A Formal Approach to Modeling and Analyzing Human Taskload in Simulated Air Traffic Scenarios

Adam M. Houser and Matthew L. Bolton, Ph.D.

Department of Industrial and Systems Engineering

Overview

In complex systems, like the modern air traffic system, human operator taskload (the number of tasks the human needs to perform) can have a profound influence on how well the system performs. Because of the system's complexity, however, it can be difficult to determine all of the situations where taskload issues can arise. Simulation and formal verification have been used separately to explore human taskload in complex systems. However, both have problems that limit their usefulness. Simulation scales well, but can miss critical operating conditions. Formal verification can mathematically prove whether or not a system does or does not adhere to desirable properties, but is severely restricted by scalability. This poster will describe the method we have developed that allows us to use formal verification synergistically with simulation. Specifically, we avoid scalability problems by using formal verification analyses to find interesting taskload conditions in abstract models of simulation traces. These conditions can then be fed back to the simulation for deeper analysis. We provide the background necessary for understanding this method. We present the method itself along with several checkable specification properties that can be used to find interesting taskload conditions. Finally, we explore how our developments can be used in future analyses of human operator task load in aerospace applications and other domains.

Background

Below discusses necessary background information on formal verification and simulation.

Formal verification comes from the area of formal methods. **Formal methods** describes a body of well-defined mathematical languages and techniques for the modeling, specification, and verification of target systems. A **model** is a system abstraction described in a mathematically-coherent manner (usual a state machine). **Specifications** are (desirable and/or undesirable) properties to which a system model should adhere (usually asserted in a temporal logic). **Formal verification** mathematically proves whether or not a system model satisfies a specification. **Model Checking** performs formal verification by exhaustively searching a model's statespace to determine if a specification holds. A proved specification is confirmed, otherwise a counterexample is generated: a trace through the model's statespace showing exactly how the violation occurred.

WMC (Work Models the Compute) is an agent-based aerospace **simulation** currently in development at Georgia Tech. WMC allows analysts to evaluate the performance of complex air traffic control scenarios that account for the performance of both the aircraft and the human operator. In particular, WMC can account for human operator **taskload** by giving human operators limited-capacity active and inactive priority queues.[1]

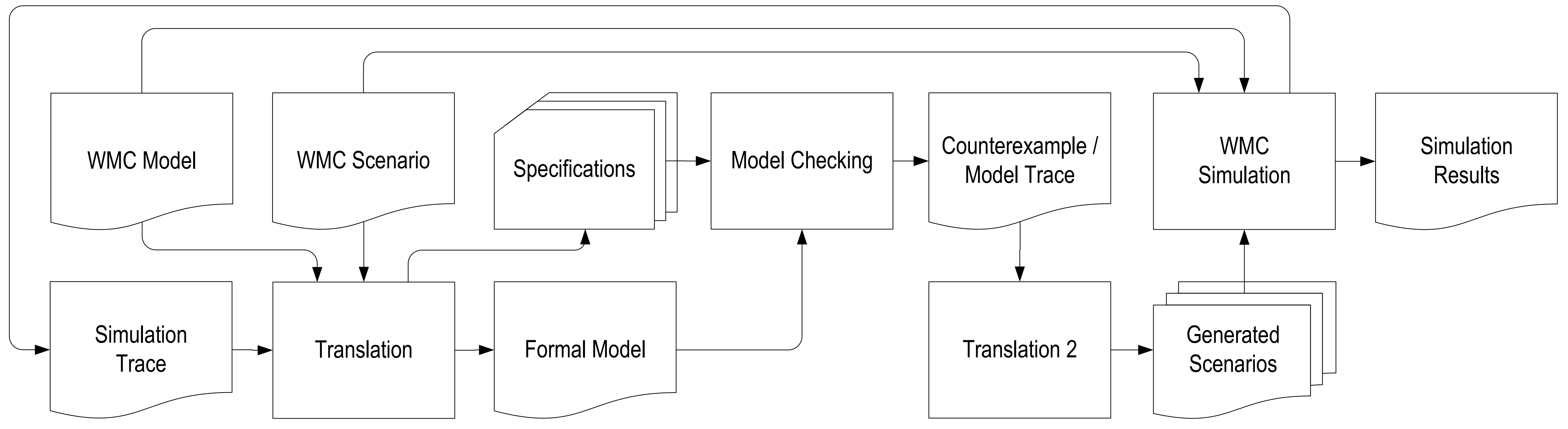

Figure 1. Method for the synergistic use of WMC simulation and model checking.

Implementation

Figure 1 illustrates how our method synergistically uses formal verification and the WMC simulation together. Several key components make the environment possible:

- The **WMC model** describes the work inherent to the air traffic control and aircraft flight domain, and the **scenario** describes a particular schedule of events to occur and a number of aircraft and ATC agents involved;[2-4]
- The **simulation** serves as the engine to compute through the scenario's dynamic, complex interactions and behavior;
- Generation of the **formal model** uses a **translator** (a software tool we have created) to read in the **simulation trace** and create a checkable model and **specifications** relating to taskload across different agents;
- Discovered **counterexamples** can be **translated** back into **WMC scenarios** and run again, thereby determining if problematic taskload conditions have been ameliorated.

One of the central challenges of this work was the development of a formal architecture that could efficiently implement WMC concepts while remaining parsimonious with its simulation framework. Our solution can be found in Figure 2, below.

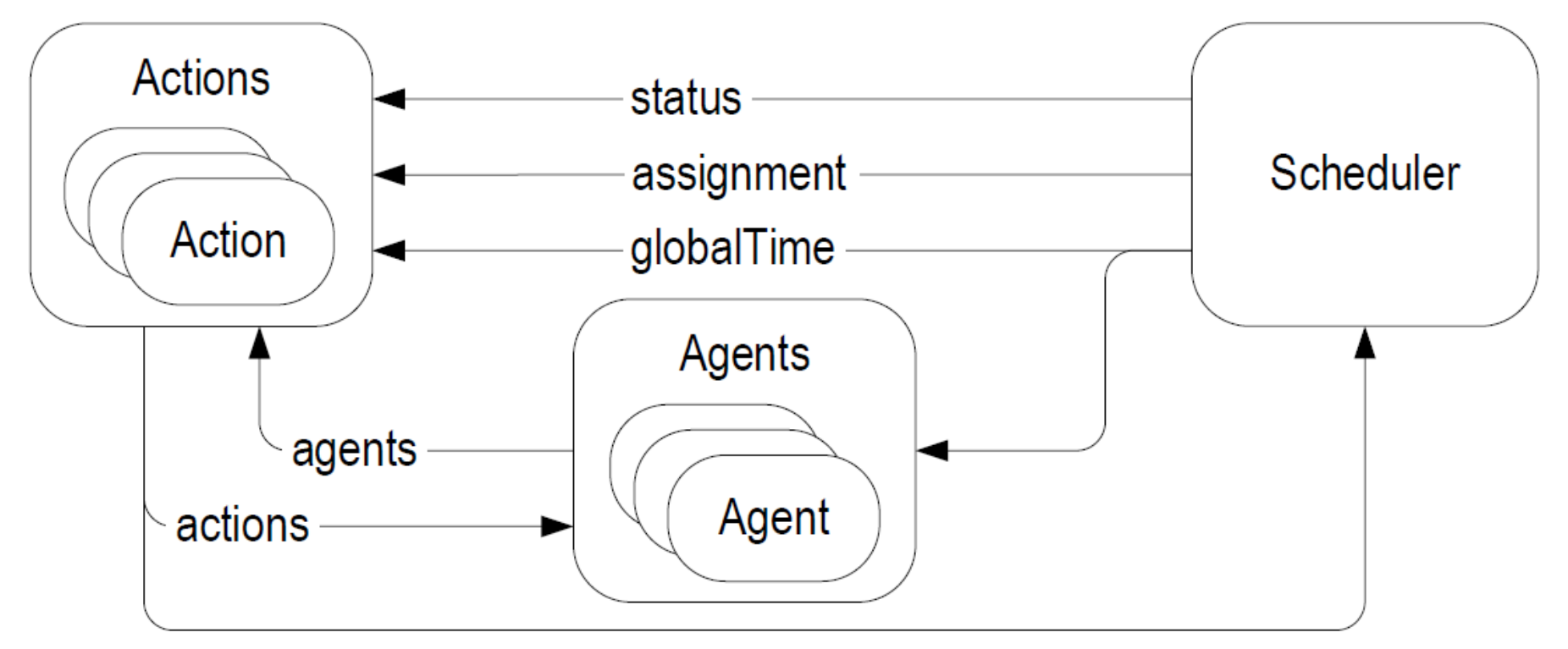

Figure 2. Formal modeling architecture representing WMC concepts.

Important aspects of functionality include interactions between the Scheduler, Actions, and Agents modules: how the simulation progresses based on its *status*; how Agents manage the actions in their active and inactive priority queues; and how the Scheduler assigns actions and advances *globalTime*. Timed automata are used to enable this behavior and maintain synchronicity as the model progresses across the scenario.[5,6]

Novel Computational Developments

Because integrating the WMC simulation environment with a formal modeling framework had not been done before, our work developed several techniques to reduce statespace complexity and accurately represent simulation constructs.

In particular, representing the dynamic action allocation and priority queue operations proved to be incredibly difficult to implement efficiently in our formal model. We addressed this by using λ-calculus to treat these operations as functions over λ-calculus sets.[7] For example, in

$$\begin{aligned} assignment = \lambda(i \in actionIDs):&\\ &actions[i].update = globalTime\\ &\wedge actions[i].state = notAssigned, \end{aligned}$$

the model checker determines which actions are ready for assignment by looking for those in the set of actionIDs with a state of *notAssigned* and an update time equal to the current *globalTime*. Drastic efficiency savings come from mapping actions ready for assignment to *True* and those not ready to *False*, rather than computing, storing, modifying, and performing operations on a complete and persistent set of *actionIDs*.

Analytic Capabilities

The following linear temporal logic (LTL) specifications are examples that can be used to search for a number of different human operator taskload conditions.

$$\mathbf{G}\left(\begin{array}{l}(status = doing)\\ \Rightarrow \left(\begin{array}{l} agent[i].activeCount\\ < agent[i].activeCapacity\end{array}\right)\end{array}\right)$$

This specification searches for situations in which a given agent's "active actions" queue count never meets or exceeds the capacity of the active queue. A counterexample would indicate an excessive taskload condition, such as an agent attempting to perform multiple actions simultaneously.

$$\mathbf{G}\left(\begin{array}{l}(status = doing)\\ \Rightarrow \left(\begin{array}{l} agent[i].inactiveCount\\ \le agent[i].inactiveCapacity\end{array}\right)\end{array}\right)$$

This specification looks for an overload of actions that have been delayed or interrupted (inactive), thereby exceeding the Inactive queue capacity; for human operators, an excessive burden on working memory can lead to forgotten actions.

$$\mathbf{G}\left(\begin{array}{l}(actions[j].state \neq notAssigned)\\ \Rightarrow \left(\left(\begin{array}{l} globalTime\\ - action[j].update\end{array}\right) < timeMax\right)\end{array}\right)$$

Human operators may also forget an action if it remains in working memory for too long. This specification searches for a condition where an action that has been assigned to an agent has not performed for an excessive amount of time (*timeMax*).

Current and Future Work

We are currently testing the capabilities of our framework. This preliminary instantiation includes hooks for finding excessive Active and Inactive queue load conditions, excessive taskload conditions, and excessive task interruptions across all simulation agents. We also have the ability to search for conflicts between agent actions. Future work will investigate whether taskload can cause air traffic delays in emerging air traffic control concepts.

References

[1]A. R. Pritchett and K. M. Feigh. (2011). "Simulating first-principles models of situated human performance." In *Proceedings of the IEEE First International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, pp. 144–151.

[2]A. R. Pritchett. (2013). "Simulation to assess safety in complex work environments," J. D. Lee and A. Kirlik, Eds. New York: Oxford University Press, pp. 352–366.

[3]A. R. Pritchett, K. M. Feigh, S. Y. Kim, and S. K. Kannan. (2014). Work models that compute to describe multiagent concepts of operation: Part 1. *Journal of Aerospace Information Systems,11*(10), 610–622.

[4]A. R. Pritchett, S. Y. Kim, and K. M. Feigh. (2014). Modeling human–automation function allocation. *Journal of Cognitive Engineering and Decision Making, 8*(1), 33–51.

[5]R. Alur and D. L. Dill. (1994). A theory of timed automata. *Theoretical Computer Science*, *126*(2), 183–235.

[6]B. Dutertre and M. Sorea, "Timed systems in SAL," SRI International, Tech. Rep. NASA/CR-2002-211858, 2004.

[7]G. Smith and L. Wildman. (2005). "Model checking z specifications using SAL." In *Formal Specification and Development*, Eds. Z and B. Springer. 85–103.

[8]D. C. McFarlane and K. A. Latorella. (2002). The scope and importance of human interruption in human-computer interaction design. *Human- Computer Interaction, 17*(1), 1–61.

This work was supported by the grant "Scenario-Based Verification and Validation of Autonomy and Authority" from the NASA Ames Research Center under award number NNX13AB71A.