

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology Madras, Chennai, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/7408>


Amel Bennaceur · Reiner Hähnle
Karl Meinke (Eds.)


Machine Learning for Dynamic Software Analysis

Potentials and Limits

International Dagstuhl Seminar 16172
Dagstuhl Castle, Germany, April 24–27, 2016
Revised Papers

Editors

Amel Bennaceur 
The Open University
Milton Keynes
UK

Karl Meinke 
KTH Royal Institute of Technology
Stockholm
Sweden

Reiner Hähnle 
Technische Universität Darmstadt
Darmstadt
Germany

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-96561-1 ISBN 978-3-319-96562-8 (eBook)
<https://doi.org/10.1007/978-3-319-96562-8>

Library of Congress Control Number: 2018948379

LNCS Sublibrary: SL2 – Programming and Software Engineering

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Cover illustration: Classification of key concepts of ML for software engineering. LNCS 11026, p. 5. Used with permission.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Machine learning of software artefacts is an emerging area of interaction between the machine learning (ML) and software analysis (SA) communities. Increased productivity in software engineering hinges on the creation of new adaptive, scalable tools that can analyze large and continuously changing software systems. For example: Agile software development using continuous integration and delivery can require new documentation models, static analyses, proofs and tests of millions of lines of code every 24 h. These needs are being addressed by new SA techniques based on ML, such as learning-based software testing, invariant generation, or code synthesis. ML is a powerful paradigm for SA that provides novel approaches to automating the generation of models and other essential artefacts. However, the ML and SA communities are traditionally separate, each with its own agenda.

This book is a follow-up of a Dagstuhl Seminar entitled “16172: Machine Learning for Dynamic Software Analysis: Potentials and Limits” that was held during April 24–27, 2016. This seminar brought together top researchers active in these two fields to present the state of the art and suggest new directions and collaborations for future research. We, the organizers, feel strongly that both communities have much to learn from each other, and the seminar focused strongly on fostering a spirit of collaboration in order to share insights and to expand and strengthen the cross-fertilization between these two communities.

Our goal in this book is to give an overview of the ML techniques that can be used for SA and provide some example applications of their use. Besides an introductory chapter, the book is structured into three parts: testing and learning, extension of automata learning, and integrative approaches as follows.

Introduction

- The chapter by Bennaceur and Meinke entitled “Machine Learning for Software Analysis: Models, Methods, and Applications” introduces the key concepts of ML focusing on models and some of their applications in software engineering.

Testing and Learning

- The chapter by Meinke entitled “Learning-Based Testing: Recent Progress and Future Prospects” reviews the fundamental concepts and theoretical principles of learning-based techniques.
- The chapter by Aichernig, Mostowski, Mousavi, Tappler and Tatomirad entitled “Model-Based Testing and Learning” provides an overview of the different models that can be used for testing and how they can be learnt.

- The chapter by Walkinshaw entitled “Testing Functional Black-Box Programs without a Specification” focuses on examining test executions and informing the selection of tests from programs that do not require sequential inputs.

Extensions of Automata Learning

- The chapter by Howar and Steffen entitled “Active Automata Learning in Practice: An Annotated Bibliography of the Years 2011 to 2016” reviews the state of the art and the open challenges for active automata learning.
- The chapter by Cassel, Howar, Jonsson and Steffen entitled “Extending Automata Learning to Extended Finite State Machines” focuses on automata learning for extended finite state machines.
- The chapter by Groz, Simao, Petrenko, and Oriat entitled “Inferring FSM Models of Systems Without Reset” presents active automata learning algorithms that relax the assumptions about the existence of an external oracle.

Integrative Approaches

- The chapter by Hähnle and Steffen entitled “Constraint-Based Behavioral Consistency of Evolving Software Systems” proposes to combine glass-box analysis with automata learning to help bridge the gap between the design and implementation artefacts.
- The chapter by Alrajeh and Russo entitled “Logic-Based Machine Learning in Software Engineering” focuses on logic-based learning and its application for declarative specification refinement and revision.

While the papers in this book cover a wide range of topics regarding ML techniques for model-based software analysis, additional research challenges and related research topics still exist for further investigation.

We hope that you enjoy this book and that it will kindle your interest in and help your understanding of this fascinating area in the overlap of ML and SA. We thank the participants of the seminar for their time and their help in reviewing the chapters. Each chapter was reviewed by at least two reviewers and many went through several revisions. We acknowledge the support of Schloss Dagstuhl—Leibniz Center for Informatics and thank the whole Dagstuhl team for their professional approach that made it easy for the participants to network, to discuss, and to have a very productive seminar. And finally, we sincerely thank the authors for their research efforts, for their willingness to respond to feedback from the reviewers and editorial team. Without their excellent contributions, this volume would not have been possible.

May 2018

Amel Bennaceur
Reiner Hähnle
Karl Meinke

Organization

Program Chairs

Amel Bennaceur
Reiner Hähnle
Karl Meinke

The Open University, UK
Technische Universität Darmstadt, Germany
KTH Royal Institute of Technology, Sweden

Program Committee

Amel Bennaceur
Roland Groz
Falk Howar
Reiner Hähnle
Karl Meinke
Mohammad Reza Mousavi
Bernhard Steffen
Frits Vaandrager
Neil Walkinshaw

The Open University, UK
Grenoble Institute of Technology, France
TU Dortmund and Fraunhofer ISST, Germany
Technische Universität Darmstadt, Germany
KTH Royal Institute of Technology, Sweden
School of IT, Halmstad University, Sweden
TU Dortmund, Germany
Radboud University, The Netherlands
The University of Leicester, UK

Contents

Introduction

Machine Learning for Software Analysis: Models, Methods, and Applications	3
<i>Amel Bennaceur and Karl Meinke</i>	

Testing and Learning

Learning-Based Testing: Recent Progress and Future Prospects.	53
<i>Karl Meinke</i>	
Model Learning and Model-Based Testing	74
<i>Bernhard K. Aichernig, Wojciech Mostowski, Mohammad Reza Mousavi, Martin Tappler, and Masoumeh Taramirad</i>	
Testing Functional Black-Box Programs Without a Specification.	101
<i>Neil Walkinshaw</i>	

Extensions of Automata Learning

Active Automata Learning in Practice: An Annotated Bibliography of the Years 2011 to 2016	123
<i>Falk Howar and Bernhard Steffen</i>	
Extending Automata Learning to Extended Finite State Machines	149
<i>Sofia Cassel, Falk Howar, Bengt Jonsson, and Bernhard Steffen</i>	
Inferring FSM Models of Systems Without Reset	178
<i>Roland Groz, Adenilso Simao, Alexandre Petrenko, and Catherine Oriat</i>	

Integrative Approaches

Constraint-Based Behavioral Consistency of Evolving Software Systems	205
<i>Reiner Hähnle and Bernhard Steffen</i>	
Logic-Based Learning: Theory and Application.	219
<i>Dalal Alrajeh and Alessandra Russo</i>	

Author Index	257
------------------------	-----