Can Someone Prove Your Operator Won't Get Distracted? A Gentle Introduction to Formal Methods in Human Factors

Stephen B. Gilbert[1], Parth Ganeriwala[5], James I. Lathrop[1], Amanda K. Newendorp[1], Stephen J. Fieffer[1], Peggy Wu[3], Isaac Amundson[2], Candice Chambers[5], Adam Kohl[1], Shayama Khan[1], Mohammadamin Sanaei[1], Junaid Babar[2], Timothy Wang[3], David Musliner[4], Robert P. Goldman[4], Jeremy Gottlieb[4], Eliot Winer[1], Michael C. Dorneich[1], and Siddhartha Bhattacharyya[5]

[1]Iowa State University
[2]Collins Aerospace
[3]RTX Technology Research Center
[4]Smart Information Flow Technologies
[5]Florida Institute of Technology

Formal methods have proven helpful in modeling complex systems and ensuring they function correctly. This paper offers some basic guidelines for HFES practitioners in choosing whether to employ formal methods. It also touches on emerging approaches to modeling human behavior, attempting to answer two questions: 1) When are formal methods worth the effort? and 2) How do formal methods apply to humans? A table of questions is offered to facilitate a discussion of tradeoffs in choosing a formal methods approach.

INTRODUCTION

Imagine that someone told you that they could mathematically prove that an operator using your system would get distracted while working only 1% of the time. Furthermore, this same person suggested that with the right planning, they could guarantee with 90% confidence that your overall human-machine system would not fail due to human error. If you are unfamiliar with formal methods, you might think these claims are quite surprising. However, formal methods is a powerful tool that can be applied in a variety of human factors contexts; this paper attempts to offer both a brief introduction to the formal methods approach and some recent examples relevant to the HFES community, particularly in the domain of modeling human behavior. This paper discusses two key questions with formal methods: 1) When are they worth the effort? and 2) How do they apply to human behavior?

A "formal" method has a mathematical or computational foundation that enables an automated approach to exploring a system's functionality (Wing, 1990). For example, imagine an engineer designing the gates at a railroad crossing that lower to block the road when a train approaches. In the U.S., an engineer must follow rules like, "The crossing signal must begin to activate 20s before the train crosses" and "The lights on the signal must flash for 3 seconds before the gates begin to lower" (U.S. DOT, 2002). Rules like this serve as specifications and requirements of the crossing gate system.

At some point the engineer might wonder, "Do I have any conflicting or incomplete requirements? Is my design correct with respect to my requirements?" When using traditional "informal" methods, there might not be an easy way to answer those questions. If, however, the engineer had a computational model of the crossing gate, including information about train speed, train distance, and typical automobile traffic flow across the railroad crossing, formal methods could be used to build a formal model and as if satisfies system requirements, incorporates realizable operating assumptions, violates any safety properties, among others.

A formal model is typically constructed by specifying all states that the system could potentially reach, including *unsafe* states, and the transition criteria between states. In our railroad crossing example, the state description would include information about the gate (up, down, in motion), trains (speed, direction), cars (position relative to tracks), and any adjacent signals such as a traffic light in close proximity. Transitions between the states would model how these various parameters can change.

Naturally, with such a model, we wish to know if there is a sequence of inputs that will lead to an unsafe state. If there exists such a sequence of inputs, it may indicate a flaw in our model induced by missing or incorrect requirements and assumptions. Formal methods tools, such as Model Checkers (Clarke et al., 1999), can effectively explore if the system can reach an unsafe or undesirable state (such as a car and train simultaneously at the same location), and an operational path from the system's initial state to this state. The model checker could discover unusual edge cases that were not tested for and that the engineer might not think of. Perhaps during heavy automotive traffic that spans the crossing, cars might not have time to avoid the cross-

ing gates. Or perhaps in the rare case of a second train arriving at the crossing from the other direction less than 20 seconds after the first train has finished crossing, the status of the gate is unspecified, and revised requirements are needed.

This example illustrates one of the key benefits of formal methods: verifying the completeness and consistency of a system's specification. Formal methods enable logically sound reasoning about precise assumptions and requirements. When these assumptions and requirements are satisfied, the conclusions are mathematically guaranteed to be true. Any issue with the validity of the conclusion stems from the validity of the assumptions or requirements themselves. Some HFES practitioners might argue that this idea puts the practicality of formal methods at risk, since assumptions are almost always somewhat questionable. However, one of the benefits of formal methods is that the approach offers a robust framework to detect inconsistencies across assumptions or requirements.

The product of employing formal methods is a formal model of system requirements and assumptions satisfying the requisite desirable properties. At this juncture, the requirements and assumptions have been *formalized*, shown to be error-free and consistent with the intention of the system designer. This formal model can serve as the basis for the next stage of system development. But is creating this model worth the system designer's time? That is the first critical question with formal methods: ***When are they worth the effort?***

Formal methods and their applications have grown in popularity in recent years (ter Beek et al., 2024). This article explores that value proposition question further with several examples and offers guidance in navigating the trade-offs of choosing formal methods.

FORMAL METHODS APPLICATIONS

While formal methods predate computing devices, formal methods used today focus on computing hardware and software. For example, formal methods are an integral part of Intel's chip design process (Harrison, 2010). A CPU is a very complicated device, and it is expensive to design. Once produced and shipped into millions of products, the chips are near impossible to fix or recall. Creating that detailed computational model makes business sense for Intel; executives want to be able to mathematically guarantee that their chips will not fail.

Formal methods are also used to make software engineering more rigorous, leading to fewer software bugs and avoiding potentially catastrophic events (Woodcock et al., 2009). It is not surprising that as software complexity increases, it is difficult to guarantee that a system will perform as expected without errors (Tucker, 2022). Complex systems are also more difficult and expensive to test. For example, a multi-flaw error that occurs only under a combination of very specific conditions may take thousands and thousands of hours of difficult-to-replicate tests to manifest itself. Such failures can be more rapidly found (and fixed) with model checking's exhaustive state-space exploration.

Developers of safety-critical applications, including NASA, FAA, and medical equipment developers, know that the consequences of errors in these systems can be costly or deadly. Indeed, they require guarantees that the systems will not fail or, in the case of flight systems, fail only incredibly rarely. Formal methods have been successfully applied to domains such as autonomous drone flight ("We can prove that if the following conditions are met, the autonomous drone will not fall out of the sky") (Lee et al., 2012), railway safety ("We can prove that there will be no collisions...") (Ferrari and Beek, 2022), Amazon Web Services security (Newcombe et al., 2015), autonomous vehicles (**O'Connor**), and other examples (ter Beek et al., 2024).

This increased use of formal methods has partially been enabled by scalable computing. If a system has 100 components, and any 5 of them might affect each other at some point, there are over 75,000,000 states to explore (100 choose 5). Only through automated reasoning and today's large scale of computing power does model-checking all these states become a tractable problem. Model checking systems can require a large amount of computing resources, but if the cost of failure is exorbitant or safety-critical, then formal methods may be worth the effort.

To appreciate the appeal of a mathematical guarantee, and understand another class of formal methods, *theorem proving*, consider high/secondary school geometry.

To prove that the interior angles of a triangle sum to 180 degrees, students do not draw every triangle possible and measure the angles. They are able to derive this result from other geometric theorems, such as (1) a line describes an angle of 180 degrees, (2) the parallel lines and transversal theorem, and (3) the opposite angle theorem. A theorem prover works similarly: it can be used to prove a theorem using a set of axioms. When the axioms are insufficient to complete the proof, additional axioms may need to be provided, or intermediate lemmas may need to be proved, to complete the proof.

Using principles of mathematics, logic, physics, and engineering, (in addition to theorems about geometry) formal methods can be similarly used to prove theorems about computational systems. In the earlier discussion of model checking, we referred to "paths" to failure states. These paths can actually be considered to be theorem proofs. An engineer may want to prove that given inputs A, B, and C, the system will successfully perform at rate X. Just as the triangle proof begins with givens and ends with a proven theorem, the engineer can enter those inputs into a theorem prover, along with the model, and ask it whether the theorem "System performs at rate X" can be proven.

FORMAL METHODS AND HUMAN BEHAVIOR

The examples described so far have focused on systems, their requirements, and their performance. But formal methods can play a large role in human-machine systems, and areas of human-computer interaction (HCI) such as user interface design. This section addresses the second question: ***How do formal methods apply to human behavior?***

In the HFES community, a symposium on formal methods in human factors was held at the 2017 conference (Bolton, 2017),

bringing together four papers that used formal methods in different ways. These papers and others mentioned below help us answer several questions that arise about human interaction with systems:

1. What workflows will emerge between humans and a particular system?
2. Can we predict errors and challenges that are both cognitive and physical?
3. How do those interactions change according to individual differences or behavioral inconsistency in the humans?
4. How do those interactions change if the system is intelligent?

Question 1 has been addressed frequently in investigations of human-machine systems. One paper used formal methods to analyze workflows between humans and autonomous flight systems (Ma and Feigh, 2017). Another paper employed formal methods to predict interactions between a human operator and a teleoperated robot under latency conditions (Cubuktepe and Topcu, 2017).

Question 2 was addressed, for example, in a formal methods paper that explored a human's use of the affordances of aircraft cabin doors (Abbate and Bass, 2017), and in an analysis of human errors that result from the use of Automatic Teller Machine (ATM) (Roggenbach et al., 2022). A recent paper used formal methods to predict some of the usability concerns that led to the Boeing 737MAX issues (Barshi et al., 2024).

Question 3 is also addressed by the ATM analysis, since it included specific parameters for short-term and long-term memory of users that could be varied within the model to reflect user differences. Another paper used probabilistic formal methods to accommodate human variability within a reliability analysis: how likely is it that employees in a pharmacy will dispense incorrect pills? (Zheng et al., 2017).

Regarding Question 4, formal methods have also been applied in the growing discipline of AI. A recent paper attempts to increase user trust in AI systems within a meteorological context using formal methods (Tavolato-Wötzl and Tavolato, 2023). Given the rise of research in human-AI teaming (HAT, also called human-autonomy teaming or human-agent teaming), a recent PhD dissertation applied formal methods to HAT and proposed a Belief Graph approach for modeling the human's various belief states about the AI system (Brännström, 2025).

Alan Dix, a long-time authority in HCI, distinguishes between three kinds of models when applying formal methods to HCI: dialogue models (a network of user actions), full state specifications (which includes both the system's states and the user's actions), and abstract interaction models (focused on the properties of classes of systems) (Dix, 2023). Another approach in applying formal methods to HCI considers models for each "layer" of the interactive system: control logic, user interface, use cases, the user, and the context (Campos and Harrison, 2025). Optimal requirements for each layer might include consistency, predictability, reversibility (undo), and completeness.

One approach to formal verification in HMSs involves model checking, where a system's behavior is exhaustively analyzed against a predefined set of logical constraints. In cognitive modeling, modeling architectures such as ACT-R and Soar offer structured representations of human cognition that can be formally specified and translated into verification environments such as nuXmv or TLA+ (Bhattacharyya et al., 2021; Ganeriwala et al., 2025; Langenfeld et al., 2018; Narayan et al., 2023). This process allows for the detection of potential failure modes in human-machine interactions, particularly when cognitive workload, task complexity, and decision-making under uncertainty are considered. By leveraging computational logic, it becomes possible to reason about human cognitive states under various operational conditions and ensure that specified behavioral constraints are met.

Another approach involves probabilistic verification, where the inherent variability in human performance is captured using stochastic processes. This is particularly useful when modeling inconsistent human behaviors in dynamic environments, such as aviation or medical systems, where errors may arise from fatigue, stress, or distractions. Probabilistic model checkers such as PRISM or STORM can accommodate human performance variability by incorporating likelihood estimates into the formal models (Hensel et al., 2022; Kwiatkowska et al., 2011), enabling an analysis of how certain human errors propagate through a system. This technique has been applied in prior research to evaluate workflow reliability in aviation and healthcare, ensuring that system constraints account for human unpredictability (Bhattacharyya et al., 2018; Paassen et al., 2014).

A final comment about the use of formal reasoning on human behavior: using formal methods for analysis of human machine systems has been done for years, but what is newer is combining formal systems models with human research data. For example, if a research paper claims statistically significant differences in human performance under Condition A vs. Condition B, could we use those research results to create a human model that serves as a component of a formal methods analysis? Answering this question is one of the goals of the project supporting this paper (Wilding, 2023). Often HFES research is useful for understanding the dynamics of the specific conditions of the study but not very generalizable beyond those conditions. A formal methods modeler would almost prefer research closer to studies from the psychophysics era of HFES research, in which results often consisted of thresholds such as perceptual just noticable differences or capacity caps in working memory.

MAKING THE CHOICE FOR FORMAL METHODS

To aid an HFES praticitioner in deciding whether to use formal methods, Table 1 offers a series of questions. The more "Yes" answers the questions yield, the more likely formal methods will be helpful.

This paper offers human factors practitioners practical guidance in considering the trade-offs associated with formalizing human behavior. Although formal verification can provide rigorous guarantees, defining an exhaustive set of human behavioral constraints remains a challenge due to the infinite variability in human cognition and interaction (Amalberti, 2001; Rushby, 1997). It is critical to explore how formal methods frameworks can balance systematic abstraction with practi-

Table 1: If the answer to many of these questions is Yes, consider using formal methods.

System Questions	
1	Is failure highly consequential?
2	Is the system highly complex?
3	Is it difficult to test your system completely?
4	Is it important to have a solid, agreed-upon specification of your system (rather than it changing frequently)?
5	Is it critical that the system be correct the first time deployed?
Human Questions	
6	Are there some reasonable parameters to describe the human reaction to the system, e.g., a distribution of reaction times?
7	If there are notable individual differences between users, e.g., between novices an experts, are there data to describe those?

cal applicability, ensuring that models remain computationally tractable while capturing critical edge cases that may lead to system failures. The focus is on understanding where formal methods can realistically provide guarantees in human factors applications and where their limitations necessitate alternative approaches. Through an interdisciplinary analysis of existing formal methods applications in cognitive systems, automation, and human-computer interaction, this paper seeks to bridge the gap between human factors research and formal verification. The findings will provide a foundation for further empirical validation and integration into real-world safety-critical applications.

CONCLUSION

Formal methods, particularly model checking and theorem proving, have been extensively used in verifying safety-critical systems and analyzing human-machine systems. Developing a way to harness current HFES research results for modeling humans within this approach could have strong potential, but more exploration within the HFES community is needed.

ACKNOWLEDGMENTS

This effort was sponsored by the Defense Advanced Research Projects Agency (DARPA) under agreement number HR0011-24-9-0439. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

Abbate, A. J., & Bass, E. J. (2017). Modeling affordance using formal methods. In *Proceedings of the HFES annual meeting* (pp. 723–727, Vol. 61). Sage. https://doi.org/10.1177/1541931213601666

Amalberti, R. (2001). The paradoxes of almost totally safe transportation systems. *Safety Science*, *37*(2), 109–126. https://doi.org/https://doi.org/10.1016/S0925-7535(00)00045-X

Barshi, I., Degani, A., Mauro, R., & Mumaw, R. J. (2024). Models of human-automation systems: Initial analysis of the boeing 737max design. In *Proceedings of the HFES annual meeting* (pp. 835–840, Vol. 68). Sage. https://doi.org/10.1177/10711813241279805

Bhattacharyya, S., Davis, J., Gupta, A., Narayan, N., & Matessa, M. (2021). Assuring increasingly autonomous systems in human-machine teams: An urban air mobility case study. In M. Farrell & M. Luckcuck (Eds.), *Third workshop on formal methods for autonomous systems (FMAS 2021)* (pp. 150–166, Vol. 348). Electronic Proceedings in Theoretical Computer Science. https://doi.org/10.4204/EPTCS.348.11

Bhattacharyya, S., Eskridge, T. C., Neogi, N. A., Carvalho, M., & Stafford, M. (2018). Formal assurance for cooperative intelligent autonomous agents. In *LNCS* (pp. 20–36, Vol. 10811). Springer. https://doi.org/10.1007/978-3-319-77935-5_2

Bolton, M. L. (2017). Novel developments in formal methods for human factors engineering. In *Proceedings of the HFES annual meeting* (pp. 715–717, Vol. 61). Sage. https://doi.org/10.1177/1541931213601664

Brännström, A. (2025). *Formal methods for verification in human-agent interaction* [Doctoral dissertation, Umeå University].

Campos, J. C., & Harrison, M. D. (2025). Formal approaches for interactive systems. In J. Vanderdonckt, P. Palanque, & M. Winckler (Eds.), *Handbook of human computer interaction* (pp. 1–28). Springer Nature Switzerland. https://doi.org/10.1007/978-3-319-27648-9_120-1

Clarke, E. M., Grumberg, O., & Peled, D. A. (1999). *Model checking*. MIT Press.

Cubuktepe, M., & Topcu, U. (2017). Intent prediction in shared control with delayed feedback. In *Proceedings of the HFES annual meeting* (pp. 733–734, Vol. 61). Sage. https://doi.org/10.1177/1541931213601668

Dix, A. (2023). Modelling interactions: Digital and physical. In A. Cerone (Ed.), *Formal methods for an informal world: Ictac 2021 summer school, virtual event, astana, kazakhstan, september 1–7, 2021, tutorial lectures* (pp. 1–29). Springer International Publishing. https://doi.org/10.1007/978-3-031-43678-9_1

Ferrari, A., & Beek, M. H. T. (2022). Formal methods in railways: A systematic mapping study. *ACM Comput. Surv.*, *55*(4), Article 69. https://doi.org/10.1145/3520480

Ganeriwala, P., Matessa, M., Bhattacharyya, S., Jones, R. M., Davis, J., Kaur, P., Rollini, S. F., & Neogi, N. (2025). Design and validation of learning aware hmi for learning-enabled increasingly autonomous systems. In *Proceedings of SysCon 2025*. https://doi.org/10.48550/arXiv.2501.18506

Harrison, J. (2010, April). Formal Methods at Intel — An Overview. Retrieved May 1, 2025, from https://shemesh.larc.nasa.gov/NFM2010/talks/harrison.pdf

Hensel, C., Junges, S., Katoen, J.-P., Quatmann, T., & Volk, M. (2022). The probabilistic model checker storm. *International Journal on Software Tools for Technology Transfer*, *24*(4), 589–610. https://doi.org/10.1007/s10009-021-00633-z

Kwiatkowska, M., Norman, G., & Parker, D. (2011). Prism 4.0: Verification of probabilistic real-time systems. *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23*, 585–591.

Langenfeld, V., Westphal, B., Albrecht, R., & Podelski, A. (2018). But does it really do that? using formal analysis to ensure desirable act-r model behaviour. In *Proceedings of the annual meeting of the Cognitive Science Society* (Vol. 40). https://escholarship.org/uc/item/3hg423jb

Lee, D.-A., Sung, S., Yoo, J., & Kim, D.-H. (2012). Formal modeling and verification of operational flight program in a small-scale unmanned helicopter. *Journal of Aerospace Engineering*, *25*(4), 530–540.

Ma, L. M., & Feigh, K. M. (2017). Jumpstarting modelling systems design: A generalized xml abstraction of simulation model. In *Proceedings of the HFES annual meeting* (pp. 718–722, Vol. 61). Sage. https://doi.org/10.1177/1541931213601665

Narayan, N., Ganeriwala, P., Jones, R. M., Matessa, M., Bhattacharyya, S., Davis, J., Purohit, H., & Rollini, S. F. (2023). Assuring learning-enabled increasingly autonomous systems*. In *2023 IEEE international systems conference (syscon)* (pp. 1–7). https://doi.org/10.1109/SysCon53073.2023.10131227

Newcombe, C., Rath, T., Zhang, F., Munteanu, B., Brooker, M., & Deardeuff, M. (2015). How amazon web services uses formal methods. *Commun. ACM*, *58*(4), 66–73. https://doi.org/10.1145/2699417

Paassen, M. M. v., Bolton, M. L., & Jiménez, N. (2014). Checking formal verification models for human-automation interaction. In *2014 IEEE international conference on systems, man, and cybernetics (SMC)* (pp. 3709–3714). https://doi.org/10.1109/SMC.2014.6974507

Roggenbach, M., Cerone, A., Schlingloff, B.-H., Schneider, G., & Shaikh, S. A. (2022). Correction to: Formal methods for software engineering. In *Formal methods for software engineering: Languages, methods, application domains* (pp. C1–C1). Springer International Publishing. https://doi.org/10.1007/978-3-030-38800-3_10

Rushby, J. (1997). Formal methods and their role in the certification of critical systems. In R. Shaw (Ed.). Springer. https://doi.org/10.1007/978-1-4471-0921-1_1

Tavolato-Wötzl, C., & Tavolato, P. (2023). Enhancing trust in machine learning systems by formal methods. In *Machine learning and knowledge extraction* (pp. 170–187). Springer.

ter Beek, M. H., Chapman, R., Cleaveland, R., Garavel, H., Gu, R., ter Horst, I., Keiren, J. J., Lecomte, T., Leuschel, M., & Rozier, K. Y. (2024). Formal methods in industry. *Formal Aspects of Computing*, *37*(1), 1–38.

Tucker, J. V. (2022). Origins and development of formal methods. In *Formal methods for software engineering: Languages, methods, application domains* (pp. 455–488). Springer International Publishing. https://doi.org/10.1007/978-3-030-38800-3_9

U.S. DOT. (2002). Grade crossing signal system safety technical manual. https://railroads.dot.gov/sites/fra.dot.gov/files/2020-08/2002-08_Signal_Technical_Manual.pdf

Wilding, M. (2023). Ics: Intrinsic cognitive security. https://www.darpa.mil/research/programs/intrinsic-cognitive-security

Wing, J. M. (1990). A specifier's introduction to formal methods. *Computer*, *23*(9), 8–22.

Woodcock, J., Larsen, P. G., Bicarregui, J., & Fitzgerald, J. (2009). Formal methods: Practice and experience. *ACM Computing Surveys*, *41*(4), 1–36. https://doi.org/10.1145/1592434.1592436

Zheng, X., Bolton, M. L., Daly, C., & Feng, L. (2017). A formal human reliability analysis of a community pharmacy dispensing procedure. In *Proceedings of the HFES annual meeting* (pp. 728–732, Vol. 61). Sage. https://doi.org/10.1177/1541931213601667