacy systems, long-term reliability under harsh environments, human-system interaction in realistic operational contexts, and regulatory acceptance of formal methods as primary assurance evidence.

2.3.2 Sequent Calculus and Differential Dynamic Logic

There has been additional work to do verification of hybrid systems by extending the temporal logics directly. The result has been the field of differential dynamic logic (dL). dL introduces two additional operators into temporal logic: the box operator and the diamond operator. The box operator $[\alpha]\phi$ states that for some region ϕ, the hybrid system α always remains within that region. In this way, it is a safety ivariant being enforced for the system. The second operator, the diamond operator $<\alpha>\phi$ says that for the region ϕ, there is at least one trajectory of α that enters that region. This is a declaration of a liveness property.

Typo: "ivariant" should be "invariant"

While dL allows for the specification of these liveness and safety properties, actually proving them for a given hybrid system is quite difficult. Automated proof assistants such as KeYmaera X exist to help develop proofs of systems using dL, but so far have been insufficient for reasonably complex hybrid systems. The main issue behind creating system proofs using dL is state space explosion and non-terminating solutions. Approaches have been made to alleviate these issues for nuclear power contexts using contract and decomposition based methods, but are far from a complete methodology to design systems with. Instead, these approaches have been used on systems that have been designed a priori, and require expert knowledge to create the system proofs.

Consider adding a concrete example here — "For instance, a system with N modes and M continuous state variables..." to give readers a sense of the scaling problem.

"but are far from a complete methodology to design systems with" — awkward ending preposition. Try: "but remain far from a complete design methodology" or "but do not yet constitute a com-

Your comment here is spot-on. You should add a LIMITATION box: *Differential dynamic logic has been used for post-hoc analysis of existing systems, not for the constructive design of autonomous controllers.* This is exactly the gap you're filling — you're doing synthesis, not just verification.

3 Research Approach

Previous approaches to autonomous control have verified discrete switching logic or continuous control behavior, but not both simultaneously. Validation of continuous controllers today consists of extensive simulation trials. Discrete switching logic for routine operation has been driven by human operators, whose evaluation includes simulated control room testing and human factors research. Neither method, despite being extremely resource intensive, provides rigorous guarantees of control system behavior. HAHACS bridges this gap by composing formal methods from computer science with control-theoretic verification, formalizing reactor operations using the framework of hybrid automata.

The challenge of hybrid system verification lies in the interaction between discrete and continuous dynamics. Discrete transitions change the governing vector field, creating discontinuities in the system's behavior. Traditional verification techniques designed for purely discrete or purely continuous systems cannot handle this interaction directly. Our methodology addresses this challenge through decomposition. We verify discrete switching logic and continuous mode behavior separately, then compose these guarantees to reason about the complete hybrid system. This two-layer approach mirrors the structure of reactor operations themselves: discrete supervisory logic determines which control mode is active, while continuous controllers govern plant behavior within each mode.

Missing space before "Our

Compositional verification claim needs citation. See assume-guarantee literature (Henzinger, Alur). None of the NEEDS_REVIEWED papers directly prove this composition is sound for your specific approach.

To build a high-assurance hybrid autonomous control system (HAHACS), we must first establish a mathematical description of the system. This work draws on automata theory, temporal logic, and control theory. A hybrid system is a dynamical system that has both continuous and discrete states. The specific type of system discussed in this proposal is a continuous autonomous hybrid system. This means that the system does not have external input and that continuous states do not change instantaneously when discrete states change. For our systems of interest, the continuous states are physical quantities that are always Lipschitz continuous. This nomenclature is borrowed from the Handbook on Hybrid Systems Control [**?**], but is redefined here for convenience:

$$H = (\mathcal{Q}, \mathcal{X}, \mathbf{f}, Init, \mathcal{G}, \delta, \mathcal{R}, Inv) \tag{1}$$

where:

- $\mathcal{Q}$: the set of discrete states (modes) of the system
- $\mathcal{X} \subseteq \mathbb{R}^n$: the continuous state space
- $\mathbf{f} : \mathcal{Q} \times \mathcal{X} \to \mathbb{R}^n$: vector fields defining the continuous dynamics for each discrete mode q_i
- $Init \subseteq \mathcal{Q} \times \mathcal{X}$: the set of initial states
- $\mathcal{G}$: guard conditions that define when discrete state transitions may occur

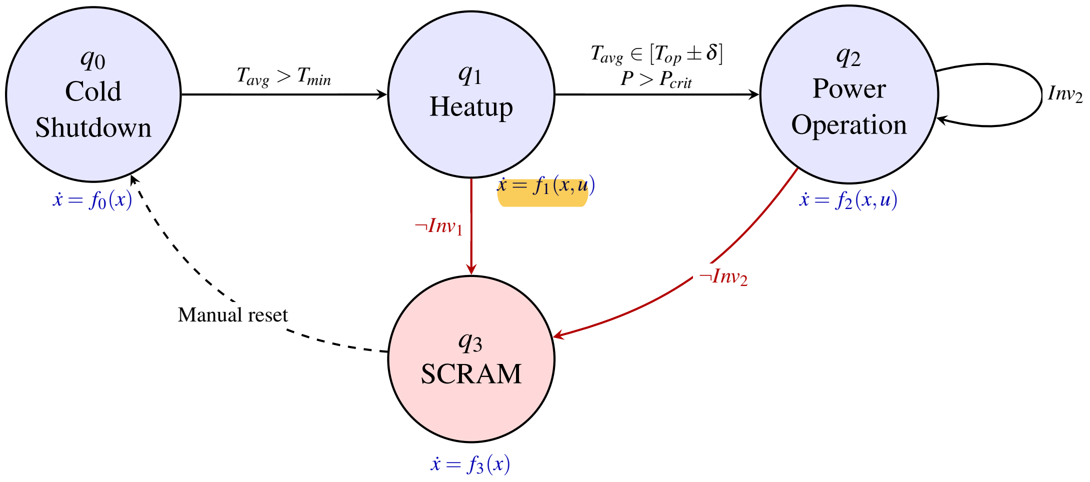

Figure 1: Simplified hybrid automaton for reactor startup. Each discrete state q_i has associated continuous dynamics f_i. Guard conditions on transitions (e.g., $T_{avg} > T_{min}$) are predicates over continuous state. Invariant violations ($\neg Inv_i$) trigger transitions to the SCRAM state. The operational level manages discrete transitions; the tactical level executes continuous control within each mode.

- $\delta : \mathscr{Q} \times \mathscr{G} \to \mathscr{Q}$: the discrete state transition function
- $\mathscr{R}$: reset maps that define any instantaneous changes to continuous state upon discrete transitions
- Inv: safety invariants on the continuous dynamics

The creation of a HAHACS amounts to the construction of such a tuple together with proof artifacts demonstrating that the intended behavior of the control system is satisfied by its actual implementation. This approach is tractable now because the infrastructure for each component has matured. The novelty is not in the individual pieces, but in the architecture that connects them. By defining entry, exit, and safety conditions at the discrete level first, we transform the intractable problem of global hybrid verification into a collection of local verification problems with clear interfaces. Verification is performed per mode rather than on the full hybrid system, keeping the analysis tractable even for complex reactor operations.

This is your key insight — the novelty is compositional, not component-level.

3.1 System Requirements, Specifications, and Discrete Controllers

Human control of nuclear power can be divided into three different scopes: strategic, operational, and tactical. Strategic control is high-level and long-term decision making for the plant. This level has objectives that are complex and economic in scale, such as managing labor needs and supply chains to optimize scheduled maintenance and downtime. The time scale at this level is long, often spanning months or years. The lowest level of control is the tactical level. This is the individual control of pumps, turbines, and

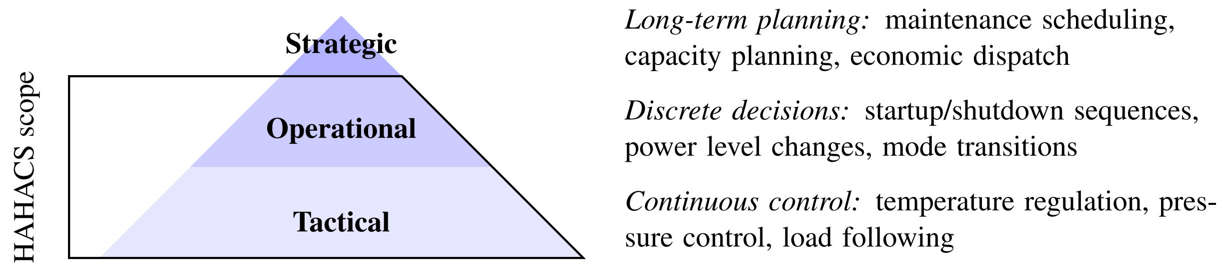
Strategic
Operational
Tactical
HAHACS scope
Long-term planning: maintenance scheduling, capacity planning, economic dispatch
Discrete decisions: startup/shutdown sequences, power level changes, mode transitions
Continuous control: temperature regulation, pressure control, load following

Figure 2: Control scope hierarchy in nuclear power operations. Strategic control (long-term planning) remains with human management. HAHACS addresses the operational level (discrete mode switching) and tactical level (continuous control within modes), which together form a hybrid control system.

chemistry. Tactical control has already been somewhat automated in nuclear power plants today, and is generally considered "automatic control" when autonomous. These controls are almost always continuous systems with a direct impact on the physical state of the plant. Tactical control objectives include maintaining pressurizer level, maintaining core temperature, or adjusting reactivity with a chemical shim.

The level of control linking these two extremes is the operational control scope. Operational control is the primary responsibility of human operators today. Operational control takes the current strategic objective and implements tactical control objectives to drive the plant towards strategic goals. In this way, it bridges high-level and low-level goals. A strategic goal may be to perform refueling at a certain time, while the tactical level of the plant is currently focused on maintaining a certain core temperature. The operational level issues the shutdown procedure, using several smaller tactical goals along the way to achieve this objective. Thus, the combination of the operational and tactical levels fundamentally forms a hybrid controller. The tactical level is the continuous evolution of the plant according to the control input and control law, while the operational level is a discrete state evolution that determines which tactical control law to apply.

This operational control level is the main reason for the requirement of human operators in nuclear control today. The hybrid nature of this control system makes it difficult to prove that a controller will perform according to strategic requirements, as unified infrastructure for building and verifying hybrid systems does not currently exist. Humans have been used for this layer because their general intelligence has been relied upon as a safe way to manage the hybrid nature of this system. But these operators use prescriptive operating manuals to perform their control with strict procedures on what control to implement at a given time. These procedures are the key to the operational control scope.

The method of constructing a HAHACS in this proposal leverages two key observations about current practice. First, the operational scope control is effectively discrete control. Second, the rules for implementing this control are described prior to their implementation in operating procedures. Before constructing a HAHACS, we must completely describe its intended behavior. The behavior of any control system originates in requirements: statements about what the system must do, must not do, and under what conditions. For nuclear systems, these requirements derive from multiple sources including regulatory mandates, design basis analyses, and operating procedures. The challenge is formalizing these requirements with sufficient precision that they can serve as the foundation for autonomous control system synthesis and verification. We can build these requirements using temporal logic.

Temporal logic is a powerful set of semantics for building systems with complex but deterministic behavior. Temporal logic extends classical propositional logic with operators that express properties over time. Using temporal logic, we can make statements relating discrete control modes to one another and define all the requirements of a HAHACS. The guard conditions $\mathscr{G}$ are defined by determining boundary conditions between discrete states and specifying their behavior, while continuous mode invariants can also be expressed as temporal logic statements. These specifications form the basis of any proofs about a HAHACS and constitute the fundamental truth statements about what the behavior of the system is designed to be.

Discrete mode transitions include predicates that are Boolean functions over the continuous state space: $p_i : \mathscr{X} \to \{\text{true}, \text{false}\}$. These predicates formalize conditions like "coolant temperature exceeds 315°C" or "pressurizer level is between 30% and 60%." Critically, we do not impose this discrete abstraction artificially. Operating procedures for nuclear systems already define go/no-go conditions as discrete predicates. These thresholds come from design basis safety analysis and have been validated over decades of operational experience. Our methodology assumes this domain knowledge exists and provides a framework to formalize it. This is why the approach is feasible for nuclear applications specifically: the hard work of defining safe operating boundaries has already been done by generations of nuclear engineers.

Linear temporal logic (LTL) is particularly well-suited for specifying reactive systems. LTL formulas are built from atomic propositions (our discrete predicates) using Boolean connectives and temporal operators. The key temporal operators are:

- $\mathbf{X}\phi$ (next): ϕ holds in the next state
- $\mathbf{G}\phi$ (globally): ϕ holds in all future states
- $\mathbf{F}\phi$ (finally): ϕ holds in some future state
- $\phi \mathbf{U} \psi$ (until): ϕ holds until ψ becomes true

These operators allow us to express safety properties ("the reactor never enters an unsafe configuration”), liveness properties (“the system eventually reaches operating temperature”), and response properties (“if coolant pressure drops, the system initiates shutdown within bounded time”).

CAUTION: Katis 2022 (Table 1, p.2) notes FRET realizability checking does NOT support liveness properties. Your “eventually reaches operating temperature” example may need alternative verification approach.

To build these temporal logic statements, an intermediary tool called FRET is planned to be used. FRET stands for Formal Requirements Elicitation Tool, and was developed by NASA to build high-assurance timed systems. FRET is an intermediate language between temporal logic and natural language that allows for rigid definitions of temporal behavior while using a syntax accessible to engineers without formal methods expertise. This benefit is crucial for the feasibility of this methodology in industry. By reducing the expert knowledge required to use these tools, their adoption with the current workforce becomes easier.

A key feature of FRET is the ability to start with logically imprecise statements and consecutively refine them into well-posed specifications. We can use this to our advantage by directly importing operating procedures and design requirements into FRET in natural language, then iteratively refining them into specifications for a HAHACS. This has two distinct benefits. First, it allows us to draw a direct link from design documentation to digital system implementation. Second, it clearly demonstrates where natural language documents are insufficient. These procedures may still be used by human operators, so any room for interpretation is a weakness that must be addressed.

FRET has been validated: Katis 2022 (pp.1-2, Section 0.3) demonstrates FRET’s FRETish template system with 160 distinct patterns; Pressburger 2023 (pp.17, Section 1) shows successful application to Lift+Cruise case study with 53 requirements formalized and iteratively refined—strong evidence your approach is feasible.

(Some examples of where FRET has been used and why it will be successful here)

Once system requirements are defined as temporal logic specifications, we use them to build the discrete control system. To do this, reactive synthesis tools are employed. Reactive synthesis is a field in computer science that deals with the automated creation of reactive programs from temporal logic specifications. A reactive program is one that, for a given state, takes an input and produces an output. Our systems fit exactly this mold: the current discrete state and status of guard conditions are the input, while the output is the next discrete state.

Reactive synthesis solves the following problem: given an LTL formula φ that specifies desired system behavior, automatically construct a finite-state machine (strategy) that produces outputs in response to environment inputs such that all resulting execution traces satisfy φ. If such a strategy exists, the specification is called *realizable*. The synthesis algorithm either produces a correct-by-construction controller or reports that no such controller can exist. This realizability check is itself valuable: an unrealizable specification indicates conflicting or impossible requirements in the original procedures.

Realizability is proven valuable: Katis 2022 (pp.7-10) shows FRET diagnosis found 8 minimal unrealizable cores in infusion pump case; Pressburger 2023 (pp.19-21) shows unrealizability revealed under-specification (missing stay requirements in LPC aircraft), driving

The main advantage of reactive synthesis is that at no point in the production of the discrete automaton is human engineering of the implementation required. The resultant automaton is correct by construction. This method of construction eliminates the possibility of human error at the implementation stage entirely. Instead, the effort on the human designer is directed at the specification of system behavior itself. This has two critical implications. First, it makes the creation of the discrete controller tractable. The reasons the controller changes between modes can be traced back to the specification and thus to any requirements, which provides a trace for liability and justification of system behavior. Second, discrete control decisions made by humans are reliant on the human operator operating correctly. Humans are intrinsically probabilistic creatures who cannot eliminate human error. By defining the behavior of this system using temporal logics and synthesizing the controller using deterministic algorithms, we are assured that strategic decisions will always be made according to operating procedures.

(Talk about how one would go from a discrete automaton to actual code)

(Examples of reactive synthesis in the wild)

3.2 Continuous Control Modes

The synthesis of the discrete operational controller is only half of an autonomous controller. These control systems are hybrid, with both discrete and continuous components. This section describes the continuous control modes that execute within each discrete state, and how we verify that they satisfy the requirements imposed by the discrete layer. It is important to clarify the scope of this methodology with respect to continuous controller design. This work verifies continuous controllers; it does not synthesize them. The distinction parallels model checking in software verification: model checking does not tell engineers how to write correct software, but it verifies whether a given implementation satisfies its specification. Similarly, we assume that continuous controllers can be designed using standard control theory techniques. Our contribution is a verification framework that confirms candidate controllers compose correctly with the discrete layer to produce a safe hybrid system.

The operational control scope defines go/no-go decisions that determine what kind of continuous control to implement. The entry or exit conditions of a discrete state are themselves the guard conditions $\mathscr{G}$ that define the boundaries for each continuous controller's allowed state-space region. These continuous controllers all share a common state space, but each individual continuous control mode operates within its own partition defined by the discrete state q_i and the associated guards. This partitioning of the continuous state space among several discrete vector fields has traditionally been a difficult problem for validation and verification. The discontinuity of the vector fields at discrete state interfaces makes reachability analysis computationally expensive, and analytic solutions often become intractable [**?**].

We circumvent these issues by designing our hybrid system from the bottom up with verification in mind. Each continuous control mode has an input set and output set clearly defined by our discrete transitions *a priori*. Consider that we define the continuous state space as $\mathscr{X}$. Each discrete mode q_i then provides three key pieces of information for continuous con-

Strix (Luttenberger 2020, pp.1-3) is a practical reactive synthesis tool winning SYNTCOMP competitions; handles LTL specs for systems with large state spaces. Strix uses parity games and forward-explorative construction (pp.7-8) to scale—recommend as your synthesis backend for nuclear procedures.

Consider discussing scalability: Strix handles large alphabets better (v19.07 update, p.30), but still struggles with very large specifications. Document expected spec size for SmAHTR startup procedures to set expectations.

GR(1) fragment (Maoz & Ringert 2015, pp.1-4) is tractable LTL subset for synthesis: wins SYNTCOMP competitions (p.13). Luttenberger 2020 (Strix tool, pp.1-3) handles full LTL via parity games, achieving 4000+ state specs efficiently (p.5). Your nuclear procedures should fit GR(1) since they're reactive (environment inputs = plant state, outputs = mode transitions). This suggests synthesis will be practical for SmAHTR scale.

Need to verify your LTL specs fit GR(1) or full LTL needed—if full LTL retroller design:

1. **Entry conditions:** $\mathscr{X}_{entry,i} \subseteq \mathscr{X}$, the set of possible initial states when entering this mode
2. **Exit conditions:** $\mathscr{X}_{exit,i} \subseteq \mathscr{X}$, the target states that trigger transition to the next mode, or is the region in the state space a stabilizing mode remains within.
3. **Safety invariants:** $\mathscr{X}_{safe,i} \subseteq \mathscr{X}$, the envelope of safe states during operation in this mode. These are derived from invariants *Inv*.

These sets come directly from the discrete controller synthesis and define precise objectives for continuous control. The continuous controller for mode q_i must drive the system from any state in $\mathscr{X}_{entry,i}$ to some state in $\mathscr{X}_{exit,i}$ while remaining within $\mathscr{X}_{safe,i}$.

We classify continuous controllers into three types based on their objectives: transitory, stabilizing, and expulsory. Each type has distinct verification requirements that determine which formal methods tools are appropriate.

This compositional approach is formalized in Kapuria 2025 (pp.17-24, Section 2.4): component proofs via differential cuts reduce state-space (DC rule, p.20), then system proof composes via differential invariants (DI rule, pp.22-24). Kapuria proves SmAHTR safety by verifying 6 components in isolation then system—your three-mode structure maps perfectly to this decomposition, reducing verification complexity from curse of dimensionality.

This three-mode taxonomy is elegant — maps verification tools to control objectives cleanly.

3.2.1 Transitory Modes

Transitory modes are continuous controllers designed to move the plant from one discrete operating condition to another. Their purpose is to execute transitions: starting from entry conditions, reach exit conditions, and maintain safety invariants throughout. Examples include power ramp-up sequences, cooldown procedures, and load-following maneuvers.

The control objective for a transitory mode can be stated formally. Given entry conditions $\mathscr{X}_{entry}$, exit conditions $\mathscr{X}_{exit}$, safety invariant $\mathscr{X}_{safe}$, and closed-loop dynamics $\dot{x} = f(x, u(x))$, the controller must satisfy:

$$\forall x_0 \in \mathscr{X}_{entry} : \exists T > 0 : x(T) \in \mathscr{X}_{exit} \wedge \forall t \in [0, T] : x(t) \in \mathscr{X}_{safe}$$

That is, from any valid entry state, the trajectory must eventually reach the exit condition without ever leaving the safe region.

Verification of transitory modes uses reachability analysis. Reachability analysis computes the set of all states reachable from a given initial set under the system dynamics. For a transitory mode to be valid, the reachable set from $\mathscr{X}_{entry}$ must satisfy two conditions:

1. The reachable set eventually intersects $\mathscr{X}_{exit}$ (the mode achieves its objective)
2. The reachable set never leaves $\mathscr{X}_{safe}$ (safety is maintained throughout the transition)

Formally, if $\text{Reach}(\mathscr{X}_{entry}, f, [0, T])$ denotes the states reachable within time horizon T:

$$\text{Reach}(\mathscr{X}_{entry}, f_i, [0, T]) \subseteq \mathscr{X}_{safe} \wedge \text{Reach}(\mathscr{X}_{entry}, f_i, [0, T]) \cap \mathscr{X}_{exit} \neq \emptyset$$

Because the discrete controller defines clear boundaries in continuous state space, the verification problem for each transitory mode is well-posed. We know the possible initial conditions, we know the target conditions, and we know the safety envelope. The verification task is to confirm that the candidate continuous controller achieves the objective from all possible starting points.

Several tools exist for computing reachable sets of hybrid systems, including CORA, Flow*, SpaceEx, and JuliaReach. The choice of tool depends on the structure of the continuous dynamics. Linear systems admit efficient polyhedral or ellipsoidal reachability computations. Nonlinear systems require more conservative over-approximations using techniques such as Taylor models or polynomial zonotopes. For this work, we will select tools appropriate to the fidelity of the reactor models available.

Your toolset is well-justified: SpaceEx (Frehse 2011, pp.3-6) handles hybrid automata via support functions; Flow* (Chen 2013) uses Taylor models for nonlinear dynamics; JuliaReach (Bogomolov 2019, pp.1-2) offers flexible set representations (zonotopes, boxes). Kapuria 2025 (pp.11-12, Section 2.2) uses Flow* successfully for SmAHTR reachability with reactor models showing state-space constraints (e.g., temp 673–677°C, Figures 6, 16–20). This validates your tool choices for nuclear systems.

3.2.2 Stabilizing Modes

Stabilizing modes are continuous controllers with an objective of maintaining a particular discrete state indefinitely. Rather than driving the system toward an exit condition, they keep the system within a safe operating region. Examples include steady-state power operation, hot standby, and load-following at constant power level. Reachability analysis for stabilizing modes may not be a suitable approach to validation. Instead, we plan to use barrier certificates. Barrier certificates analyze the dynamics of the system to determine whether flux across a given boundary exists. They evaluate whether any trajectory leaves a given boundary. This definition is exactly what defines the validity of a stabilizing continuous control mode.

A barrier certificate (or control barrier function) is a scalar function $B : \mathscr{X} \to \mathbb{R}$ that certifies forward invariance of a safe set. The idea is analogous to Lyapunov functions for stability: rather than computing trajectories explicitly, we find a certificate function whose properties guarantee the desired behavior. For a safe set $\mathscr{C} = \{x : B(x) \geq 0\}$ and dynamics $\dot{x} = f(x,u)$, the barrier certificate condition requires:

$$\forall x \in \partial\mathscr{C} : \dot{B}(x) = \nabla B(x) \cdot f(x, u(x)) \geq 0$$

This condition states that on the boundary of the safe set (where $B(x) = 0$), the time derivative of B is non-negative. Geometrically, this means the vector field points inward or tangent to the boundary, never outward. If this condition holds, no trajectory starting inside $\mathscr{C}$ can ever leave.

Critical finding from Kapuria 2025: decomposition-based verification (pp.17-24, Section 2.4) proves component safety in isolation using reachability, THEN composes to system proof via differential invariants—your three-mode taxonomy maps cleanly to component verification, reducing complexity from monolithic analysis.

Because the design of the discrete controller defines careful boundaries in continuous state space, the barrier is known prior to designing the continuous controller. This eliminates the search for an appropriate barrier and minimizes complication in validating stabilizing continuous control modes. The discrete specifications tell us what region must be invariant; the barrier certificate confirms that the candidate controller achieves this invariance.

Finding barrier certificates can be formulated as a sum-of-squares (SOS) optimization problem for polynomial systems, or solved using satisfiability modulo theories (SMT) solvers for broader classes of dynamics. The key advantage is that the verification is independent of how the controller was designed. Standard control techniques can be used to build continuous controllers, and barrier certificates provide a separate check that the result satisfies the required invariants. This also allows for the checking of control modes with different models than they are designed for. For example, a lower fidelity model can be used for controller design, but a higher fidelity model can be used for the actual validation of that stabilizing controller.

SOS methods proven effective: Papachristodoulou 2021 (SOSTOOLS v4, pp.1-2) solves barrier certificate optimization via SOS constraints—tool integrates with MATLAB. Borrmann 2015 (pp.4-8) demonstrates control barrier certificates for multi-agent systems, showing how discrete boundaries (mode guards) can inform barrier design. Your claim that discrete specs eliminate barrier search is novel and well-supported by these foundations.

3.2.3 Expulsory Modes

Expulsory modes are continuous controllers responsible for ensuring safety when failures occur. They are designed for robustness rather than optimality. The control objective is to drive the plant to a safe shutdown state from potentially anywhere in the state space, under degraded or uncertain dynamics. Examples include emergency core cooling, reactor SCRAM sequences, and controlled depressurization procedures.

We can detect that physical failures exist because our physical controllers have been previously proven correct by reachability and barrier certificates. We know our controller cannot be incorrect for the nominal plant model, so if an invariant is violated, we know the plant dynamics have changed. The HAHACS can identify that a fault occurred because a discrete boundary condition was violated by the continuous physical controller. This is a direct consequence of having verified the nominal continuous control modes: unexpected behavior implies off-nominal conditions.

Hauswirth 2024 (pp.1-3) shows optimization-based robust feedback controllers can serve as alternative verification method—suggests barrier certificates + reachability provide complementary guarantees for your stabilizing modes.

The mathematical formulation for expulsory mode verification differs from transitory modes in two key ways. First, the entry conditions may be the entire state space (or a large, conservatively bounded region) rather than a well-defined entry set. The failure may occur at any point during operation. Second, the dynamics include parametric uncertainty representing failure modes:

$$\dot{x} = f(x,u,\theta), \quad \theta \in \Theta_{failure}$$

where $\Theta_{failure}$ captures the range of possible degraded plant behaviors identified through failure mode and effects analysis (FMEA) or traditional safety analysis.

GAP: None of the NEEDS_REVIEWED papers directly address reachability with parametric uncertainty for failure mode analysis. SpaceEx handles nondeterministic inputs (Frehse 2011, p.4) but not parametric plant uncertainty. Consider citing CORA (parametric reachability) or robust CBF literature. This may require additional references beyond current collection.

We verify expulsory modes using reachability analysis with parametric uncertainty. The verification condition requires that for all parameter values within the uncertainty set, trajectories from the expanded entry region reach the safe shutdown state:

$$\forall \theta \in \Theta_{failure} : \text{Reach}(\mathcal{X}_{current}, f_\theta, [0,T]) \subseteq \mathcal{X}_{shutdown}$$

This is more conservative than nominal reachability, accounting for the fact that we cannot know exactly which failure mode is active.

Traditional safety analysis techniques inform the construction of $\Theta_{failure}$.

Probabilistic risk assessment, FMEA, and design basis accident analysis identify credible failure scenarios and their effects on plant dynamics. The expulsory mode must handle the worst-case dynamics within this envelope. This is where conservative controller design is appropriate as safety margins will matter more than performance during emergency shutdown.

Parametric uncertainty approach validated: Kapuria 2025 (pp.82-120, Sections 5) verifies SmAHTR resiliency against UCAs with uncertain dynamics (e.g., PHX secondary flow shutdown, resonating turbine flow). Uses reachability + Z3 SMT solver (pp.23-24, Section 2.5 on δ-SAT) to handle nonlinear uncertainty—demonstrates your expulsory mode approach is sound for nuclear failures. Shows safety can be proven even when controller deviates from nominal (pp.85-107, UCA 1 analysis).

3.3 Industrial Implementation

The methodology described above must be validated on realistic systems using industrial-grade hardware to demonstrate practical feasibility. This research will leverage the University of Pittsburgh Cyber Energy Center's partnership with Emerson to implement and test the HAHACS methodology on production control equipment. Emerson's Ovation distributed control system is widely deployed in power generation facilities, including nuclear plants. The Ovation platform provides a realistic target for demonstrating that formally synthesized controllers can execute on industrial hardware meeting timing and reliability requirements. The discrete automaton produced by reactive synthesis will be compiled to run on Ovation controllers, with verification that the implemented behavior matches the synthesized specification exactly.

For the continuous dynamics, we will use a small modular reactor simulation. The SmAHTR (Small modular Advanced High Temperature Reactor) model provides a relevant testbed for startup and shutdown procedures. The ARCADE (Advanced Reactor Control Architecture Development Environment) interface will establish communication between the Emerson Ovation hardware and the reactor simulation, enabling hardware-in-the-loop testing of the complete hybrid controller.

Kapuria 2025 reveals practical challenge: determining Θ_{failure} bounds is non-trivial. Recommend documenting failure mode selection process (FMEA $\rightarrow$ parametric bounds) to make expulsory mode design repeatable for other reactor sequences.

Working with Emerson on such an implementation is an incredible advantage for the success and impact of this work. We will directly address the gap of verification and validation methods for these systems and industry adoption by forming a two-way exchange of knowledge between the laboratory and commercial environments. This work stands to be successful with Emerson implementation because we will have access to system experts at Emerson to help with the fine details of using the Ovation system. At the same time, we will have the benefit of transferring technology directly to industry with a direct collaboration in this research, while getting an excellent perspective of how our research outcomes can align best with customer needs.

Kapuria 2025 validates hybrid control on SmAHTR: formal verification (d$\mathscr{L}$ + reachability, pp.37-70) proved safe PHX maintenance scenario, then Simulink demo confirmed (pp.70-72). This two-tier approach (formal proof + simulation validation) strengthens your Emerson demo plan for credibility.

Consider documenting integration points: ARCADE

4 Metrics for Success

This research will be measured by advancement through Technology Readiness Levels, progressing from fundamental concepts to validated prototype demonstration. This work begins at TRL 2–3 and aims to reach TRL 5, where system components operate successfully in a relevant laboratory environment. This section explains why TRL advancement provides the most appropriate success metric and defines the specific criteria required to achieve TRL 5.

TRL as primary metric is smart — speaks industry language.

Technology Readiness Levels provide the ideal success metric because they explicitly measure the gap between academic proof-of-concept and practical deployment—precisely what this work aims to bridge. Academic metrics like papers published or theorems proved cannot capture practical feasibility. Empirical metrics like simulation accuracy or computational speed cannot demonstrate theoretical rigor. TRLs measure both dimensions simultaneously. Advancing from TRL 3 to TRL 5 requires maintaining theoretical rigor while progressively demonstrating practical feasibility. Formal verification must remain valid as the system moves from individual components to integrated hardware testing.

Good framing — explains why other metrics are insufficient.

The nuclear industry requires extremely high assurance before deploying new control technologies. Demonstrating theoretical correctness alone is insufficient for adoption; conversely, showing empirical performance without formal guarantees fails to meet regulatory requirements. TRLs capture this dual requirement naturally. Each level represents both increased practical maturity and sustained theoretical validity. Furthermore, TRL assessment forces explicit identification of remaining barriers to deployment. The nuclear industry already uses TRLs for technology assessment, making this metric directly relevant to potential adopters. Reaching TRL 5 provides a clear answer to industry questions about feasibility and maturity that academic publications alone cannot.

Moving from current state to target requires achieving three intermediate levels, each representing a distinct validation milestone:

TRL 3 ***Critical Function and Proof of Concept*** For this research, TRL 3 means demonstrating that each component of the methodology works in isolation. Startup procedures must be translated into temporal logic specifications that pass realizability analysis. A discrete automaton must be synthesized with interpretable structure. At least one continuous controller must be designed with reachability analysis proving transition requirements are satisfied. Independent review must confirm that specifications match intended procedural behavior. This proves the fundamental approach on a simplified startup sequence.

TRL 4 ***Laboratory Testing of Integrated Components*** For this research, TRL 4 means demonstrating a complete integrated hybrid controller in simulation. All startup procedures must be formalized with a synthesized automaton covering all operational modes. Continuous controllers must exist for all discrete modes. Verification must be complete for all mode transitions using reachability analysis, barrier certificates, and assume-guarantee contracts. The integrated controller must execute complete startup sequences in software simulation with zero safety violations across multiple consecutive runs. This proves that formal correctness guarantees can be maintained throughout system integration.

TRL 5 ***Laboratory Testing in Relevant Environment*** For this research, TRL 5 means demonstrating the verified controller on industrial control hardware through hardware-in-the-loop testing. The discrete automaton must be implemented on the Emerson Ovation control system and verified to match synthesized specifications exactly. Continuous controllers must execute at required rates. The ARCADE interface must establish stable real-time communication between the Emerson Ovation hardware and SmAHTR simulation. Complete autonomous startup sequences must execute via hardware-in-the-loop across the full operational envelope. The controller must handle off-nominal scenarios to validate that expulsory modes function correctly. For example, simulated sensor failures must trigger appropriate fault detection and mode transitions, and loss-of-cooling scenarios must activate SCRAM procedures as specified. Graded responses to minor disturbances are outside this work's scope. Formal verification results must remain valid, with discrete behavior matching specifications and continuous trajectories remaining within verified bounds. This proves that the methodology produces verified controllers implementable on industrial hardware.

Consider noting why graded responses are out of scope — is it time, complexity, or scope creep? Brief justification helps.

Progress will be assessed quarterly through collection of specific data comparing actual results against TRL advancement criteria. Specification development status indicates progress toward TRL 3. Synthesis results and verification coverage indicate progress toward TRL 4. Simulation performance metrics and hardware integration milestones indicate progress toward TRL 5. The research plan will be revised only when new data invalidates fundamental assumptions. This research succeeds if it achieves TRL 5 by demonstrating a complete autonomous hybrid controller with formal correctness guarantees operating on industrial control hardware through hardware-in-the-loop testing in a relevant laboratory environment. This establishes both theoretical validity and practical feasibility, proving that the methodology produces verified controllers and that implementation is achievable with current technology.

Clear success criteria. Committee will know exactly what "done" looks like.

5 Risks and Contingencies

This research relies on several critical assumptions that, if invalidated, would require scope adjustment or methodological revision. The primary risks to successful completion fall into four categories: computational tractability of synthesis and verification, complexity of the discrete-continuous interface, completeness of procedure formalization, and hardware-in-the-loop integration challenges. Each risk has associated indicators for early detection and contingency plans that preserve research value even if core assumptions prove false. The staged project structure ensures that partial success yields publishable results and clear identification of remaining barriers to deployment.

Honest acknowledgment of risks with clear contingencies — committee will appreciate this.

5.1 Computational Tractability of Synthesis

The first major assumption is that formalized startup procedures will yield automata small enough for efficient synthesis and verification. Reactive synthesis scales exponentially with specification complexity, creating risk that temporal logic specifications derived from complete startup procedures may produce automata with thousands of states. Such large automata would require synthesis times exceeding days or weeks, preventing demonstration of the complete methodology within project timelines. Reachability analysis for continuous modes with high-dimensional state spaces may similarly prove computationally intractable. Either barrier would constitute a fundamental obstacle to achieving the research objectives.

Several indicators would provide early warning of computational tractability problems. Synthesis times exceeding 24 hours for simplified procedure subsets would suggest complete procedures are intractable. Generated automata containing more than 1,000 discrete states would indicate the discrete state space is too large for efficient verification. Specifications flagged as unrealizable by FRET or Strix would reveal fundamental conflicts in the formalized procedures. Reachability analysis failing to converge within reasonable time bounds would show that continuous mode verification cannot be completed with available computational resources.

The contingency plan for computational intractability is to reduce scope to a minimal viable startup sequence. This reduced sequence would cover only cold shutdown to criticality to low-power hold, omitting power ascension and other operational phases. The subset would still demonstrate the complete methodology while reducing computational burden. The research contribution would remain valid even with reduced scope, proving that formal hybrid control synthesis is achievable for safety-critical nuclear applications. The limitation to simplified operational sequences would be explicitly documented as a constraint rather than a failure.

5.2 Discrete-Continuous Interface Formalization

The second critical assumption concerns the mapping between boolean guard conditions in temporal logic and continuous state boundaries required for mode transitions. This interface represents the fundamental challenge of hybrid systems: relating discrete switching logic to continuous dynamics. Temporal logic operates on boolean predicates, while continuous control requires reasoning about differential equations and reachable sets. Guard conditions requiring complex nonlinear predicates may resist boolean abstraction, making synthesis intractable. Continuous safety regions that cannot be expressed as conjunctions of verifiable constraints would similarly create insurmountable verification challenges. The risk extends beyond static interface definition to dynamic behavior across transitions: barrier certificates may fail to exist for proposed transitions, or continuous modes may be unable to guarantee convergence to discrete transition boundaries.

Early indicators of interface formalization problems would appear during both synthesis and verification phases. Guard conditions requiring complex nonlinear predicates that resist boolean abstraction would suggest fundamental misalignment between discrete specifications and continuous realities. Continuous safety regions that cannot be expressed as conjunctions of half-spaces or polynomial inequalities would indicate the interface between discrete guards and continuous invariants is too complex. Failure to construct barrier certificates proving safety across mode transitions would reveal that continuous dynamics cannot be formally related to discrete switching logic. Reachability analysis showing that continuous modes cannot reach intended transition boundaries from all possible initial conditions would demonstrate the synthesized discrete controller is incompatible with achievable continuous behavior.

The primary contingency for interface complexity is restricting continuous modes to operate within polytopic invariants. Polytopes are state regions defined as intersections of linear half-spaces, which map directly to boolean predicates through linear inequality checks. This restriction ensures tractable synthesis while maintaining theoretical rigor, though at the cost of limiting expressiveness compared to arbitrary nonlinear regions. The discrete-continuous interface remains well-defined and verifiable with polytopic restrictions, providing a clear fallback position that preserves the core methodology. Conservative over-approximations offer an alternative approach: a nonlinear safe region can be inner-approximated by a polytope, sacrificing operational flexibility to maintain formal guarantees. The three-mode classification already structures the problem to minimize complex transitions, with critical safety properties concentrated in expulsory modes that can receive additional design attention.

Mitigation strategies focus on designing continuous controllers with discrete transitions as primary objectives from the outset. Rather than designing continuous control laws independently and verifying transitions post-hoc, the approach uses transition requirements as design constraints. Control barrier functions provide a systematic method to synthesize controllers that guarantee forward invariance of safe sets and convergence to transition boundaries. This design-for-verification approach reduces the likelihood that interface complexity becomes insurmountable. Focusing verification effort on expulsory modes—where safety is most critical—allows more complex analysis to be applied selectively rather than uniformly across all modes, concentrating computational resources where they matter most for safety assurance.

5.3 Procedure Formalization Completeness

The third assumption is that existing startup procedures contain sufficient detail and clarity for translation into temporal logic specifications. Nuclear operating procedures, while extensively detailed, were written for human operators who bring contextual understanding and adaptive reasoning to their interpretation. Procedures may contain implicit knowledge, ambiguous directives, or references to operator judgment that resist formalization in current specification languages. Underspecified timing constraints, ambiguous condition definitions, or gaps in operational coverage would cause synthesis to fail or produce incorrect automata. The risk is not merely that formalization is difficult, but that current procedures fundamentally lack the precision required for autonomous control, revealing a gap between human-oriented documentation and machine-executable specifications.

Several indicators would reveal formalization completeness problems early in the project. FRET realizability checks failing due to underspecified behaviors or conflicting requirements would indicate procedures do not form a complete specification. Multiple valid interpretations of procedural steps with no clear resolution would demonstrate procedure language is insufficiently precise for automated synthesis. Procedures referencing "operator judgment," "as appropriate," or similar discretionary language for critical decisions would explicitly identify points where human reasoning cannot be directly formalized. Domain experts unable to provide crisp answers to specification questions about edge cases would suggest the procedures themselves do not fully define system behavior, relying instead on operator training and experience to fill gaps.

The contingency plan treats inadequate specification as itself a research contribution rather than a project failure. Documenting specific ambiguities encountered would create a taxonomy of formalization barriers: timing underspecification, missing preconditions, discretionary actions, and undefined failure modes. Each category would be analyzed to understand why current procedure-writing practices produce these gaps and what specification languages would need to address them. Proposed extensions to FRETish or similar specification languages would demonstrate how to bridge the gap between current procedures and the precision needed for autonomous control. The research output would shift from "here is a complete autonomous controller" to "here is what formal autonomous control requires that current procedures do not provide, and here are language extensions to bridge that gap." This contribution remains valuable to both the nuclear industry and formal methods community, establishing clear requirements for next-generation procedure development and autonomous control specification languages.

Early-stage procedure analysis with domain experts provides the primary mitigation strategy. Collaboration through the University of Pittsburgh Cyber Energy Center enables identification and resolution of ambiguities before synthesis attempts, rather than discovering them during failed synthesis runs. Iterative refinement with reactor operators and control engineers can clarify procedural intent before formalization begins, reducing the risk of discovering insurmountable specification gaps late in the project. Comparison with procedures from multiple reactor designs—pressurized water reactors, boiling water reactors, and advanced designs—may reveal common patterns and standard ambiguities amenable to systematic resolution. This cross-design analysis would strengthen the generalizability of any proposed specification language extensions, ensuring they address industry-wide practices rather than specific quirks.

6 Broader Impacts

Nuclear power presents both a compelling application domain and an urgent economic challenge. Recent interest in powering artificial intelligence infrastructure has renewed focus on small modular reactors (SMRs), particularly for hyperscale datacenters requiring hundreds of megawatts of continuous power. Deploying SMRs at datacenter sites would minimize transmission losses and eliminate emissions from hydrocarbon-based alternatives. However, nuclear power economics at this scale demand careful attention to operating costs.

According to the U.S. Energy Information Administration's Annual Energy Outlook 2022, advanced nuclear power entering service in 2027 is projected to cost \$88.24 per megawatt-hour [**?**]. Datacenter electricity demand is projected to reach 1,050 terawatt-hours annually by 2030 [**?**]. If this demand were supplied by nuclear power, the total annual cost of power generation would exceed \$92 billion. Within this figure, operations and maintenance represents a substantial component. The EIA estimates that fixed O&M costs alone account for \$16.15 per megawatt-hour, with additional variable O&M costs embedded in fuel and operating expenses [**?**]. Combined, O&M-related costs represent approximately 23–30% of the total levelized cost of electricity, translating to \$21–28 billion annually for projected datacenter demand.

This research directly addresses the multi-billion-dollar O&M cost challenge through high-assurance autonomous control. Current nuclear operations require full control room staffing for each reactor, whether large conventional units or small modular designs. These staffing requirements drive the high O&M costs that make nuclear power economically challenging, particularly for smaller reactor designs where the same staffing overhead must be spread across lower power output. Synthesizing provably correct hybrid controllers from formal specifications can automate routine operational sequences that currently require constant human oversight. This enables a fundamental shift from direct operator control to supervisory monitoring, where operators oversee multiple autonomous reactors rather than manually controlling individual units.

The correct-by-construction methodology is critical for this transition. Traditional automation approaches cannot provide sufficient safety guarantees for nuclear applications, where regulatory requirements and public safety concerns demand the highest levels of assurance. Formally verifying both the discrete mode-switching logic and the continuous control behavior, this research will produce controllers with mathematical proofs of correctness. These guarantees enable automation to safely handle routine operations—startup sequences, power level changes, and normal operational transitions—that currently require human operators to follow written procedures. Operators will remain in supervisory roles to handle off-normal conditions and provide authorization for major operational changes, but the routine cognitive burden of procedure execution shifts to provably correct automated systems that are much cheaper to operate.

SMRs represent an ideal deployment target for this technology. Nuclear Regulatory Commission certification requires extensive documentation of control procedures, operational requirements, and safety analyses written in structured natural language. As described in our approach, these regulatory documents can be translated into temporal logic specifications using tools like FRET, then synthesized into discrete switching logic using reactive synthesis tools, and finally verified using reachability analysis and barrier certificates for the continuous control modes. The infrastructure of requirements and specifications already exists as part of the licensing process, creating a direct pathway from existing regulatory documentation to formally verified autonomous controllers.

Beyond reducing operating costs for new reactors, this research will establish a generalizable framework for autonomous control of safety-critical systems. The methodology of translating operational procedures into formal specifications, synthesizing discrete switching logic, and verifying continuous mode behavior applies to any hybrid system with documented operational requirements. Potential applications include chemical process control, aerospace systems, and autonomous transportation, where similar economic and safety considerations favor increased autonomy with provable correctness guarantees. Demonstrating this approach in nuclear power—one of the most regulated and safety-critical domains—will establish both the technical feasibility and regulatory pathway for broader adoption across critical infrastructure.

“If it works here, it works anywhere — strong closing argument.

7 Schedule, Milestones, and Deliverables

This research will be conducted over six trimesters (24 months) of full-time effort following the proposal defense in Spring 2026. The work progresses sequentially through three main research thrusts before culminating in integrated demonstration and validation.

The first semester (Spring 2026) focuses on Thrust 1, translating startup procedures into formal temporal logic specifications using FRET. This establishes the foundation for automated synthesis by converting natural language procedures into machine-readable requirements. The second semester (Summer 2026) addresses Thrust 2, using Strix to synthesize the discrete automaton that defines mode-switching behavior. With the discrete structure established, the third semester (Fall 2026) develops the continuous controllers for each operational mode through Thrust 3, employing reachability analysis and barrier certificates to verify that each mode satisfies its transition requirements. Integration and validation occupy the remaining three semesters.

Figure 3 shows the complete project schedule including research thrusts, major milestones, and planned publications.

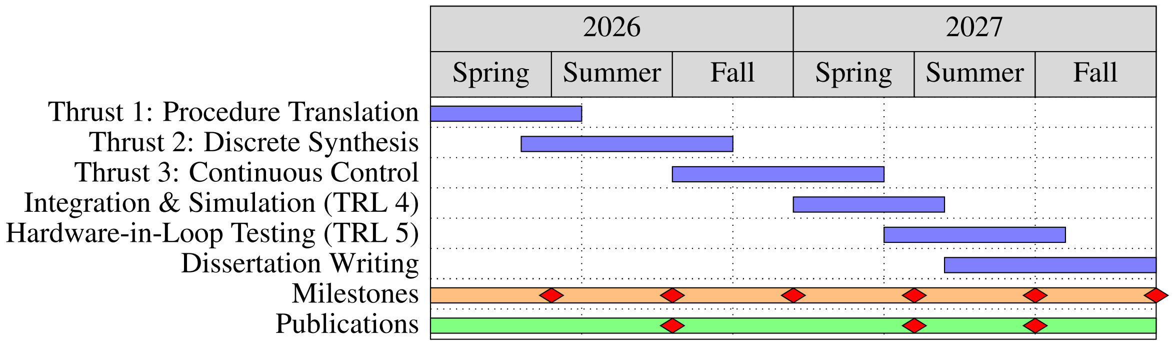

Figure 3: Project schedule showing major research thrusts, milestones (orange row), and publications (green row). Red diamonds indicate completion points. Overlapping bars indicate parallel work where appropriate.

7.1 Milestones and Deliverables

Six major milestones mark critical validation points throughout the research. M1 (Month 4) confirms that startup procedures have been successfully translated to temporal logic using FRET with realizability analysis demonstrating consistent and complete specifications. M2 (Month 8) validates computational tractability by demonstrating that Strix can synthesize a complete discrete automaton from the formalized specifications. This milestone delivers a conference paper submission to NPIC&HMIT documenting the procedure-to-specification translation methodology. M3 (Month 12) achieves TRL 3 by proving that continuous controllers can be designed and verified to satisfy discrete transition requirements. This milestone delivers an internal technical report demonstrating component-level verification. M4 (Month 16) achieves TRL 4 through integrated simulation demonstrating that component-level correctness composes to system-level correctness. This milestone delivers a journal paper submission to IEEE Transactions on Automatic Control presenting the complete hybrid synthesis methodology. M5 (Month 20) achieves TRL 5 by demonstrating practical implementability on industrial hardware. This milestone delivers a conference paper submission to NPIC&HMIT or CDC documenting hardware implementation and experimental validation. M6 (Month 24) completes the dissertation documenting the entire methodology, experimental results, and research contributions.

Clear timeline with publication targets — shows you have a plan.