DARPA-PA-20-02
DARPA AIE PA
Volume 1: Technical and Management Volume

AIE Opportunity #	DARPA-PA-20-02-01
Proposal Title	SMELLCPS: Symbolic Math Expressions from Low-level Logic in Cyber-physical Systems
Proposer Organization	University of Southern California
Technical Point of Contact (POC)	Name: Luis Garcia Address: University of Southern California Information Sciences Institute 4676 Admiralty Way, Suite 1001 Marina del Rey, CA 90292-6611 Telephone: 310.448.9190 Email: lgarcia@isi.edu
Administrative POC	Name: Sapphire Masterson Address: University of Southern California Dept. of Contracts and Grants 4676 Admiralty Way, Suite 1001 Marina Del Rey, CA 90292-6611 Telephone: 310.448.9161 Email: sapphirm@usc.edu
Date Proposal was Prepared	09/17/2020
Proposal Validity Period (minimum 365 days)	365 days

Table of Contents

1. Proposal Summary

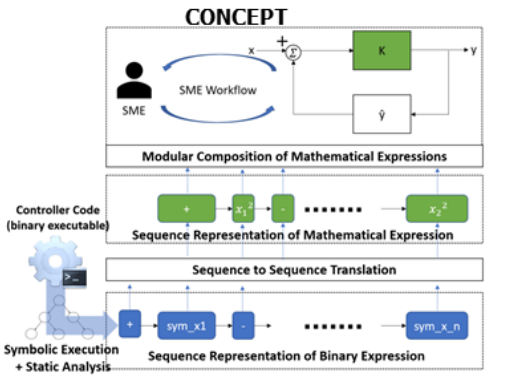
a. Slide

DARPA-PA-20-02 Artificial Intelligence Exploration (AIE)

SMELLCPS: Symbolic Math Expressions from Low-level Logic in Cyber-physical Systems

University of Southern California, Information Sciences Institute; PIs: Luis Garcia, Christophe Hauser

CONCEPT



APPROACH

We will create a unified framework to integrate static analysis and symbolic execution with neural translation as follows:

- Static program analysis and symbolic execution to generate low-level symbolic expressions
- Neural translation to lift the expressions to composed mathematical expressions
- Hierarchical composition of complex mathematical structures from local mathematical expressions based on control-flow and data-flow analyses.
- Semantic alignment and communication of domain-specific expressions to SME

IMPACT

We will study and address the fundamental problem of reconstructing high-level, natural mathematical expressions from their low-level code representation:

- Innovative applications of neural-symbolic approaches to binary analysis.
- Collaborative analysis between system engineers and domain experts.
- Generalization across a broad set of architectures and hardware targets of industrial control systems.

CONTEXT

Symbolic execution representations fail to capture composed mathematical expressions for several reasons:

- State-space of "path" explosion
- End-to-end composition of mathematical expressions spread across several subfunctions or branches
- Readability of symbolic expressions from low-level code

Symbolic execution methods can benefit from neural translation techniques to compose high-level math expressions—which is what we propose.

Source Selection Information

b. Discussion

Our proposal titled “SMELLCPS: Symbolic Math Expressions from Low-level Logic in Cyber-physical Systems” will create a unified framework to address the fundamental problem of reconstructing high-level, natural mathematical expressions from their low-level code representation. SMELLCPS integrates static program analysis and symbolic execution with machine translation techniques to extract domain-specific mathematical expressions from binary code and communicate them to subject matter experts for collaborative analysis.

To demonstrate our framework, we will tackle the problem of scalable and automatic recovery of mathematical expressions within closed-source binaries of safety-critical cyber-physical systems. Traditional symbolic execution frameworks are capable of locally generating mathematical expressions of individual symbolic variables up to a certain code path size or program scope. However, the problem of automatic extraction of mathematical expressions is beyond the scope of standalone symbolic execution frameworks for several reasons, including the state-space or “path” explosion problem, the end-to-end composition of mathematical expressions spread across several subfunctions or program branches, and the readability of symbolic expressions generated from low-level code. Composite mathematical expressions such as polynomials can be simply decomposed and translated to a set of programmatic arithmetic operations. However, the problem of recomposing them from mathematical primitives is a difficult one, requiring careful modeling of operator semantics, some of which may be embedded within intricate control-flow structures (e.g., loops), and intertwined as part of non-associative mathematical expressions. Our approach addresses these challenges through a novel application of binary analysis abstraction and neural translation.

The SMELLCPS framework will deconstruct complex CPS binary code to communicate abstract representations of the underlying control system design to an associated domain expert. We first leverage static program analysis and symbolic execution in order to identify and locate arithmetic operations in binary code and generate low-level symbolic expressions in the form of Abstract Syntax Trees (ASTs) of low-level operations. From there, we will leverage recent advances in neural translation to lift individual symbolic expressions into high-level, natural mathematical expressions. In particular, we will translate sequence representations of a binary’s symbolic expression to a sequence representation of human-readable mathematical expressions. We will generate copious amounts of training data using the infinite set of mathematical expressions to train a machine translation model that identifies local mathematical expressions.

SMELLCPS will then build a hierarchical model of expression composability to recover complex mathematical structure from local mathematical expressions. We will integrate domain-specific concepts using control-flow and data-flow analysis to capture their dependencies and causality of their sequences. We will elucidate the efficacy of using SMELLCPS to analyze workflows of closed-source safety-critical control system design under the guidance of our subject matter expert (SME), Dr. David Barnhart, and his Space Engineering Research Center (SERC). The project will require $990K and will complete in 18 months.

2. Goals and Impact

Key innovations: Our project will lay the foundation for principled synthesis of Artificial Intelligence, Symbolic Program Analysis, and Cyber-physical Modeling methods applied to industrial control system reverse engineering. We propose a hierarchical, neural-symbolic approach rooted in machine translation applied to distributed binary representations at multiple levels of abstraction. In particular, we apply machine translation to map binary-level expressions to higher-level mathematical expressions. We contextualize the mathematical expressions by inferring the higher-level control systems module realized by the mathematical expression. ***Scenario:*** We will study the impact of symbolic mathematical expression extraction for analyzing safety-critical cyber-physical systems. Communicating clear and succinct expressions to domain experts enables collaborative analysis for safety-critical systems such as industrial controllers and drones that are increasingly targeted by nation-state malware. ***Impact:*** If successful, SMELLCPS will bring a wealth of innovations to automated binary program analysis. Qualitatively, SMELLCPS will demonstrate the feasibility and generalizability of extracting mathematical expressions from intricate modular binary expressions. Quantitatively, SMELLCPS will pave the way for end-to-end expression extraction that is generalizable across a wide set of architectures and hardware targets of industrial control systems.

3. Technical Plan

Machine learning approaches have proven successful in a range of binary analysis applications, including malware analysis and vulnerability discovery. However, binary analysis of software deployed in safety-critical cyber-physical systems requires an additional step of extracting relevant information that pertains to the modeling and analysis of the underlying physical system. Extracting high-level domain concepts of algorithmic implementations in cyber-physical software, and in particular mathematical expressions of control systems, enables collaborative analysis between system engineers and domain experts.

Existing approaches use deep learning to identify the existence of algorithms or pieces of code within a larger program--usually in the context of vulnerability detection--using static and dynamic software binary feature patterns. These approaches depend on correct feature extraction and representation of binary features, and we recently taxonomized and evaluated learning representations [4] for binary executable files for security tasks. However, the extraction of fine-grained semantic information, and in particular mathematical expressions, in a way that is digestible by human beings requires more fine-grained and targeted binary program analysis techniques.

3.1. Research Objectives

We will study the application of sequence to sequence translation and deep learning to extract fine-grained semantic information in the form of mathematical expressions from industrial control systems binaries. We will create novel approaches for iteratively mapping learning

representations of binary executable files to mathematical expressions. We will first explore the simplest case of mapping mathematical expressions to isolated algorithms in a binary instead of analyzing binary files as a whole. We will develop a sequence representation to enable the sequence to sequence translation of abstracted software binary representations to mathematical expressions. Specifically, we will explore mapping a symbolic, intermediate representation (IR) of an implemented algorithm's binary to the associated mathematical expression. Analogous to deep learning approaches applied in symbolic mathematics [3], we will leverage the infinite set of mathematical expressions to synthesize large training data sets.

A second challenge for automatic CPS binary analysis frameworks is understanding the impact of binary code at various abstraction levels. Control systems are inherently modular, consisting of cascading control components that estimate sensor states, calculate errors relative to reference points, and subsequently adjust actuation. A control system representation may comprise several algorithms, each algorithm may include multiple functions, and any function may contain many basic blocks implementing mathematical subexpressions. Thus, automatic abstraction and communication of mathematical expressions require appropriate interfaces for respective domain experts.

We will also study how to integrate these techniques into existing workflows with control systems under the guidance of our SME, Dr. David Barnhart. We will iteratively tune our approach of domain concept extraction and communication for programmable logic controller and robotic vehicle binaries. We will work with Dr. Barnhart and Space Engineering Research Center (SERC) to analyze binaries in the context of real control system design workflows, guidance, navigation, and control (GNC) algorithms used for lunar lander spacecrafts.

3.2. Sequence Representation for Mathematical and Software Expressions

Recent developments in sequence-to-sequence machine learning translation models for symbolic mathematics [3] leverage the notion that mathematical expressions can ultimately be expressed as a sequence of tokens. First, the expressions are represented as abstract syntax trees: operators and functions act as internal nodes, and all other terminal operands (e.g., numbers, variables, and constants) are leaves. The associated tree representations are easily translated to a preorder sequence of operations and corresponding operands. Similarly, symbolic execution of binary code involves abstract syntax tree representations. Our team has been exploring graphical representations [4] of binary programs for deep learning security tasks. We will take inspiration from recent machine translation approaches [6] in order to develop a sequence representation for both mathematical expressions as well as abstracted software expressions.

Developing a sequence representation of a binary's abstract syntax tree is the crux of our framework and will be the initial emphasis for Phase 1. In particular, we will assess the applicability of the abstract syntax tree representations of symbolic function outputs from our prior work [1] in the context of sequence to sequence translation. If we initially assume to know the input and output variables in a binary program, we can use a symbolic execution engine (e.g., KLEE [8] or angr [9]) to generate a symbolic abstract syntax tree representation of the

output, focusing only on arithmetic operations on the arithmetic variables.

The core challenge of our feasibility study will be analyzing the algorithmic expressions that are made obscure by each stage of the translation process: the source code implementation of the algorithmic expression, the binary compilation of the source code, as well as the abstracted intermediate representation (IR) of the binary. Concretely, going from the natural symbolic representation of a design-level math expression to its instantiation in source code followed by its compilation to a low-level representation is a well-defined process; reversing this process is not. In Phase 1, we will start with source-level representations to establish an abstract syntax tree representation represented as a sequence. We hypothesize that a solution component will entail leveraging prior techniques to partially simplify the symbolic output and yield more user-friendly symbolic expressions [7] before performing sequence to sequence translation. We will leverage the intuitions gained as we explore sequence representations for binary-based representations in Phase 2.

3.3 Software Expression Sequence to Mathematical Expression Translation

Before making any decisions regarding the architecture design for a machine translation model, we must ensure sufficient training data. Assuming we will have a sequence representation of a binary's symbolic abstract syntax tree, we will need to synthesize a vast amount of examples (potentially up to tens of millions) of different binary symbolic output sequence representations to the respective mathematical expression sequence representations. We will draw upon previous works that leveraged deep learning for symbolic mathematics [1] and similarly generated an ample amount of data sample pairs of mathematical problem expressions to their solution expressions. In our context, we will overcome data scarcity by generating a sufficient amount of random mathematical expressions and translate them to the respective binary symbolic representations.

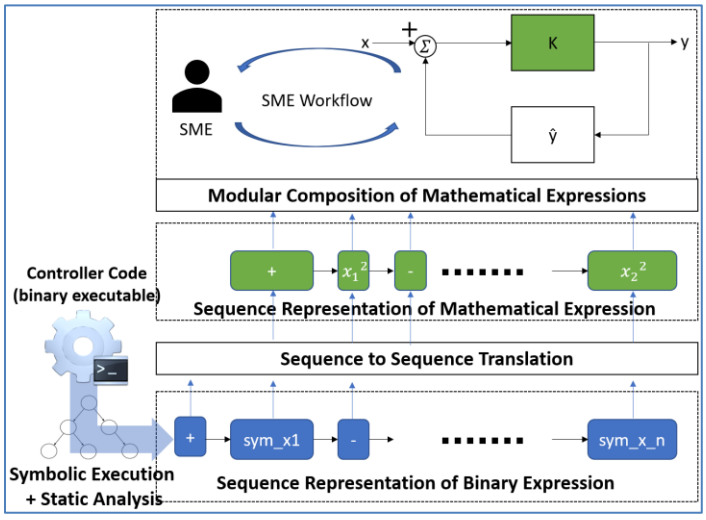
Generating millions of mathematical expressions is a challenge, but this problem is extensively addressed in prior work [1] and, thus, is outside of the scope of our contributions. In Phase 1, Task 1.2, will focus on the generation of mathematical expressions to source-level representations. Concretely, we aim to generate source code subroutines representing the respective implementation of a mathematical expression. A source-code representation represents a middle-ground between user-friendliness and expressivity: programmatic mathematical expressions are meant to be human-readable and closer to their natural form. Moreover, a source-code representation provides us the flexibility to augment the dataset with many target representations. In particular, we will generate a large dataset that maps mathematical expressions to a source-level subroutine, and the source-level subroutine will have a symbolically executed output representation as well as any number of binary representations. The binary representations can be varied with respect to compiler-level optimizations and architecture targets. We will initially target the C programming language as it is a practical industry standard for CPS and is supported by a wide range of program analysis tools.

We anticipate that we will need to ensure that our distribution facilitates generalizable machine

translation models. Initially, we will assess the inherent randomness from the generation of the associated mathematical expressions. However, because we plan on exploring several permutations of mathematical expressions to different software abstractions, we will explicitly employ generalization techniques for machine translation across many different architectures. As part of Tasks 2.3 and 2.4, we will explore translating low-level expressions from binary files directly to natural mathematical expressions. Thus, we will discuss with ISI's Dr. Jonathan May, machine translation expert, regarding the model design choices for different scenarios. For instance, in the case of binary files, we would like to build a model that translates from many different possible architectures to a mathematical expression. In this case, we would choose a machine translation design that would allow us to fine-tune the model by exposing it to a new architecture [18]. On the other hand, when composing more complex mathematical operations, we may encounter instances where we have a few examples implemented for a particular architecture. In this case, we can leverage low-resource approaches for generalizing translational models [16]. This approach entails training a parent model based to encode extracted features from the target binary with an ample number of binary samples from a single architecture, and then transferring some of the learned parameters to the low-resource child model. We will test and validate various permutations of the model design against our problem setting.

3.4 Modular composition and extraction of CPS control algorithm expressions

The previous sections focused on the fundamental tasks of mapping a binary implementation of a single function that implements the associated mathematical expression. In reality, cyber-physical binaries in the wild are undeniably much larger and more complex. Moreover, the mathematical expressions of control systems are typically modular, comprised of different mathematical components. For instance, the path from an inertial measurement unit (IMU) sensor to a motor actuator on an unmanned aerial vehicle (UAV) may include a state estimator (e.g., a Kalman filter)--to estimate the position and velocity--as well as a controller (e.g., a proportional-integral-derivative (PID) controller) to calculate the actuation signal based on the computed setpoint error. Composing mathematical expressions into their appropriate modules will be critical for interfacing with domain concepts communicated to the SME.



In order to automate the entire algorithmic expression composition, we need to identify a set of candidate modules. The output of our framework will generate abstractions of mathematical subexpressions that explicitly interact with each other in a cascading fashion from the control system's input values (sensors) to the output values (actuators), i.e., the input from a sensor is fed into one module's input, and the module's output is fed into a subsequent series of modules

until an actuator is reached. However, identifying the path from sensor to actuator is not trivial.

Identification of memory-mapped peripherals (sensors and actuators) in embedded control systems firmware is an open problem. Initially, we will draw upon the same assumptions from our previous PLC reverse engineering work [16] and ICSREF [2], where knowledge of memory-mapped peripherals is obtained from data sheets or configuration files. However, we will explore the feasibility of automatically identifying peripherals via firmware rehosting techniques. Recent works have used firmware rehosting along with general peripheral models to fuzz embedded systems firmware [14, 15]. We hypothesize that these general peripheral models can be utilized for input-to-output analysis, i.e., information flow and data analysis techniques may be utilized to extract the modules and their associated CFGs. Regardless of the approach, automatic identification of the input-to-output control loop cycle will be a significant advancement from prior research that assumes a set of candidate modules [1].

Once candidate modules and their input/output variables have been identified, we can symbolically execute the module and generate the symbolic output representation. However, our prior work [1] highlighted the fact that several control systems function symbolic outputs are noisy due to potentially irrelevant branches such as error-checking. In Task 1.3, we will investigate the super-graph inference, leveraging functional relationships between modules. We will explore static analysis methods to infer data dependencies as well as various clustering approaches to infer super-classes (modules), such as approaches based on graph spectral measures [5, 10].

The end-to-end automation of the symbolic extraction will have profound impacts beyond communicating domain concepts from a binary program. Our novel approach to input-to-output modularization generalizes to a broad class of binary analysis research.

3.5 Subject Matter Expert and Identified Use Cases

Our subject matter expert, Dr. David Barnhart, leads USC ISI's Space Systems and Technology group, and has extensive experience with control system design of robotics applied to and with space systems. He is actively working on lunar lander technology, and will provide us guidance and access to the lunar lander design and simulation software. The particular instantiation that this platform is attempting to create is to merge two different actuation systems that both can provide attitude control and translation, and to optimize their use based on various factors.

More importantly, we have identified a workflow use case with Dr. Barnhart to iteratively evaluate: any controller software will be initially designed in a modeling, simulation, and dynamic analysis system such as Simulink [11] and MatrixX [12]. The programming environment generates embedded systems code (i.e., C code) that will subsequently be integrated into the target platform. SMELLCPS will extract the mathematical expressions from both the generated source as well as the compiled binary file to verify the fidelity of the translations against the intended mathematical expression, i.e., verifying whether the algorithmic expression from the modeling software matches the algorithmic expression extracted from the generated embedded systems code as well as the expression extracted from

the compiled binary program.

In addition to the lunar lander model in Task 1.4, we will also analyze two classes of systems that our team has extensive experience with: robotic vehicles based on the open-source Ardupilot project [13] as well as programmable logic controllers (PLCs). Both classes of devices regularly use control system algorithm implementations.

3.6 Schedule

Our tasks described above contribute to the following milestones, in ways defined in the Task Description Document (TDD):

Phase 1:

Month 1: Detailed experimental plan for the identified SME use cases. We will identify a sequence representation to enable the generation of our training corpora for sequence to sequence translation.

Month 3: Reliable identification of mathematical primitives aided by source-level representations (source code, pseudo-code, or code in a domain-specific language). In this task we will generate pairs of source-level symbolic representations to train our sequence to sequence machine translation models.

Month 6: Reliable extraction of formulaic expressions aided by source-level representations. We will have a preliminary demonstration evaluating machine translation techniques for translating source-level representations to mathematical expressions. The demonstration will subsequently compose the expressions into their high-level representations in the context of CPS source-level programs.

Month 9: Extraction of SME domain communication units aided by source-level representations. We will evaluate SMELLCPS with three case studies, including the SME workflow.

Phase 2:

Month 12: Reliable identification of mathematical primitives from binary-level code, unaided by source-level representations. We will identify a sequence representation for binary programs to enable the generation of our training corpora for sequence to sequence translation. We will subsequently generate datasets for mapping binary software expressions to mathematical expressions.

Month 15: Reliable extraction of formulaic expressions from binary-level code (compiled executable code or bytecode), unaided by source-level representations. We will integrate the full automated pipeline of binary lifting and mathematical expression composition into a single demonstration.

Month 18: Extraction of SME domain communication units from binary-level representations,

final prototype and report. We will finalize our prototype in coordination with the SME workflow use case

4. Capabilities/Management Plan

Team: To address the wealth of challenges outlined in this proposal, we assembled a world-class team of researchers with proven records of innovation and collaboration in CPS/IoT systems, binary program analysis, and artificial intelligence. Our team includes researchers from the USC Information Sciences Institute (USC/ISI), with Dr. **Luis Garcia** (Computer Research Scientist at USC/ISI) as PI, Dr. **Christophe Hauser** (Computer Research Scientist at USC/ISI) as Co-PI, Dr. **Aram Galstyan** (Director of the Artificial Intelligence Division at USC/ISI) as Senior Personnel, and Dr. **David Barnhart** (Director of Space Systems and Technology Division at USC/ISI) as Senior Personnel and Subject Matter Expert (SME). Dr. **Luis Garcia** is working on the safety, security, and verification of learning-enabled CPS. He has a large body of research on program analysis of CPS at various levels of abstraction, including mathematical expression extraction within CPS binaries. Dr. **Hauser** developed an extensive research portfolio on program binary analysis. Dr. **Hauser** was part of the research and development efforts for angr, a next-generation platform for binary analysis that will likely be a core enabler of SMELLCPS. Dr. **Galstyan** has contributed several foundational papers in Artificial Intelligence at the intersection of machine learning, information theory, and statistical physics. Dr. **Barnhart** leads the USC Space Engineering Research Center (SERC) and specializes in developing innovative technologies and architectures for 2nd generation space morphologies, satellite robotics, and inspiration-based engineering techniques. **Synergies:** **Galstyan** is the Co-PI of the DARPA programs *"CORAL : Combined Representations for Adept Learning,"* "*Discerning Group Biases in Online Communities via Linguistics Analysis*," "*DSBOX: Data Scientist in A Box,*" and "*Global Analysis of Weak Signals for Enterprise Event Detection (GAWSEED)*." As a senior space Project Manager at DARPA, **Barnhart** pioneered cellular spacecraft morphologies, satbotics, and space robotics on the Phoenix and SeeMe projects. **Galstyan** and **Hauser** are actively collaborating on deep learning representations for binary program analysis. All personnel on the team are located at the USC/ISI campus in Marina Del Rey, which will facilitate communication and interaction. **Management:** Dr. **Garcia** will coordinate and manage the activities of the USC/ISI-led team. He will serve as the POC with DARPA and will have overall responsibility for program management and technical direction. USI/ISI employs program assistants who will support Dr. **Garcia** to ensure smooth and efficient operations. Furthermore, Dr. **Garcia** will be responsible for research directions that interface CPS analysis with program analysis, while also serving as facilitator between the SMELLCPS development and the SME, Dr. **Barnhart**. Dr. **Hauser** will coordinate the research tasks on binary program analysis. Dr. **Galstyan** will advise research tasks on Artificial Intelligence. Dr. **Barnhart** will provide subject matter expertise for control systems design workflows. **Garcia** will supervise the work of a Research programmer, while **Galstyan**, **Barnhart**, and **Hauser** will oversee three graduate students. Rich intra-team interactions will guarantee the ability to integrate machine translation and binary program analysis under a unified framework.

5. Task Description Document (TDD)

The objective of this proposal is to address the fundamental problem of reconstructing high-level, natural mathematical expressions from their low-level code representation.

Phase 1 (Base) Tasks

Task 1.1		
Objective: Month 1 Plan Development and Sequence Representation	**Task Description** In this task we will identify a sequence representation to enable the generation of our training corpora for sequence to sequence translation.	**Location** USC ISI Marina Del Rey Campus
Primary Organization Responsible	Garcia and Hauser, with support from Galstyan	
Human Subjects or Animal Research?	No	
Associated Milestones	Deliverables	
1. Sequence representation report	Our formal report will include the syntactical and semantic properties of our source math expressions and the associated software targets used for training.	
2. Draft and final feasibility analysis reports		
3. Literature studies and structured interviews with iterative feedback on solutions		

Task 1.2

Objective: Month 3 Data Set Generation	**Task Description** In this task we will generate pairs of source-level symbolic representations to train our sequence to sequence machine translation models.	**Location** USC ISI Marina Del Rey Campus
Primary Organization Responsible	Garcia and Hauser, with support from Galstyan	
Human Subjects or Animal Research?	No	
Associated Milestones	Deliverables	
1. Dataset with tens of millions of training samples	We will have generated a dataset that maps tens of millions of math expressions to a target source-code subroutine, and each source-code sample will map to the symbolic execution output representation.	
2. Draft and final feasibility analysis reports		

Task 1.3		
Objective: Month 6 Demonstration	**Task Description** In this task we will have a preliminary demonstration evaluating machine translation techniques for translating source-level representations to mathematical expressions. The demonstration will subsequently compose the expressions into their high-level representations in the context of CPS source-level programs.	**Location** USC ISI Marina Del Rey Campus
Primary Organization Responsible	Garcia and Hauser, with support from Galstyan	

Human Subjects or Animal Research?	No
Associated Milestones	Deliverables
1. Preliminary results for machine translation of source-level representation to Math expressions	Experimental results illustrating the feasibility of mapping source-level representations to high-level mathematical expressions
2. Demonstration for end-to-end automation for source expression extraction	Preliminary demonstration choice and targets for end-to-end automation for an entire CPS program—not just at a function-level.
3. Draft and final feasibility analysis reports	

Task 1.4		
Objective: Month 9 Case Studies and Phase 1 Report	**Task Description** In this task, we will evaluate SMELLCPS with three case studies, including the SME workflow.	**Location** USC ISI Marina Del Rey Campus
Primary Organization Responsible	Garcia and Hauser, with support from Galstyan	
Human Subjects or Animal Research?	No	
Associated Milestones	Deliverables	
1. Structured interviews with iterative feedback	We will explicitly evaluate our approach on systems within the SERC group and gather feedback for integration into our design and implementation choices for Phase 2.	
2. Case study results	We will include a report of our preliminary results on source-level firmware representations of an unmanned aerial vehicle, a programmable logic controller, and our SME workflow.	

3. Draft and final feasibility analysis reports	

Phase 2 (Option) Tasks

Task 2.1		
Objective: Month 12 Automated Training Corpora	**Task Description** In this task we will identify a sequence representation for binary programs to enable the generation of our training corpora for sequence to sequence translation. We will subsequently generate datasets for mapping binary software expressions to mathematical expressions.	**Location** USC ISI Marina Del Rey Campus
Primary Organization Responsible	Garcia and Hauser, with support from Galstyan	
Human Subjects or Animal Research?	No	
Associated Milestones	Deliverables	
1. Sequence representation report	The report will include the design choices considered, as well as the syntax and semantics expected for a standard sequence representation across architectures.	
2. Dataset with tens of millions of training samples	We will augment the previous dataset to include many examples of binary targets for each source-code function sample, where the binary files could vary depending on compiler optimizations or hardware target.	
3. Draft and final feasibility analysis reports		

Task 2.2		
Objective: Month 15 Demonstrated Integration	**Task Description** In this task we will integrate the full automated pipeline of binary lifting and mathematical expression composition into a single demonstration.	**Location** USC ISI Marina Del Rey Campus
Primary Organization Responsible	Garcia and Hauser, with support from Galstyan	
Human Subjects or Animal Research?	No	
Associated Milestones	Deliverables	
1. Preliminary demonstration for machine translation of binary-level representation to Math expressions	Experimental results illustrating the feasibility of mapping binary-level representations to high-level mathematical expressions	
2. Demo for end-to-end automation for binary-level expression extraction	Final design choice and targets for end-to-end automation for an entire CPS program—not just at a function-level.	
3. Draft and final feasibility analysis reports		

Task 2.3		
Objective: Month 18 Final Prototype	**Task Description** In this task we will finalize our prototype in coordination with the SME workflow use case.	**Location** USC ISI Marina Del Rey Campus

Primary Organization Responsible	Garcia and Hauser, with support from Galstyan
Human Subjects or Animal Research?	No
Associated Milestones	Deliverables
1. Prototype demonstration on SME workflow	We will explicitly evaluate our approach on systems within the SERC group for a final prototype design.
2. Draft and final feasibility analysis reports	The report will include an overview of how the prototype solves the problems identified in the feasibility analysis from prior tasks.

6. Bibliography (optional)

[1] Sun, P., Garcia, L., & Zonouz, S. (2019, June). Tell Me More Than Just Assembly! Reversing Cyber-Physical Execution Semantics of Embedded IoT Controller Software Binaries. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 349-361). IEEE. https://ieeexplore.ieee.org/abstract/document/8809529 https://dl.acm.org/doi/abs/10.1145/3243734.3243752

[2] Keliris, A., & Maniatakos, M. (2018). Icsref: A framework for automated reverse engineering of industrial control systems binaries. https://arxiv.org/pdf/1812.03478.pdf

[3] Lample, G., & Charton, F. (2019, September). Deep Learning For Symbolic Mathematics. In *International Conference on Learning Representations*. https://arxiv.org/abs/1912.01412

[4] Arakelyan, S., Hauser, C., Kline, E., & Galstyan, A. (2020). Towards Learning Representations of Binary Executable Files for Security Tasks. https://arxiv.org/pdf/2002.03388.pdf

[5] Chauhan, J., Nathani, D., & Kaul, M. (2020). Few-Shot Learning on Graphs via Super-Classes based on Graph Spectral Measures. https://arxiv.org/pdf/2002.12815.pdf

[6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008). https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[7] Hensel, J., Giesl, J., Frohn, F., & Ströder, T. (2016, July). Proving termination of programs with bitvector arithmetic by symbolic execution. In *International Conference on Software Engineering and Formal Methods* (pp. 234-252). Springer, Cham. https://verify.rwth-aachen.de/giesl/papers/talkSEFM2016.pdf

[8] Cadar, C., Dunbar, D., & Engler, D. R. (2008, December). Klee: unassisted and automatic generation of high-coverage tests for complex systems programs. In *OSDI* (Vol. 8, pp. 209-224). https://static.usenix.org/events/osdi08/tech/full_papers/cadar/cadar.pdf

[9] Wang, F., & Shoshitaishvili, Y. (2017, September). Angr-the next generation of binary analysis. In *2017 IEEE Cybersecurity Development (SecDev)* (pp. 8-9). IEEE. https://ieeexplore.ieee.org/abstract/document/8077799/

[10] Jovanović, I., & Stanić, Z. (2012). Spectral distances of graphs. *Linear algebra and its applications*, *436*(5), 1425-1435. https://www.sciencedirect.com/science/article/pii/S0166218X15001912

[11] Simulink. https://www.mathworks.com/products/simulink.html

[12] MatrixX. http://www.ni.com/pdf/manuals/371148b

[13] Ardupilot. https://ardupilot.org/

[14] Feng, B., Mera, A., & Lu, L. (2020). P 2 IM: Scalable and Hardware-independent Firmware Testing via Automatic Peripheral Interface Modeling. In *Proceedings of the 29th USENIX Security Symposium*. https://www.usenix.org/system/files/sec20spring_feng_prepub_0.pdf

[15] Clements, A. A., Gustafson, E., Scharnowski, T., Grosen, P., Fritz, D., Kruegel, C., ... & Payer, M. (2020). HALucinator: Firmware Re-hosting Through Abstraction Layer Emulation. In *29th USENIX Security Symposium (USENIX Sec)* (pp. 1-18). https://www.usenix.org/system/files/sec20summer_clements_prepub.pdf

[16] Garcia, L., Brasser, F., Cintuglu, M. H., Sadeghi, A. R., Mohammed, O. A., & Zonouz, S. A. (2017, February). Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. In *NDSS*. https://www.ndss-symposium.org/wp-content/uploads/2017/09/ndss2017_08-1_Garcia_paper.pdf

[17] Zoph, B., Yuret, D., May, J., & Knight, K. (2016). Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201*. https://arxiv.org/pdf/1604.02201

[18] Pust, M., Hermjakob, U., Knight, K., Marcu, D., & May, J. (2015, September). Parsing English into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1143-1154). https://www.aclweb.org/anthology/D15-1136.pdf