A Generic Static Analysis Framework for Domain-specific Languages

Avijit Mandal, Devina Mohan, Raoul Jetley
ABB Corporate Research
Bangalore, India
{avijit.mandal, devina.mohan, raoul.jetley}@in.abb.com

Sreeja Nair†
ABB Corporate Research
Bangalore, India
sreeja.in@gmail.com

Meenakshi D'Souza
IIIT Bangalore
Bangalore, India
meenakshi@iiitb.ac.in

***Abstract*—Software used to monitor and control operations within an automation system is defined using domain-specific languages. Latent errors in the control code, if left undetected, can lead to unexpected system failures compromising the safety and the security of the automation system. Traditional analysis techniques are insufficient to detect such errors as they do not cater specifically to the underlying domain-specific language. However, given the diversity of different automation domains, there is no standard platform for analysis of these languages.**
This paper proposes a generic static analysis framework for domain-specific languages used in the automation domain. The analysis approach exhaustively detects runtime errors in control code and ensures compliance to good programming practices. These runtime errors and coding violations are checked against abstract syntax trees and control flow graphs derived from the code. Data Flow Analysis (DFA), Abstract interpretation and pattern-based matching techniques are used to identify domain specific errors and coding violations for control languages.

***Index Terms*—Static analysis, Data Flow Analysis (DFA), Abstract Interpretation, Generic programming errors, Safety requirements, Interrupt, Exception, Inter-procedural CFG.**

I. Introduction

Automation engineering systems rely on software to monitor and control various operations like batch processing, arc welding etc. This software, however, is prone to errors that may creep in during development. Such errors, if left undetected, can manifest themselves as failures or be exploited by malicious intruders. Moreover, fixing such errors at later stages of development or after deployment entails high maintenance cost and requires extra effort.

Automation systems are implemented using domain-specific languages for monitoring and control. Due to the use of complex data structures, task-based parallel execution and unique semantics for execution order, analysis techniques for general-purpose programming languages are not always adequate for these systems. Thus, we need specialized tools and techniques that address the domain-specific languages. One solution to detect errors in automation systems is to use static code analysis to identify potential sources of errors during compile time. The technique involves examining all possible execution paths in the source code. Static code analysis for general-purpose programming languages is widespread across the software industry. Many static analyzers are widely adopted, e.g., Coverity for C, PolySpace for C/C++/Ada etc. However, such tools are rarely used for automation systems. Typically, verifications tools for control software focus on pattern based matching to detect code violations. Such tools are platform dependent and do not port well across various developing environments.

The work presented in this paper proposes a generic static analysis framework for industrial software. The framework can be used to efficiently perform DFA and pattern-matching based checks on industrial software. The framework can be extended to various programming languages addressing different industrial domains. The main objective is to build a verification platform to ensure correctness of safety-critical software. This paper focuses on the adaptation of the proposed framework to real-world control applications. To this effect, the framework has been extended for analysis of the following languages:

- IEC 61131-3 languages for PLC programming [5]
- Electronic Device Description Language (EDDL), used for configuration of field devices [7], and
- RAPID, a domain-specific language used for programming industrial robots [6]

Key contributions in this paper are,

- Generic datatype to represent the parsed information for the three languages
- Flexible DFA engine to encode more data flow rules as needed by varying the domain
- Flexible rule engine to process data for further analysis

The rest of the paper is structured as follows: Section II explores existing related work. Section III introduces the basic concepts involved. Section IV explains the framework of the prototype. Section V discusses the implementation and extensions for domain-specific languages. Section VI presents analyses of real-world applications and results thereof. Finally, Section VII summarizes the paper and lists future directions.

†Sreeja Nair is currently at Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France

978-1-5386-7108-5/18/$31.00 ©2018 IEEE