Verification of Programmable Logic Controller Code using Model Checking and Static Analysis

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Sebastian Biallas
aus Düsseldorf

Berichter: Universitätsprofessor Dr.-Ing. Stefan Kowalewski
Universitätsprofessor Dr.-Ing. Alexander Fay

Tag der mündlichen Prüfung: 14.7.2016

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: D 82 (Diss. RWTH Aachen University, 2016)

Sebastian Biallas
Informatik 11 — Embedded Software
biallas@embedded.rwth-aachen.de

Aachener Informatik Bericht AIB-2016-07

Herausgeber: Fachgruppe Informatik
RWTH Aachen University
Ahornstr. 55
52074 Aachen
GERMANY

ISSN 0935-3232

ISBN 978-3-8440-4711-0

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen
Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

ABSTRACT

Programmable Logic Controllers (PLCs, ger. Speicherprogrammierbare Steuerungen) are control devices used in industry to control, operate and monitor plants, machines and robots. PLCs comprise input connectors, typically connected to sensors, output connectors, typically connected to actuators, and a program, which controls the behavior, computing new output values based on the input values and an internal state. Since PLCs operate in safety-critical environments, where a malfunction could seriously harm the environment, humans, or the plant, it is recommended to verify their programs using formal methods.

This dissertation studies the formal methods *model checking* and *static analysis* to prove the correctness of PLC programs. For this, we developed the tool ARCADE.PLC, which allows the automatic application of these methods to PLC programs written in different vendor-specific dialects. It extracts a model from the program by abstract simulation, which over-approximates the possible program behavior. The user is then able to verify whether the model obeys a specification, which can be written in the logic CTL or using automata.

For applying model checking, we demonstrate how the model can be extracted automatically, such that the approach scales to industrial size programs. For this, we introduce two different abstraction techniques: First, we develop an abstraction refinement guided by the model checker that automatically creates an abstracted model by iteratively analyzing counterexamples. Second, we implemented a predicate abstraction that evaluates a formalized program semantics using an SMT solver. Both techniques are evaluated using different programs from industry and academia. Further, we introduce a simplified formalism to write specifications, which is influenced by an automata-based language established in industry. We implement an algorithm to check programs using this formalism and show that this technique is even able to detect errors in the specification. Finally, we detail how counterexamples generated by the model checker can be analyzed automatically to locate the actual erroneous line in the program.

The static analysis we developed is able to detect program errors in a fully automatic way. We detect typical errors such as division by zero and illegal array accesses, but also PLC specific errors, e. g., during a restart. The analysis is based on a value-set analysis, which determines the values of all program variables in each program location. These sets are then verified against the predefined checks or user-provided annotations. We show how to implement this analysis such that it scales to industrial size programs. The approach is evaluated on an industrial case study.

ZUSAMMENFASSUNG

Speicherprogrammierbare Steuerungen (SPSen, engl. Programmable Logic Controller) sind Automatisierungsgeräte, welche zur Steuerung, Regelung und Überwachung von industriellen Anlagen und Maschinen eingesetzt werden. Sie besitzen dazu Eingänge, die mit Sensoren verbunden sind, Ausgänge, die mit Aktuatoren verbunden sind und ein Programm, welches die Ausgänge in Abhängigkeit der Eingänge und eines internen Speichers belegt. Da SPSen häufig in kritischen Bereichen eingesetzt werden, in denen eine Fehlfunktion Gefahren für Mensch, Umwelt oder die Anlage bergen kann, ist die Korrektheit des Programms zu prüfen.

Diese Dissertation untersucht die formalen Methoden *Model-Checking* und *Statische Analyse*, um die Korrektheit von PLC-Programmen zu beweisen. Wir haben dazu das Tool ARCADE.PLC geschrieben, welches es ermöglicht, diese Techniken vollautomatisch auf PLC-Programme verschiedener Hersteller anzuwenden. Es extrahiert durch abstrakte Simulation ein Modell, welches sämtliches Programmverhalten widerspiegelt. Der Benutzer kann dann überprüfen, ob das Modell einer Spezifikation entspricht, welche er in der Logik CTL formulieren muss oder als Automaten eingegeben kann.

Zum Bereich Model-Checking zeigen wir in dieser Dissertation, wie das Modell automatisch abstrahiert werden kann, so dass der Ansatz auch für industrielle Programme skaliert. Es werden dazu zwei verschiedene Abstraktionstechniken eingeführt: Eine durch den Model-Checker gesteuerte Abstraktionsverfeinerung erstellt ein abstrahiertes Modell iterativ durch Analyse von Gegenbeispielen. Außerdem haben wir eine automatische Prädikat-Abstraktion implementiert, welche mithilfe einer SMT-Solvers die formalisierte Programmsematik auf Prädikaten auswertet. Beide Techniken werden anhand verschiedener Programme evaluiert. Weiterhin führen wir einen vereinfachten Spezifkationsformalismus ein, welcher sich an einer in der Industrie etablierten Automatensprache orientiert. Wir implementieren einen Algorithmus, um Programme mit diesem Formalismus zu überprüfen und zeigen, dass durch diese Technik auch Spezifikationsfehler entdeckt werden können. Schließlich zeigen wir noch, wie vom Model-Checker gefundene Gegenbeispiele analysiert werden können, um die eigentlich fehlerhafte Programmzeile automatisch zu lokalisieren.

Die von uns implementierte Statische Analyse kann vollautomatisch Programmfehler entdecken. Dazu gehören beispielsweise eine Division durch Null, unerlaubte Array-Zugriffe oder PLC-spezifische Fehler z.B. beim Neustart. Die Analyse basiert auf einer Wertebereichsanalyse, welche eine Übermenge der Werte aller Variablen in allen Programmstellen berechnet. Wir zeigen, wie diese Analyse skalierbar implementiert werden kann. Der Ansatz wird an einer großen industriellen Fallstudie ausgewertet.

ACKNOWLEDGEMENTS

I worked on this dissertation while I was employed as a research assistant at the chair *Informatik 11 — Embedded Software* at the RWTH Aachen University. This work would not have been possible without the support of many others. First of all, I have to thank Prof. Dr.-Ing. Stefan Kowalewski for giving me the opportunity to join his group, for supporting my thesis, and for the excellent collaboration during this time. I would also like to thank Prof. Dr.-Ing. Alexander Fay for serving as a second supervisor and for helpful remarks. Furthermore, I thank Prof. Dr. Bernhard Rumpe and apl. Prof. Dr. Thomas Noll for participating in my examination committee.

I have to thank Dr. Bastian Schlich, whom I first met when I was a student while he was a postdoctoral researcher at the chair. He introduced me to the topic of formal verification of embedded software and supervised by Diploma thesis. Later, he went to ABB and was able to establish an industrial research collaboration. I would also like to thank his colleagues, especially Dr. Stefan Hauck-Stattelmann, for the great collaboration during this time.

Furthermore, I would like to thank all my former colleagues and friends for the great time I had at the chair. In particular, I enjoyed the numerous interesting discussions I had with Dr. Jörg Brauer even outside the work environment.

I also have to thank Dr. Ralf Huuck for giving me the possibility to join his group at NICTA during a research visit. I learned a lot about the static analysis of C and C++ programs and how to scale such an analysis to industrial size programs.

I especially have to thank my students. They wrote excellent bachelor and master theses, implemented algorithms and user interfaces, performed case studies, and contributed as co-authors of publications. This work would not have been possible without them.

Financially, my work was supported by the *Deutsche Forschungsgemeinschaft*. Further, I was supported by the DFG research training group 1298 *Algorithmic Synthesis of Reactive and Discrete-Continuous Systems* and the DFG Cluster of Excellence on *Ultra High Speed Mobile Information and Communication*. I am very grateful for this support and I also have to thank these groups for many interesting and stimulating discussions.

Finally, I have to thank my parents and my sister for support and proof-reading.

Sebastian Biallas
July 2016, Berlin

CONTENTS