AFRL-RI-RS-TR-2017-042

PERPETUAL MODEL VALIDATION

MARCH 2017

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* March 2017	**2. REPORT TYPE** FINAL TECHNICAL REPORT	**3. DATES COVERED** *(From - To)* OCT 2014 – SEP 2016

4. TITLE AND SUBTITLE PERPETUAL MODEL VALIDATION	**5a. CONTRACT NUMBER** IN HOUSE –R1AK
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER 61102F
6. AUTHOR(S) Steven Drager, Stanley Bak, Matthew Anderson	**5d. PROJECT NUMBER** T1PM
	5e. TASK NUMBER IN
	5f. WORK UNIT NUMBER HO
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/Information Directorate Rome Research Site/RITA 525 Brooks Road Rome NY 13441-4505	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/Information Directorate Rome Research Site/RITA 525 Brooks Road Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TR-2017-042

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2017-0755
Date Cleared: 27 Feb 2017

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This research effort investigated fundamental techniques to provide perpetual model validation, where design-time models are validated continuously during runtime. The research considered two fronts: validation of the software model, and validation of the model of the system interacting with the physical world. Since the field of runtime verification has extensively focused on the challenge of runtime model validation in software using direct models, the approach employed here instead considered using indirect models of software execution, for example memory access patterns, to check for security intrusions. Additional research was performed to tackle the essential problem of model validation for systems which contain interactions with the physical world using hybrid automata models. Perpetual model validation will ensure that the actual behavior of the system conforms to the analysis model, raising the level of confidence that can be placed in the results due to formal nature of the analysis.

15. SUBJECT TERMS
Software model validation, run time verification, formal analysis of computer systems, indirect models of software execution, hybrid automata models of cyberphysical systems

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** STEVEN L. DRAGER
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	29	**19b. TELEPHONE NUMBER** *(Include area code)* N/A

Table of Contents

Table of Figures

1 - SUMMARY

Formal analysis of computer systems ultimately relies upon accurate mathematical models. When systems are software based, models can be created based on software semantics and the underlying platform. When systems involve the physical world, a model identification step may be performed.

A deviation of system behavior from the model indicates that behaviors may occur which were not anticipated during system design and analysis. Such deviations may occur if the model identification step was done incorrectly, if simplifications (such as linearization of nonlinear components) are oversimplifications of actual behavior, if the environment at design time varies from the environment at deployment, or if wear and tear of components changes their behavior. In software, model mismatches may be indicative of security flaws in the implementation being exploited to produce unintended behavior. In any case, the results of any earlier formal analysis do not apply when the model is not an accurate representation of the system.

This research effort investigated fundamental techniques to provide *perpetual model validation*, where design-time models are validated continuously during runtime. The research considered two fronts: validation of the software model, and validation of the model of the system interacting with the physical world. Since the field of runtime verification has extensively focused on the challenge of runtime model validation in software using direct models, the approach employed here instead considered using indirect models of software execution, for example memory access patterns, to check for security intrusions. Additional research was performed to tackle the essential problem of model validation for systems which contain interactions with the physical world using hybrid automata models. Perpetual model validation will ensure that the actual behavior of the system conforms to the analysis model, raising the level of confidence that can be placed in the results of formal analysis.

2 - INTRODUCTION

A growing percentage of systems across a variety of domains are comprised of Cyber-physical systems (CPS). Often these systems belong to such domains as vehicles, power plants, and medical devices whose security and reliability are critical to the safety and well-being of their users. In the past such systems were generally considered to be relatively safe from malicious adversaries because they were isolated from their environments and running specialized software on unique hardware platforms. However, this is becoming less and less the case, especially with regard to connectivity, where Cyber-physical systems are increasingly interacting with their environment through short and long range wireless media.

2.1 – THE CHALLENGE FOR THE AIR FORCE

The mission of the Air Force is to "fly, fight and win in air, space and cyberspace". Software is a key contributor to meeting the requirements necessary to fulfill the Air Force mission. "Simply stated, absent secure and resilient software at the core of our cyber defenses, the nation's critical infrastructure is at risk" [1]. Software controls an increasing number of mission critical systems ranging from planning to weapon systems. Since software failures can cause extensive damage resulting in loss of productivity, loss of mission capability, loss of assets, and loss of life, it is important that these systems are developed and verified to be correct. Software for mission assured systems should consist of several attributes including but not limited to correctness, security, safety, resilience, availability, performance and reliability. The correctness of software can be increased with a design environment that allows for the modeling, early analysis, and synthesis of software.

General purpose software development has demonstrated that, with almost certainty, software bugs will be present in the resultant code. For this reason, formal methods must play a larger role in future Air Force software systems. Safety verification, building-in security from the earliest design steps, and other early analysis approaches all require models to be made which capture the expected behavior of the system.

Formal analysis of computer systems ultimately relies upon accurate mathematical models. When systems are software based, models can be created based on software semantics and the underlying platform. When systems involve the physical world, a model identification step may be performed. A deviation of system behavior from the model indicates that behaviors may occur which were not anticipated during system design and analysis. Such deviations may occur if the model identification step was done incorrectly, if simplifications (such as linearization of nonlinear components) are oversimplifications of actual behavior, if the environment at design time varies from the environment at deployment, or if wear and tear of components changes their behavior. In software, model mismatches may be indicative of security flaws in the implementation being exploited to produce unintended behavior. In any case, the results of any earlier formal analysis do not apply when the model is not an accurate representation of the system.

Autonomy in systems, provides the Big Challenge as we see it, and reason why formal methods must have an end-to-end role in our system development, beginning at the earliest of system incarnation. Figure 1 shows Cyber Resilience along the x-axis and Trust on the y-axis, with the goal being a system which is highly trusted and resilient against even zero-day attacks. Let us define that Trust represents the users' belief in the reliability and effectiveness of the system as measured by its correctness and security. In other words, Trust and Cyber Resilience are the ability of the CPS to continue to operate and maintain mission essential functions while coping with on-going cyber-attacks or system failures. To achieve these goals, we have been pursuing development of techniques, methodologies and tools to enable trust and resilience (as measured by correctness, security, reliability, predictability, and survivability) and migrate the analysis from execution (testing and monitoring) to design (correct and formal/security specifications) and development (composition and auto-generation).

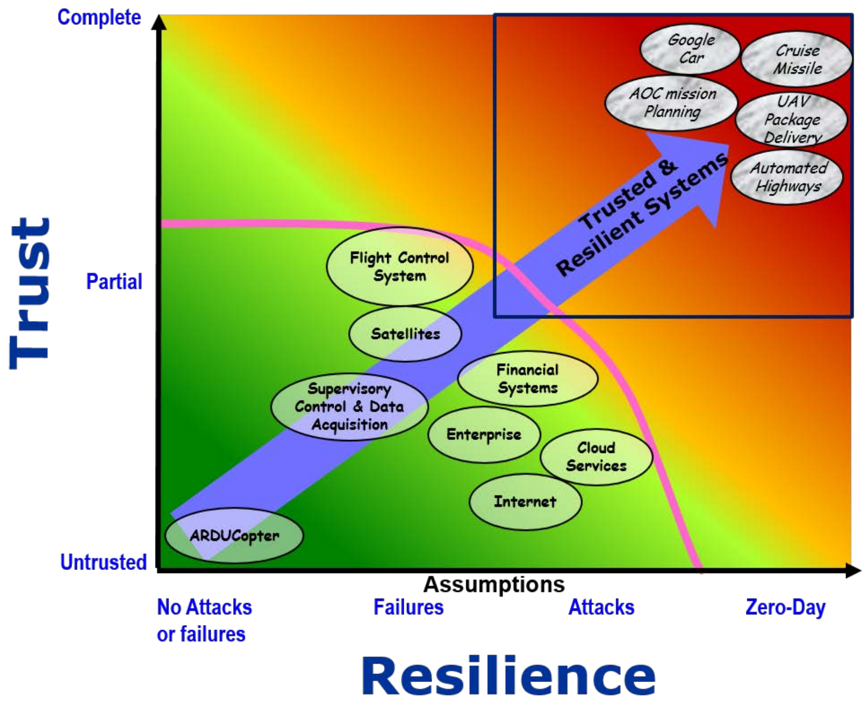


Figure 1: Towards Trusted and Resilient Systems.

2.2 – Objective of this Effort

Perpetual model validation enhances and complements these formal method techniques, by checking that the expected model reflects, as far as one can tell, the behavior of the deployed system. If a violation is detected, it does not necessarily mean that the formal properties are wrong and will be violated, only that their proof is inadmissible. This in itself is extremely valuable information, and inferring it before stressing the system can prevent unexpected violations of safety and security. This research project aims to increase confidence in the correctness of the models, and therefore increase confidence in the correctness of the resultant formal guarantees.

This research effort investigated fundamental techniques to provide perpetual model validation, where design-time models are validated continuously during runtime. The research considered two fronts: validation of the software model, and validation of the model of the system interacting with the physical world. Since the field of runtime verification has extensively focused on the challenge of runtime model validation in software using direct models, the approach employed instead considered using indirect models of software execution, for example memory access patterns, to check for security intrusions. Additional research was performed to tackle essential problems of model validation for systems which contain interactions with the physical world, using hybrid automata models. Perpetual model validation seeks to ensure that the actual behavior of the system conforms to the analysis model, raising the level of confidence that can be placed in the results of formal analysis.

Models of computer systems enable a myriad of formal analysis and verified design approaches. Sound application of formal methods can guarantee with certainty properties regarding the behavior of models of software systems and the way in which they will interact with models of their environment. However, any formal guarantees proven about the system are inapplicable when the model used for the proofs does not correspond to the deployed system. This research effort, therefore, sought to investigate fundamental techniques to increase the confidence that a system's analysis-time model corresponds to the deployed system.

Significant research efforts have been underway at the Air Force Research Laboratory (AFRL) to encourage the use of formal methods and model-based design in software development. The "Correct-by-Construction Software for Embedded Multi-core Systems" (CxC SEMS) is one effort where provably correct models of reactive systems are used to generate multicore implementation code, and then another step will be used to prove the generated code corresponds to the initial model. This second step is essential since a divergence between code and model invalidates any guarantees, which is also the motivation behind perpetual model validation. Another effort at AFRL is the Office of Secretary of Defense (OSD) sponsored, "Techniques and Tools for Trustworthy Composition of Pre-Designed Embedded Software Components". In this work, software components' indirect assumptions, for example timeliness, are exported along with the components to enable more dependable integration. Perpetual model validation could be compatible with this work by, for example, monitoring these indirect assumptions at runtime to validate that they conform to the specification. Lastly, the Air Force Office of Scientific Research (AFOSR) Laboratory Research Initiation Request (LRIR) "Design and Analysis of Trustworthy Software" is making use of domain-specific models to perform upfront analysis of large software systems. After early analysis of models, trusted code generation creates software which conforms to the models, and artifacts are generated to help with testing and deployment. These artifacts could potentially include elements of perpetual model validation to check that non-software components conform to their analysis models. For example, software timeliness ultimately depends not only on the software itself, but also on a host of other factors including the underlying CPU, memory hierarchy, cache interference from other programs running in the system, the effectiveness of speculative prefetch and branch predictors, and the effects of memory-access reordering in the DRAM controller. Perpetual model validation can monitor factors like timeliness, to make sure the underlying system meets requirements from the models.

This research effort contributes toward the science needed to build safe and secure systems. It is important for the success of the Air Force to overcome current approaches which consider security in the late stages of the development of a system, and then with formulations of security incapable of providing comprehensive guarantees. The result is a proliferation of attacks. One of the Essential Focus Areas for the Air Force is Cyber Resilience, whereby systems would be able to withstand attacks. A superior approach is to prevent many of those attacks. Research into perpetual model validation contributes toward that goal.

2.3 - Review of State-of-the-Art

Runtime verification [2, 3] methods employ monitors which check a program's behavior against a model of the expected behavior. For example, an API may require functions are called in a particular order, or software can require a resource is acquired and released in a particular function. In classic runtime verification, monitor code is inserted into the original program, and properties are checked at runtime. This classical checking approach uses *direct models*. Direct model validation need not be limited to the CPU; for example one runtime verification approach monitors PCI bus transactions to make sure that registers being read and written on an I/O peripheral satisfy a device interface model [4].

In perpetual model validation, the proposed research will instead focus on indirect models, which capture aspects that arise as a result of the implementation (side channels). While side channels are typically mentioned in the context of attacking cryptography systems by measuring timing [5] or power fluctuations [6], they are employed here instead as a defense mechanism against code injection. For example, the proposed memory access patterns are not something directly specified by the software engineer, but instead arise as byproducts of the implementation. In earlier work, this technique has been shown as sound, by monitoring timing channels of real-time systems to detect intrusions [7]. Other work has successfully shown that monitoring things like chains of system calls can differentiate normal system behavior from that of malware [8].

In terms of systems containing physical aspects, formal control theoretic approaches exist for model-based fault diagnosis for purely continuous systems [9, 10]. This was generally done for analytical redundancy, that is, to use software to detect when a sensor or actuator has failed. In the proposed research, however, models will be used of combined software / hardware systems represented as hybrid automata which also include discrete system states not considered with the control theoretic approaches.

Model checking frameworks such as Maude [11] can be used to compute reachability and check safety specifications for deterministic and nondeterministic discrete systems, without continuous states. Extensions of Maude such as Real-Time Maude [12] and HI-Maude [13] allow continuous states to be expressed and analyzed within the Maude engine, although complete analysis is limited to decidable classes of hybrid automata (with timed or rectangular dynamics). Systems with more complicated, nonlinear dynamics, can be sampled through time, although this strategy is generally not sound (it may “verify” a model which actually can reach error states). In the proposed approach and research, the reachable set of states for nonlinear hybrid automata is over approximated, preserving soundness at the cost of potential false positives.

Other researchers have also recently considered performing online model checking for hybrid systems [14, 15]. The proposed approach was used to verify a parameterized system by constructing a new hybrid automaton online for the current instance of the problem, and the purpose was to drive a supervisory controller. This approach, therefore, was susceptible to errors in the model, just as offline analysis approaches. The proposed research instead strives to validate the system model, so it is compatible with this earlier work. From a theoretical standpoint, the computation considered was one of safety (there was an explicit unsafe state) rather than reachability, and only linear hybrid systems were allowed. Furthermore, the reachability algorithm was not considered in the research (instead it was treated like a black box), so its worst-case performance was not established. In the proposed research, in contrast, systems will be considered with more general, nonlinear dynamics, and a deadline-aware algorithm to do the online reachability computation will be investigated for the purpose of perpetual model validation.

Another approach which considers online hybrid systems model checking does it because a medical device safeguard system is considered [16] where the human body is part of the system. They cannot, therefore, construct an offline model with enough accuracy and instead, use online analysis to infer a model from runtime data. Then, based on the constructed model they drive supervisory control logic. Rather than providing timeliness guarantees for the online checks, the authors measure how often the online check fails to meet deadlines and reason about the possible consequences. The accuracy of the model is considered (since it was constructed online), which is similar in spirit to the idea of perpetual model validation. Here, a probabilistic argument is made about how parameters in the model may be wrong where, again, the consequences of the model inaccuracies are reasoned about. It is argued that medical safeguards commonly strive to reduce the chance of unsafe states, so a probabilistic guarantee is still useful for the considered system.

3 - METHODS, ASSUMPTIONS AND PROCEDURES

Cyber-physical systems are increasingly interacting with their environment through short and long range wireless media. Our research focuses on checking the accuracy of continuous models at runtime. Applying formal analysis techniques to CPS in such a way requires accurate mathematical models and can allow for the detection of attacks or degraded performance. Our approach to this is to validate the software and that software's interaction with its environment through hybrid automata based modeling.

The "big picture" of the proposed research: as described in the task narrative, this research can address the crucial challenge of increasing the trustworthiness of formal approaches to system design. The research directions proposed are motivated by experience with designing predictable systems and formal design methods, summarized as follows:

- Software implementations contain deterministic side channels which can be used as indirect models of their execution. By measuring last level cache misses on SPEC2006 benchmarks, one can see clear differences in memory access patterns among different benchmark programs.

- Memory access patterns can be modified without significantly affecting observable program behavior. By prefetching memory at the application level, memory access can be clustered into specific regions in the code. For several benchmarks in the MiBench benchmark suite [17], this clustering did not result in increased execution time or total memory usage.

- Current hybrid automata analysis techniques allow only limited analysis of physical systems with nonlinear dynamics. By improving on earlier-developed approaches, it is possible to analyze nonlinear hybrid automata in a small number of dimensions, or alternatively over a short time horizon.

The ultimate vision is a system where formal methods are used up-front, at system design time, to provide guarantees of safety and security. Perpetual model validation is employed at runtime on the same models that were used to do the offline verification step, increasing confidence in the correctness of the models, and therefore the correctness of the resultant guarantees. Based on the observations above, the plan is to develop techniques to validate design-time models during system execution. Specifically, this research will focus effort in two areas:

1) Research and develop the theory and techniques to monitor indirect models of software, initially on memory access patterns. A monitoring module will check, at runtime that the observed memory access pattern matches the pattern the software is expected to produce, which has been previously shown to vary significantly across programs. This means that a successful attack needs to not only exploit the system, but also have an identical memory access pattern. By using earlier results on reshaping memory access, the proposed approach can create a system-specific memory profile, meaning that an attacker needs to individually craft each attack to the system being exploited; the same exploit code cannot be used to infect multiple systems.

2) Monitor assumptions of software which interacts with the physical world by using hybrid automata models. By periodically sampling the system, this proposed approach can leverage earlier results on nonlinear reachability in hybrid automata to perform a time-bounded reachability computation to check if the observed state is reachable in the model from the previously sampled state. If not, the assumed model of the system is incorrect, and any guarantees proven are not applicable to the deployed system.

4 - Results and Discussion

This section presents the results of the investigation into fundamental techniques for providing perpetual model validation, that is validation to ensure that the actual behavior of the system conforms to the analysis model, raising the level of confidence that can be placed in the results of formal analysis, and where design-time models are validated continuously during runtime using both techniques for validation of the software model and validation of the model of the system interacting with the physical world.

4.1 – Year One

As the field of runtime verification has extensively focused on the challenge of runtime model validation in software using direct models, the approach employed considered using indirect models of software execution, for example memory access patterns, to check for security intrusions. Additional research was performed on essential problems of model validation for systems which contain interactions with the physical world, using hybrid automata models. Further details are presented below under the two research areas: (1) model validation through improved analysis of hybrid systems reachability, and (2) model validation using analysis of indirect models of computation.

4.1.1 - Model Validation using Hybrid Systems Reachability:

Accomplishments:

A technique to deal with reducing error during the continuous successor operation in hybrid automata reachability algorithms was developed. A reduction in this error can result in better accuracy and lower computation time. For perpetual model validation, quick computation of the continuous successor operation is necessary. Results are presented in the Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (CyPhy) 2014 [18] and the ACM International Conference on Hybrid Systems: Computation and Control (HSCC) 2014 [19].

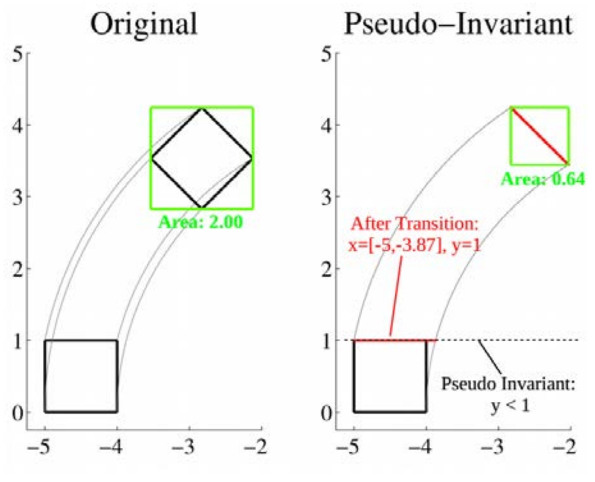
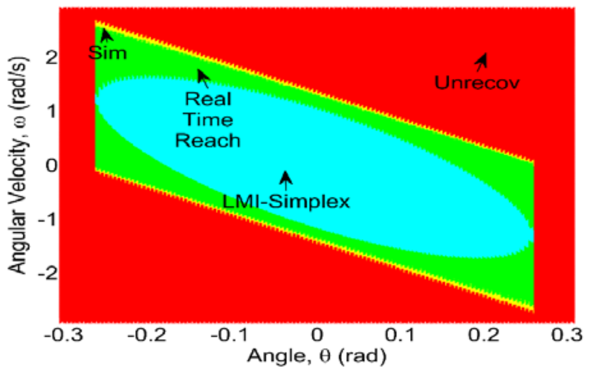
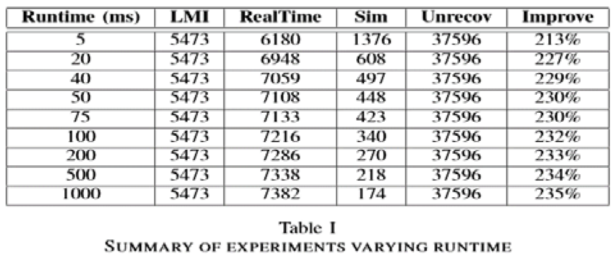


Figure 2: Wrapper Effect reduction with Pseudo-Invariants [19].

Figure 2 illustrates at a high level what the application of this approach of extracting and using Pseudo-invariants can achieve when computing the reachable next states of a harmonic oscillator. The left side shows the reachability approximated through the standard computational means while the right employs pseudo-invariants. The comparative area of the resulting approximation is noticeably reduced, demonstrating a significant decrease in the wrapping error introduced through over-approximation. Further details are presented in the full paper [19].

A key enabling technology for perpetual model validation, the ability to compute hybrid systems reachability at runtime, was developed. The first Anytime algorithm for reachability was proposed where runtime could be traded off for accuracy. Results were presented at the 2014 Safe & Secure Systems & Software Symposium (S5) [20] and the IEEE Real-Time Systems Symposium (RTSS) 2014 [21].





Runtime (ms)	LMI	RealTime	Sim	Unrecov	Improve
5	5473	6180	1376	37596	213%
20	5473	6948	608	37596	227%
40	5473	7059	497	37596	229%
50	5473	7108	448	37596	230%
75	5473	7133	423	37596	230%
100	5473	7216	340	37596	232%
200	5473	7286	270	37596	233%
500	5473	7338	218	37596	234%
1000	5473	7382	174	37596	235%

Table I
SUMMARY OF EXPERIMENTS VARYING RUNTIME

Figure 3: Comparative improvement in State-space [21].

The algorithm allowed for a large improvement in the manageable state-spaces for small run-times. Figure 3 shows the results of the work published in [20, 21] which demonstrates an over 200% improvement for short (~5-10ms) run times and continues to increase the longer it is able to run.

Additionally, Professor Taylor Johnson's summer faculty project, "Inferring Physical System Specifications from Embedded Software Tests," addressed the question of how systems can be verified without a formal specification. In perpetual model validation, some form of model is necessary. By using the developed approach, a model may be derived from initial tests or simulations, which will then be formally checked at runtime. A divergence in this model would imply that the runtime behavior is somehow different from any of the tests, which should be investigated by a system designer.

Further Research

The real-time reachability results need to be evaluated on larger systems (more continuous variables), as well as be able to address the possibility of a change on model during system execution. L1 adaptive control allows for this despite limit changes in a model, and at each time step produces a model estimate which could be used by model validation approaches and will be integrated.

Real-time reachability, given a fixed model, can check if the runtime sensor readings are reachable from the previous step of the model, which is similar to sensor-spoofing. A simple remotecontrolled car and indoor localization system are working and a formal model of the vehicle is being identified. After this step, the real-time reachability algorithm will be used during runtime to detect when the localization system tries to guide the car towards an obstacle.

4.1.2 - Model Validation using Analysis of Indirect Models of Computation

Accomplishments:

A Cooperative Research and Development Agreement (CRADA) with the University of Illinois at Urbana-Champaign (UIUC) was established to build on this basic research project and leverage their earlier research on the Predictable Execution Model (PREM) for real-time system computation. This model allows for the division of the computation into memory and execution phases. Light-PREM is UIUC research towards automated code refactoring, which was presented at the 20th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) 2014 [22]. "Light-PREM: Automated Software Refactoring for Predictable Execution on COTS Embedded Systems," used runtime testing to automatically determine which memory was accessed by a function which could then be perfected in the memory phases.

Further Research

The current automated memory refactoring is good; however, it relies on observed memory accesses. Due to variations in run-time addressing, this can, in some cases cause SEGFAULTs. This has been mitigated by capturing the SEGFAULTs and restoring the execution state. To achieve better efficiency, application of formal techniques for enumerating the possible memory access using concolic execution should be implemented.

The MadT tool, developed by Marco Cesati at the University of Rome (also known as University of Rome Tor Vergata), should be investigated, as this tool uses the OS to exactly detect which memory addresses a program touches and it is expected that a more accurate and symbolic memory-address identification technique, compared with PREM-light, will result.

The longer term step would be to create patterns of memory access inside of specific programs to be detected by the OS as a side-channel as part of perpetual model validation.

4.2 – Year Two

The research continued to focus on fundamental issues for performing validation using *indirect* models of software execution, for example the behavior of physical systems which are coupled with the software in a cyber-physical system. Mismatches in the predicted model and observed model could indicate security intrusions or other critical problems. The research this year focused primarily on systems formalized using hybrid automata models. Roughly speaking, there were two research thrusts: (1) providing improved techniques to perform hybrid systems reachability analysis, and (2) applying the developed techniques to perform system validation.

4.2.1 - IMPROVED TECHNIQUES FOR HYBRID SYSTEMS REACHABILITY:

Accomplishments:

A key enabling technology for perpetual model validation, the ability to compute hybrid systems reachability at runtime, was developed. The first Anytime algorithm for reachability was proposed where runtime could be traded off for accuracy. Results were published at RTSS 2014, and a journal extension of this work which evaluated the approach on a nonlinear model as well as its performance on embedded hardware was accepted for publication in the journal Association for Computing Machinery (ACM) Transactions on Embedded Computing Systems (TECS) [23].

Professor Taylor Johnson's held a summer faculty research position and his project, "Inferring Physical System Specifications from Embedded Software Tests," addressed the question of how systems can be verified without a formal specification. The results of his project on automatic model synthesis was published in the International Conference on Cyber-Physical Systems (ICCPS) 2015 [24, 25].

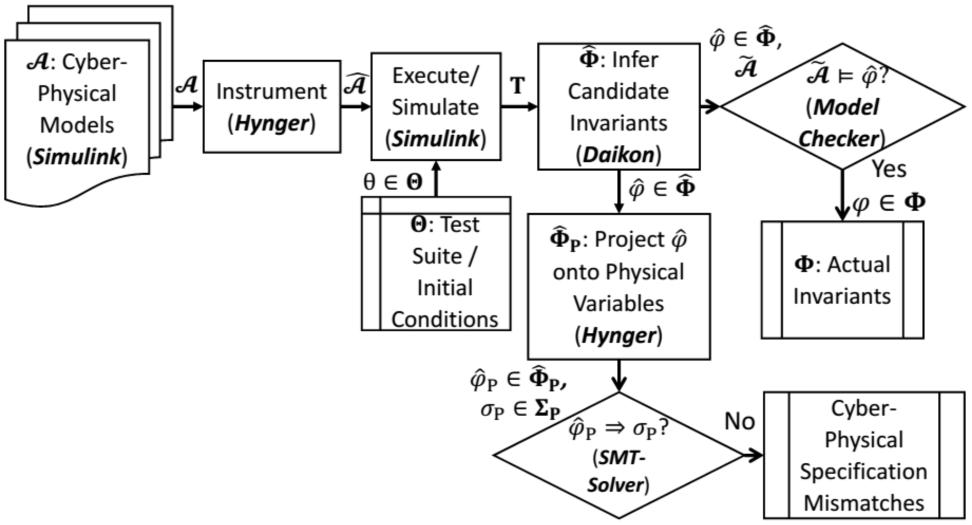

$\mathcal{A}$: Cyber-Physical Models (*Simulink*)
$\mathcal{A}$
Instrument (*Hynger*)
$\widehat{\mathcal{A}}$
Execute/ Simulate (*Simulink*)
T
$\widehat{\Phi}$: Infer Candidate Invariants (*Daikon*)
$\hat{\varphi} \in \widehat{\Phi}$, $\widetilde{\mathcal{A}}$
$\widetilde{\mathcal{A}} \models \hat{\varphi}$? (*Model Checker*)
Yes
$\varphi \in \Phi$
Φ: Actual Invariants
$\theta \in \Theta$
Θ: Test Suite / Initial Conditions
$\hat{\varphi} \in \widehat{\Phi}$
$\widehat{\Phi}_P$: Project $\hat{\varphi}$ onto Physical Variables (*Hynger*)
$\hat{\varphi}_P \in \widehat{\Phi}_P$, $\sigma_P \in \Sigma_P$
$\hat{\varphi}_P \Rightarrow \sigma_P$? (*SMT-Solver*)
No
Cyber-Physical Specification Mismatches

Figure 4: Hynger Overview [24, 25].

This inference process is carried out by a tool referred to as Hynger (Hybrid iNvariant GEneratoR), overviewed in Figure 4, which is a MATLAB and Java based tool that accepts Simulink/Stateflow models as input which it then simulates and observes in order to generate a set of candidate invariants.

A tool for performing hybrid automaton model transformations, Hyst, was developed and published as a Tools paper in HSCC 2015 [26]. It was also showcased in the poster / demo session of CPS week 2015. Results on using pseudo-invariants from last year were automated into a Hyst model transformation pass, increasing the ease of applicability of earlier results. Figure 5 illustrates the flow and the formats supported by the Hyst tool. It was designed with the intent of being extensible, through the intermediate format, to potentially support other sources and tools in the future.

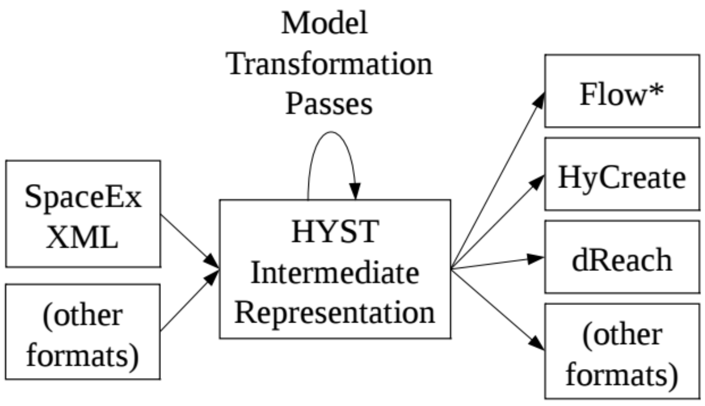


Figure 5: Hyst Model Transformation Toolflow [26].

Further Research

The Hyst model transformation tool will continue to be developed with other passes, such as hybridization, which would enable nonlinear hybrid automata to be analyzed by tools which only handle linear dynamics. Furthermore, the automated running of tools and automated tool chaining is planned to make analysis both more scalable, as well as easier to apply for users of the analysis tools.

Models of cyber-physical systems often include tight control loops. Reachability tools scale poorly on such models, due to a large number of discrete transitions which occur whenever the controller is run. However, these system are of critical importance and require development of approaches which permit their formal analysis by creating continuous abstractions of the periodically-actuated systems, which are likely to scale better for analysis.

4.2.2 - SYSTEM VALIDATION USING THE DEVELOPED TECHNIQUES:

Accomplishments:

Runtime results of reachability can be used to provide necessary and sufficient conditions for safety to a distributed cyber-physical system where the communication is unreliable was demonstrated in the ACM TECS [27] journal paper. Additionally, progress guarantees were possible, under the assumption that communication eventually gets through.

Based off of the CRADA established with the University of Illinois at Urbana-Champaign, the developed real-time reachability algorithm was applied towards detecting sensor spoofing. A remote-control car testbed in UIUC’s lab was utilized with faked sensor readings in Matlab to demonstrate that the technique, which made use of real-time reachability, detected the model violations.

Further Research

Extend current results by utilizing the developed real-time reachability technique to do the run-time reachability computation. This would enable the analysis of more complicated dynamics compared with the existing approach, which requires solutions to the differential equations be provided.

4.3 – YEAR THREE

The last year of the research project focused primarily on ways to make formal methods for hybrid systems reasoning 1) more automated and 2) applicable to a broader application space.

4.3.1 - IMPROVED AUTOMATION TECHNIQUES:

In terms of automation, one key contribution was the Hypy tool. Hypy is a set of python libraries which automate the previously manual task of converting models, running analysis tools, and interpreting tool results. This allows powerful high-level analysis approaches where tools are run multiple times and models and tool parameters can be tuned between each run. The applicability of Hypy in automated parameter tuning, automatic model modification (via the previously manual method of pseudo-invariants) was shown to improve verification accuracy, and iterative abstraction refinement methods for hybrid systems. The audience at Applied Verification for Continuous and Hybrid Systems (ARCH) 2016 [28] voted Hypy the Best Tool Award. Another contribution for automation is a time-triggered method for performing static hybridization (multi-domain linearization). This methodology was shown to scale significantly better than existing approaches which perform a state-based division of the variables. The computational component for this work was written with Hypy, and this paper was selected for the Best Repeatability Package at 19th ACM International Conference on Hybrid Systems: Computation and Control 2016 [29] from approximately 20 entries.

4.3.2 - BROADENED APPLICATION SPACE:

While the formal analysis methods researched under this effort have traditionally been used to find bugs in existing system designs, they also open up new applications which expand the impact of the research results. Over the last year, a journal paper in Quantum Information Processing that compares traditional parallel solvers for graph-theoretic NP-complete problems against both quantum computing approaches and against formal methods tools (SMT solvers) has been published. The SMT-based verification tools in this case solved the largest problems in the least time. Additionally, this work examined conditions on when a cyber-physical system which requires continuous actuation could be safely restarted (which disconnects the controller for a bounded amount of time). This serves as a novel robustness mechanism, by keeping the system state in a region where resetting is possible, the system can tolerate faults that can be resolved through a full system restart (which generally works well for traditional, software-only systems). Further details may be found in the presentation at the 21st IEEE Conference on Emerging Technologies and Factory Automation (ETFA) 2016 [30]. The research effort also looked at the applications of hybrid systems reachability in order to detect model mismatches. This can be used for quickly finding certain kinds of security intrusions, as well as detecting when the physical model of a system, due to physical deterioration for example, no longer corresponds to the model used during verification time. Finally, the research looked at ways to combine hybrid systems reasoning with software model checking. Through the use of a new object, contract automaton, results from the two domains could be soundly combined to enable end-to-end verification for a cyber-physical system, and is further described in ACM Special Interest Group on Embedded

Systems (SIGBED) International Conference on Embedded Software (EMSOFT) 2016 [31].

Further Research

Current research directions include, improving Hyst to allow for model specifications that include lookup tables, time delays and stochastic analysis; detecting deviations from predicted memory access profiles; and use of real-time reachability as a detection mechanism for runtime model mismatch.

5 - Conclusions

With the Air Force's increasing demand for mission-critical functionality, the development of correct software systems is fundamental to mission assurance. The current way software systems are developed and maintained often produces brittle, out of date, and vulnerable systems. The systems that the Air Force relies on need to be predictable, dependable and resilient. The software should improve mission assurance by providing a fight through capability that might degrade performance but still accomplish the mission objectives. Modernization of software systems to patch newly discovered flaws or hardware updates should be rapid and increase the robustness of the system, not make the system more brittle.

General purpose software development has demonstrated that, with almost certainty, software bugs will be present in the resultant code. For this reason, formal methods must play a larger role in future Air Force software systems. Safety verification, security-up-front design, and other early analysis approaches all require models to be made which capture the expected behavior of the system. Perpetual model validation enhances and complements these formal methods techniques, by checking that the expected model reflects, as far as one can tell, the behavior of the deployed system. If a violation is detected, it does not necessarily mean that the formal properties are wrong and will be violated, only that their proof is inadmissible. This in itself is extremely valuable information, and inferring it before stressing the system can prevent unexpected violations of safety and security. The proposed research will increase confidence in the correctness of the models, and therefore increase confidence in the correctness of the resultant formal guarantees.

Formal design of the software cannot only benefit from perpetual model validation using indirect models, but as the space and air domains have a physical aspect to them, they may be modeled using hybrid automata in order to validate assumptions the software makes about the physical components, and the assumptions the physical components make about the software.

This research contributes toward the science needed to build safe and secure systems. It is important for the success of the Air Force to overcome current approaches which consider security in the late stages of the development of a system, and then with formulations of security incapable of providing comprehensive guarantees. The result is a proliferation of attacks. One of the Essential Focus Areas for the Air Force is Cyber Resilience, whereby systems would be able to withstand attacks. A superior approach is to prevent many of those attacks. Research into perpetual model validation contributes toward that goal.

The research progressed along a number of areas centered around the use of formal methods and hybrid automata models for the run-time analysis and verification of Cyber Physical Systems. Improved methods for Run-Time Assurance have been tested, frameworks for model transformation and generation created, and Real-Time scheduling techniques adapted to incorporate security constraints.

The research has focused on checking the accuracy of continuous models at runtime through applying formal analysis techniques to CPS in such a way requiring accurate mathematical models which allows for the detection of attacks or degraded performance. This approach thus validates the software and that software's interaction with its environment through hybrid automata based modeling.

With the Air Force's increasing demand for mission-critical functionality, the development of correct software systems is fundamental to mission assurance. The current way software systems are developed and maintained often produces brittle, out of date, and vulnerable systems. The systems that the Air Force relies on need to be predictable, dependable and resilient. The software should improve mission assurance by providing a fight through capability that might degrade performance but still accomplish the mission objectives. Modernization of software systems to patch newly discovered flaws or hardware updates should be rapid and increase the robustness of the system, not make the system more brittle.

5.1 – WAY AHEAD

Research areas for both new designs and maintenance actions include but are not limited to: Scalable formal methods for establishing trust of resilient systems; Methodologies for complex software design, development, analysis, synthesis, repair, and validation and verification; System composition analytics; Formal Models of composable properties; Trustworthy architectures for system of systems; Modeling, assessment, and vulnerability analysis; Assessment and measurement for end-to-end system analysis; Software comprehension, curation and diagnostics tools; Methodologies to improve understanding of software including the reasons behind design choices; ultimately leading to Modular, automated, interoperable, & affordable systems.

Current approaches to developing resilient systems typically lack a rigorous assessment of trust. Resiliency approaches affect system change in an attempt to fight through failures and attacks but, at most, only hand wave about whether or not these changes should be trusted. How do we know these changes will lead to mission success or mission failure? This project will leverage the Foundations of Trust Program and maturing resiliency research as a testbed to experiment with assessing/reestablishing trust during resilience actions.

There are two thrusts within this project. First is the extension of the calculus of trust to encompass resiliency. Second is extension of maturing resiliency research prototypes with trust and critical experiment and demonstration of its viability.

Correctness, vulnerabilities, bugs and maintainability (legacy software) have been key DoD and industry challenges for as long as software have been developed, deployed and maintained. After years of investment from both the public and private sectors, we are now at a crossroads where the fundamental techniques are stabilizing and research prototypes are being demonstrated on real world systems. Take for example, the DARPA High Assurance Cyber Military Systems (HACMS) program that seeks to use automated tools to either partially or fully synthesize the control software for various Unmanned Platforms. The automatically synthesized code has correctness, safety and security guarantees with respect to the specification. While the mathematical proof provides a high level of trust, there is still room for vulnerabilities and flaws at the specification level. The time and expertise required to use the formal methods tools makes it necessary for any repair to be accomplished at a software depot or prime contractor. The technology developed under the Foundation of Trust program and others promoting the use of formal methods would benefit from the resiliency research.

The later thrust will research two existing resiliency research prototypes GenProg and Formally Generating Adaptive Security Protocols. GenProg is an evolutionary approach that is inherently resilient due to its ability to evolve a solution but is inherently less trusted since it uses a stochastic approach to fight through attacks. Formally Generating Adaptive Security Protocols is a formal approach that is inherently trusted since it uses formal logic to synthesize correct by construction code but is inherently less resilient since it is currently difficult to change the original high level specification to respond/adapt to successful attacks. This project will explore the middle ground between evolutionary and formal approaches to develop trusted and resilient systems.

Resilient software approaches that rely on self-adapting code are currently not trusted by the end user. While this program will explore technological solutions to this problem, it will also have to explore the human issues of trusting self-adapting software. To accomplish this task the program will have multiple directions. One important area that will be pursued is the readability of code generated by the resiliency tools. Trust can be gained if a human can read the repair code and quickly understand and confirm the modified code is correct.

Now is the right time to take on this research challenge. Adaptive response is widely accepted as a necessity to survive and operate through attacks, but system owners and operators (warfighters) are reluctant to use this new technology because it lacks the technological advances required to earn their trust. Leveraging past and ongoing research in trust and resilience presents an opportunity for developing this holistic foundation for trusted resilient systems.

REFERENCES:

[1] Brown. Fortifying Our Cyber Defenses. CrossTalk, Vol.22 No.6 Sept/Oct 2009.

[2] K. Havelund and G. Rosu, “Synthesizing monitors for safety properties,” in In Tools and Algorithms for Construction and Analysis of Systems (TACAS 02), 2002, pp. 342–356.

[3] F. Chen and G. Rosu, “Mop: an efficient and generic runtime verification framework,” SIGPLAN Not., vol. 42, no. 10, Oct. 2007.

[4] R. Pellizzoni, P. Meredith, M. Caccamo, and G. Rosu, “Hardware runtime monitoring for dependable COTS-based real-time embedded systems,” in Real-Time Systems Symposium, 2008.

[5] P. C. Kocher, “Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems.” Springer-Verlag, 1996, pp. 104–113.

[6] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis.”, Springer-Verlag, 1999, pp. 388–397.

[7] S. Mohan, S. Bak, E. Betti, H. Yun, L. Sha, and M. Caccamo, “S3A: Secure System Simplex Architecture for Enhanced Security and Robustness of Cyber-Physical Systems,” in Proceedings of the 2nd International Conference on High Confidence Networked Systems (HiCoNS), 2013.

[8] S. A. Hofmeyr, S. Forrest, and A. Somayaji, “Intrusion detection using sequences of system calls,” Journal of Computer Security, vol. 6, pp. 151–180, 1998.

[9] A. Edelmayer, J. Bokor, and L. Keviczky, “An h infin; filtering approach to robust detection of failures in dynamical systems,” in Decision and Control, 1994.

[10] S. Ding, Model-Based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools. Springer-Verlag Berlin Heidelberg, 2008.

[11] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, Eds., All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic, ser. Lecture Notes in Computer Science, vol. 4350. Springer, 2007.

[12] P. C. Ölveczky and J. Meseguer, “Semantics and pragmatics of Real-Time Maude,” Higher-Order and Symbolic Computation, vol. 20, no. 1-2, pp. 161–196, 2007.

[13] M. Fadlisyah, P. C. Ölveczky, and E. Abraham, “Formal modeling and analysis of human body exposure to extreme heat in HI-Maude,” in Rewriting Logic and Its Applications, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, vol. 7571, pp. 139–161.

[14] L. Bu, Q. Wang, X. Chen, L. Wang, T. Zhang, J. Zhao, and X. Li, “Toward online hybrid systems model checking of cyber-physical systems’ time-bounded short-run behavior,” SIGBED Rev., vol. 8, no. 2, pp. 7–10, Jun. 2011.

[15] L. Bu, X. Chen, L. Wang, and X. Li, “Online verification of control parameter calculations in communication based train control system,” CoRR, vol. abs/1101.4271, 2011.

[16] T. Li, F. Tan, Q. Wang, L. Bu, J.-N. Cao, and X. Liu, “From offline toward real-time: A hybrid systems model checking and cps co-design approach for medical device plug-and-play (MDPnP),” in Cyber-Physical Systems (ICCPS), 2012.

[17] MiBench Version 1.0, University of Michigan at Ann Arbor, 2001, http://www.eecs.umich.edu/mibench/.

[18] Bak, Stanley, “Reducing the Wrapping Effect in Flowpipe Construction using Pseudo-Invariants.” Fourth Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (CyPhy 2014). April 14-17 2014 Berlin, Germany.

[19] Bak, Stanley, “Reducing the Wrapping Effect when Computing Continuous Successors.” 17th International Conference on Hybrid Systems: Computation and Control (HSCC 2014). April 15-17 2014 Berlin, Germany.

[20] Bak, Stanley, Johnson, Taylor, Caccamo, Marco, Sha, Lui, “A Unified Run-Time Assurance Scheme using Real-Time Reachability.” 2014 Safe & Secure Systems & Software Symposium (S5) June 10-12 2014 Dayton, OH.

[21] Bak, Stanley, Johnson, Taylor T., Caccamo, Marco, Sha, Lui, “Real-Time Reachability for Verified Simplex Design.” 35th IEEE Real-Time Systems Symposium (RTSS 2014). December 2-5 2014 Rome, Italy.

[22] Mancuso, Renato, Dudko. Roman, Caccamo, Marco “Light-PREM: Automated Software Refactoring for Predictable Execution on COTS Embedded Systems,” 20th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2014)

[23] T. Johnson, S. Bak, M. Caccamo, L. Sha , "Real-Time Reachability for Verified Simplex Design" (journal version), ACM Transactions on Embedded Computing Systems (TECS)

[24] Johnson, Taylor, Bak, Stanley, Drager, Steven, "Cyber-Physical Specification Mismatch Identification with Dynamic Analysis", 6th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS 2015). April 13-17 2015 Seattle.

[25] Johnson, Taylor, Bak, Stanley, Drager, Steven, “Inferring Physical System Specifications from Embedded Software Tests,” IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS 2015). April 13-17 2015 Seattle.

[26] Bak, Stanley, Bogomolov, Sergiy, Johnson, Taylor T. “HYST: A Source Transformation and Translation Tool for Hybrid Automaton Models,” ACM/IEEE 18th International Conference on Hybrid Systems: Computation and Control (HSCC 2015). April 13-17 2015 Seattle. This work was also presented at the poster session of CPS Week 2015.

[27] S. Bak, F. Abdi, Z. Huang, M. Caccamo, "Safety and Progress for Distributed Cyber-Physical Systems with Unreliable Communication", ACM Transactions on Embedded Computing Systems

(TECS), September 2015

[28] S. Bak, S. Bogomolov, C. Schiling, "High-level Hybrid Systems Analysis with Hypy", Applied Verification for Continuous and Hybrid Systems (ARCH 2016), Best Tool Award

[29] S. Bak, S. Bogomolov, T. Henzinger, T. Johnson, P. Prakash, “Scalable Static Hybridization Methods for Analysis of Nonlinear Systems", 19th International Conference on Hybrid Systems: Computation and Control (HSCC 2016), 49% acceptance rate, Best Repeatability Evaluation Package Award

[30] V. Horan, S. Adachi, S. Bak,"A Comparison of Approaches for Finding Minimum Identifying Codes on Graphs", Quantum Information Processing, 2016

[31] S. Bak and S. Chaki, “Verifying Cyber-Physical Systems by Combining Software Model Checking with Hybrid Systems Reachability”, EMSOFT 2016.

APPENDIX A: PUBLICATIONS FOR THE PROJECT

Yujian Fu, S. Drager, “Modeling & Verification of Humanoid Robot Task Coordination”, Proceedings of 15th IEEE International Symposium on High Assurance Systems Engineering (HASE), January 9 – 11, 2014, Miami, FL, USA.

Bak, Stanley, “Reducing the Wrapping Effect when Computing Continuous Successors.” 17th International Conference on Hybrid Systems: Computation and Control (HSCC 2014). April 15-17 2014 Berlin, Germany.

Bak, Stanley, “Reducing the Wrapping Effect in Flowpipe Construction using Pseudo- Invariants.” Fourth Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (CyPhy 2014). April 14-17 2014 Berlin, Germany.

Bak, Stanley, Johnson, Taylor, Caccamo, Marco, Sha, Lui, “A Unified Run-Time Assurance Scheme using Real-Time Reachability.” 2014 Safe & Secure Systems & Software Symposium (S5) June 10-12 2014 Dayton, OH.

Zalewski, J., Drager, S., McKeever, W., & Kornecki, A. J. (2014). Measuring Security: A Challenge for the Generation. Position papers of the 2014 Federated Conference on Computer Science and Information Systems, September 7–10, 2014, Warsaw, Poland

Bak, Stanley, Johnson, Taylor T., Caccamo, Marco, Sha, Lui, “Real-Time Reachability for Verified Simplex Design.” 35th IEEE Real-Time Systems Symposium (RTSS 2014). December 2-5 2014 Rome, Italy.

Bak, Stanley, Bogomolov, Sergiy, Johnson, Taylor T. “HYST: A Source Transformation and Translation Tool for Hybrid Automaton Models,” ACM/IEEE 18th International Conference on Hybrid Systems: Computation and Control (HSCC 2015). April 13-17 2015 Seattle. This work was also presented at the poster session of CPS Week 2015.

Pellizzoni, Rodolfo, Paryab, Nada, Yoon, Man-Ki, Bak, Stanley, Mohan, Sibin, Bobba, Rakesh B. “A Generalized Model for Preventing Information Leakage in Hard Real-Time Systems,” 21st IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2015). April 13-17 2015 Seattle.

Johnson, Taylor, Bak, Stanley, Drager, Steven, "Cyber-Physical Specification Mismatch Identification with Dynamic Analysis", 6th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS 2015). April 13-17 2015 Seattle.

Johnson, Taylor, Bak, Stanley, Drager, Steven, “Inferring Physical System Specifications from Embedded Software Tests,” IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS 2015). April 13-17 2015 Seattle.

S.Bak, S. Bogomolov, M. Greitschus and T. Johnson, "Benchmark Generator for Stratified Controllers of Tank Networks", Applied Verification for Continuous and Hybrid Systems (ARCH), April 2015

T. Johnson, S. Bak, M. Caccamo, L. Sha , "Real-Time Reachability for Verified Simplex Design" (journal version), ACM Transactions on Embedded Computing Systems (TECS)

S. Bak, F. Abdi, Z. Huang, M. Caccamo, "Safety and Progress for Distributed Cyber-Physical Systems with Unreliable Communication", ACM Transactions on Embedded Computing Systems (TECS), September 2015

Zalewski, Janusz, Drager, Steven, Kornecki, Andrew, Czejdo, Bogdan “Modeling Resiliency and Its Essential Components for Cyberphysical Systems” 2nd International Workshop on Cyberphysical Systems, Lodz, Poland, Sept. 13-16, 2015.

Fu, Yujian, Drager, Steven “Reconfiguration of Autonomous Robotics” International Journal of Robotics Applications and Technology

AJ Kornecki, J Zalewski, “Threat Modeling for Aviation Computer Security”, CrossTalk, Nov 2015

S. Bak, S. Bogomolov and T. Johnson, "Hybrid Systems Model Transformations with HyST", Demo and Poster Session, ACM/IEEE 19th International Conference on Hybrid Systems: Computation and Control (HSCC 2016)

S. Bak, S. Bogomolov, C. Schiling, "High-level Hybrid Systems Analysis with Hypy", Applied Verification for Continuous and Hybrid Systems (ARCH 2016), Best Tool Award

S. Bak, S. Bogomolov, T. Henzinger, T. Johnson, P. Prakash, “Scalable Static Hybridization Methods for Analysis of Nonlinear Systems", 19th International Conference on Hybrid Systems: Computation and Control (HSCC 2016), 49% acceptance rate, Best Repeatability Evaluation Package Award

Janusz Zalewski, Ingrid A Buckley, Bogdan Czejdo, Steven Drager, Andrew J Kornecki, Nary Subramanian, “A Framework for Measuring Security as a System Property in Cyberphysical Systems”, Information, 2016

V. Horan, S. Adachi, S. Bak,"A Comparison of Approaches for Finding Minimum Identifying Codes on Graphs", Quantum Information Processing, 2016

Fardin Abdi Taghi Abad, Stanley Bak, Rohan Tabish, Or Dantsker, Marco Caccamo, “Run-Time Fault Detection for Real-Time Hybrid Systems with Imperfect Models Using Reachability Analysis,” under review, ACM Transactions on. Cyber-Physical Systems (TCPS Journal)

Fardin Abdi Taghi Abad, Stanley Bak, Renato Mancuso, Or Dantsker, and Marco Caccamo, "Reset-Based Recovery for Real-Time Cyber-Physical Systems with Temporal Safety Constraints," under review, Euromicro Conference on Real-Time Systems (ECRTS 2016)

S. Bak and S. Chaki, “Verifying Cyber-Physical Systems by Combining Software Model Checking with Hybrid Systems Reachability”, EMSOFT 2016.

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

ACM	Association for Computing Machinery
AFOSR	Air Force Office of Scientific Research
AFRL	Air Force Research Laboratory
ARCH	Applied Verification for Continuous and Hybrid Systems
CPS	Cyber-physical systems
CRADA	Cooperative Research and Development Agreement
CxC SEMS	Correct-by-Construction Software for Embedded Multi-core Systems
CyPhy	Workshop on Design, Modeling and Evaluation of Cyber Physical Systems
EFTA	EEE Conference on Emerging Technologies and Factory Automation
EMSOFT	ACM SIGBED International Conference on Embedded Software
HACMS	High Assurance Cyber Military Systems
HSCC	ACM International Conference on Hybrid Systems: Computation and Control
Hynger	Hybrid iNvariant Generator
ICCPS	International Conference on Cyber-Physical Systems
LRIR	Laboratory Research Initiation Request
OSD	Office of Secretary of Defense
PREM	Predictable Execution Model
RTCSA	IEEE International Conference on Embedded and Real-Time Computing Systems and Applications
RTSS	IEEE Real-Time Systems Symposium
S5	Safe & Secure Systems & Software Symposium
SIGBED	Special Interest Group on Embedded Systems
TECS	Transactions on Embedded Computing Systems
UIUC	University of Illinois at Urbana-Champaign