Bachelor & Master Projects and Theses

Prof. Dr. Stefan Leue

Software and Systems Engineering

http://sen.uni-konstanz.de/

Summer Term 2024
Members

- Prof. Dr. Stefan Leue {SL}
  - Email: Stefan.Leue@uni.kn, Room: PZ 902
- Rafael Senn {RS}
  - Email: Raffael.Senn@uni.kn, Room: PZ 912
- David Boetius {DB}
  - Email: David.Boetius@uni.kn, Room: PZ 912
Projects and Theses at the Chair

♦ Our Objectives
  ▸ projects and theses close to ongoing research projects
  ▸ links to practical and relevant applications
  ▸ completion of project and theses within defined time limits (examination regulations / Prüfungsordnung)

♦ What We Offer
  ▸ close and individual supervision
  ▸ regular meetings and guidance
  ▸ if possible and applicable, supervision in collaboration with external partners
    – research institutions
    – industry
Projects and Theses at the Chair

♦ Our Expectations

- **project** is typically a literature survey, problem statement or similar
  - leads to definition of thesis topic (not mandatory, but recommended)
  - project report: approx. 10-20 p.

- **thesis**
  - requires some own contribution
    - **Bachelor**: problem solution idea, critical literature survey, innovative case study, ...
    - **Master**: own problem solution concept, evolving an existing approach, algorithmic concept and implementation, revealing comparison with other approaches, ...
Scope and Duration of Projects/Theses

- **Project (Bachelor and Master)**
  - 1 semester
  - 9 ECTS (270h work)

- **Thesis (Bachelor)**
  - 3 months (1/2 Semester)
  - 12 ECTS (Thesis) + 3 ECTS (Colloquium) = 15 ECTS (450h work)

- **Thesis (Master)**
  - 6 Months (1 Semester)
  - 30 (Thesis + Colloquium) ECTS (900h work)
For BA/MA Projects and Theses, the Following Dates Apply

<table>
<thead>
<tr>
<th>Projekt</th>
<th>Anmeldung</th>
<th>Abgabe bis</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.10. – 15.11.</td>
<td>15.01.</td>
<td></td>
</tr>
<tr>
<td>15.01. – 15.02.</td>
<td>15.04.</td>
<td></td>
</tr>
<tr>
<td>01.04. – 01.05.</td>
<td>30.06.</td>
<td></td>
</tr>
<tr>
<td>01.07. – 01.08.</td>
<td>30.09.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Abschlussarbeit</th>
<th>Anmeldung</th>
<th>Bearbeitungsbeginn</th>
<th>Abgabe bis*</th>
</tr>
</thead>
<tbody>
<tr>
<td>01.02. – 15.02.</td>
<td>01.03.</td>
<td>01.06.</td>
<td></td>
</tr>
<tr>
<td>01.05. – 15.05.</td>
<td>01.06.</td>
<td>01.09.</td>
<td></td>
</tr>
<tr>
<td>15.07. – 01.08.</td>
<td>15.08.</td>
<td>15.11.</td>
<td></td>
</tr>
<tr>
<td>15.10. – 02.11.</td>
<td>15.11.</td>
<td>01.03.</td>
<td></td>
</tr>
</tbody>
</table>

* ungefähre Angabe; der genaue Zeitpunkt wird vom ZPA festgelegt
Typical Generic Structure:

1. Introduction
   – motivation of work, state of the art, related work, contributions

2. Preliminaries
   – which facts / concepts / definitions / algorithms / approaches / methods does this work rely on (“standing on the shoulders of giants”)
   – i.e., any technical information that is needed but not developed in the course of this report / thesis

3. Approach
   – technical contribution of the thesis (concepts / definitions / algorithms / approaches / methods etc.)

4. Implementation
   – software that has been implemented

5. Evaluation
   – case studies, experiments, quantitative and qualitative assessment, etc.

6. Conclusion
   – what has been accomplished
   – future research directions

7. Bibliography
Formal Requirements

♦ Before you start your work
  ‣ submit written proposal (≈ 1-2 pages) to sen@uni-konstanz.de containing
    – the topic you want to choose
    – how well you match the prerequisites
    – schedule for the project / thesis
      • what will be achieved at which point in time
        * requires a careful break-down of the project / thesis topic into subgoals
      • when will the project / thesis be officially registered

♦ During your preparation of the project work / thesis
  ‣ regular consultation with your supervisor
    – approx. every 4 weeks
Deliverables

- project report to the supervisor
- thesis
  - must be submitted to the examination office
  - in parallel: electronic copy (pdf) to supervisor
- any models / code / data / binaries you created for the project
  - include in DVD attached to the thesis
  - in parallel: electronic copy to supervisor
Topic Areas for Projects and Theses

♦ I. Safety Analysis, Causality, Real-Time Systems and Repair
  ‣ Causality Checking
  ‣ Causality in DNNs
  ‣ Functional Safety of Automotive Systems
  ‣ QuantUM+: Model Based System Engineering, implementation of Causality Checking
  ‣ TarTar: Analysis and Automated Repair of Timed Systems
    – synthesis of repairs using SMT technology

♦ II. Legal Tech
  ‣ logical modeling and analysis of legal artefacts
  ‣ understanding legal contracts using Natural Language Processing

♦ III. Formal Verification and Repair for Machine Learning
  ‣ counterexample computation for DNNs
  ‣ automated repair of DNNs
  ‣ applications in health science
I. Safety Analysis, Causality, Real-Time Systems and Repair

- Causality Checking
- Causality in DNNs
- Functional Safety of Automotive Systems
- QuantUM+: Model Based System Engineering, implementation of Causality Checking
- TarTar: Analysis and Automated Repair of Timed Systems
  - synthesis of repairs using SMT technology

II. Legal Tech

- logical modeling and analysis of legal artefacts
- understanding legal contracts using Natural Language Processing

III. Formal Verification and Repair for Machine Learning

- counterexample computation for DNNs
- automated repair of DNNs
- applications in health science
SysML v2 Dynamic Semantics [M] {SL}

♦ Setting
  ‣ ongoing standardization of version 2 of the OMG SysML
    - [Link](https://www.omgsysml.org/SysML-2.htm)
  ‣ own meta-model and state-machine semantics independent from UML and SysML v. 1.x

♦ Research Questions
  ‣ study and understanding of the state-machine semantics of SysML V2
  ‣ study of the underlying "4D semantics" by Conrad Bock (NIST)
    - [Link](https://www.conradbock.org/bockonline.html)

♦ Goal
  ‣ devising a strategy for automated formal analysis of SysML V2 state machine models
  ‣ prototype tool / case study
  ‣ potential for link to QuantUM
Causality Checking for Symbolic Execution [M] {SL}

♦ Setting

- symbolic execution is the logical representation of program execution paths
- it is the foundation of many program analysis and testing methods, e.g., concolic testing, fuzzing, etc.
- tools (examples)
  - SAGE, https://queue.acm.org/detail.cfm?id=2094081

♦ Research Question

- study of the application of counterfactual causal analysis / Causality Checking to symbolic execution

♦ Goal

- algorithm and tool development, case study
Causality Checking for Software Model Checking [B,M] {SL}

♦ Setting

◊ software model checkers analyze programming language code (often C code) instead of modeling languages

◊ tools
  – CPAChecker https://cpachecker.sosy-lab.org/
  – Ultimate https://www.ultimate-pa.org/

♦ Research Question

◊ how can the Causality Checking approach be conceptually applied to software model checking
  – interpretation of counterfactuality in counterexamples and non-faulty executions

♦ Goal

◊ concept of Causality Checking for SMC

◊ prototypical tool development, case study
Causality Checking and Hyperproperties [B,M] \{SL\}

Setting

\(\checkmark\) Coenen et al. propose to use hyperproperties (properties of sets of traces) and Hyper LTL model checking to compute causalities

- [https://doi.org/10.1007/978-3-031-13185-1_20](https://doi.org/10.1007/978-3-031-13185-1_20)

- [https://doi.org/10.1007/978-3-031-19992-9_13](https://doi.org/10.1007/978-3-031-19992-9_13)

\(\checkmark\) Causality Checking as implemented in QuantUM relies on simple explicit-state model checking

Research Objective

\(\checkmark\) comparison of the capabilities of both approaches to compute actual causes / counterfactual causal explanations

\(\checkmark\) development of recommendations for a reconciliation of both approaches, if possible

\(\checkmark\) case studies
Symbolic Encoding of Causality Checking [M] {RS, SL}

♦ Causality Checking
  ‣ computing ordered sequences of events in a system model as being causal for reaching a dangerous system state (e.g., car and train in the railroad crossing)
  ‣ currently relies on explicit state space search
  ‣ bottleneck
    – number of traces to be stored

♦ Objective
  ‣ symbolic encoding of causality conditions using Binary Decision Diagrams (BDDs)
  ‣ computation of causes based on BDD libraries
  ‣ implementation in the QuantUM toolset

♦ Literature
I. Safety Analysis, Causality, Real-Time Systems and Repair
   - Causality Checking
   - Causality in DNNs
   - Functional Safety of Automotive Systems
   - QuantUM+: Model Based System Engineering, implementation of Causality Checking
   - TarTar: Analysis and Automated Repair of Timed Systems
     - synthesis of repairs using SMT technology

II. Legal Tech
   - logical modeling and analysis of legal artefacts
   - understanding legal contracts using Natural Language Processing

III. Formal Verification and Repair for Machine Learning
   - counterexample computation for DNNs
   - automated repair of DNNs
   - applications in health science

IV. Centre for Human | Data | Science
A State Machine Model for Contract Execution [B,M] {RS, SL}

♦ **Legal Tech**
  ‣ existing model of legal contract execution using strongest precondition semantics for claims
  ‣ encoded as propositional logic
  ‣ analyzed using SMT-solving / Contract Check

♦ **Goal**
  ‣ deriving a concurrent state machine model for the contract execution

♦ **Context**
  ‣ existing joint project with Prof. R. Wilhelmi, Department of Law
  ‣ existing tool ContractCheck

♦ **Reading**
  ‣ [https://doi.org/10.1007/978-3-031-15077-7_1](https://doi.org/10.1007/978-3-031-15077-7_1)
Legal Tech
- modeling of legal contracts
  - share purchase agreements
    - seller
    - purchaser
- they "play a game"
  - satisfy / not satisfy claims
  - execution on time / delayed
  - ...
- which moves put which player into an advantageous situation?
  - ⇒ game theory
- possible extensions
  - quantifiable loss / risk

Objective
- formulate contract execution as two-party game
Testing Logic Encodings of Sales Contracts [B,M] {DB, SL}

♦ Setting
  ‣ ContractCheck translates a Sales Purchase Agreement (SPA) into a logic encoding
  ‣ Encoding knowledge and processes in logic bears the danger of producing a faulty encoding

♦ Question
  ‣ How can we test an encoding in logic to gain confidence that it faithfully captures the encoded artefact?
    – Develop testing methods for gaining confidence in a logic encoding of an SPA
    – Important when encoding is provided automatically (e.g., by an ML model)

♦ Possible Directions
  ‣ Compute a diverse set of satisfying assignments of the logic encoding

♦ Literature
  ‣ https://doi.org/10.48550/arXiv.2212.03349 (ContractCheck)
Topic Areas for Projects and Theses

♦ I. Safety Analysis, Causality, Real-Time Systems and Repair
  ‣ Causality Checking
  ‣ Causality in DNNs
  ‣ Functional Safety of Automotive Systems
  ‣ QuantUM+: Model Based System Engineering, implementation of Causality Checking
  ‣ TarTar: Analysis and Automated Repair of Timed Systems
    – synthesis of repairs using SMT technology

♦ II. Legal Tech
  ‣ logical modeling and analysis of legal artefacts
  ‣ understanding legal contracts using Natural Language Processing

♦ III. Formal Verification and Repair for Machine Learning
  ‣ NN repair for cyber-physical systems
  ‣ verifying global specifications using cutting planes
  ‣ faster NN repair
  ‣ verification of Self-Driving cars
NN Repair for Cyber-Physical Systems [B,M] {DB, SL}

Setting

- Deep learning models can be embedded as controllers in safety-critical cyber-physical systems (autonomous driving, steering aircraft and spaceships).
- Such systems require safety guarantees

Question

- Repair DNN controllers using SpecRepair and Safety Critics
- Verification using JuliaReach, NNV, or others

Literature

- [https://doi.org/10.1007/978-3-031-15077-7_5](https://doi.org/10.1007/978-3-031-15077-7_5) (SpecRepair)
- [https://openreview.net/forum?id=iaO86DUuKi](https://openreview.net/forum?id=iaO86DUuKi) (Safety Critics)
Verifying Global NN Specifications using Cutting Planes [B] {DB, SL}

Description

- Many desirable properties of NNs apply for the entire input region (global specifications)
- Current NN verifiers are targeted at local specifications
- Recent development in verifiers: cutting planes (GCP-CROWN)
- Cutting planes generalize the ReluDiff approach for verifying specific global specifications
- Can cutting planes verify general global specifications?

Tasks

- Formalize global specifications in GCP-CROWN
- How far does this scale?
- Comparison with ReluDiff

Literature

- https://arxiv.org/abs/2306.12495 (Global Specifications)
- https://papers.nips.cc/paper_files/paper/2022/hash/0b06c8673eb3453e5e468f7743d8f54e-Abstract-Conference.html (GCP-CROWN)
- https://dl.acm.org/doi/10.1145/3377811.3380337 (ReluDiff)
Faster Neural Network Repair [B, M] \{DB, SL\}

Setting

- SpecRepair uses piecewise-linear violation functions to quantify counterexample violations
- These violation functions have certain drawbacks
  - Only gradient for the least-violated term is given for disjunction
  - Violation functions are not differentiable everywhere and, therefore, do not have a continuous gradient.

Question

- Are there different violation functions that can accelerate repair or improve network performance?
- Can we learn violation functions and improve repair?

Literature

- https://doi.org/10.1007/978-3-031-15077-7_5 (SpecRepair)
- http://proceedings.mlr.press/v97/fischer19a.html (DL2, has a different set of violation functions)
Verification of a Self-Driving Car [M] {DB, SL}

♦ Motivation
  ‣ neural networks are applied in self-driving cars where mistakes are fatal

♦ Identify Safety Constraints and Apply Them to the AI Training Procedure
  ‣ What safety constraints are important during driving?
  ‣ can the neural network made safe by altering the training procedure?
  ‣ develop a simulation of a self-driving car:
    – DeepDrive (https://deepdrive.io)
    – F1tenth (https://f1tenth.org)

♦ Prerequisites
  ‣ machine learning models and training algorithms
  ‣ interest in verification
Important

♦ Own Ideas Welcome!

› if you have own ideas
  – topics not included in our catalog
  – modifications of proposed topics

please talk to us!

• topic finding is an iterative, deliberative process!
Interested? Contact...

... either one of us at any time!

- Prof. Dr. Stefan Leue
  - Email: Stefan.Leue@uni.kn
- David Boetius
  - Email: David.Boetius@uni.kn
- Raffael Senn
  - Email: Raffael.Senn@uni.kn

or: sen@uni-konstanz.de
Questions
Causality Checking and Choice Functions
Setting

- Real-Time Systems can be modelled with Timed Automata (TA)
- designing a TA which is correct w.r.t. some property is hard
- TarTar can repair a faulty TA by removing a single counterexample.
  - This repair does not guarantee that the TA is correct

Objective

- Apply the repair computation of TarTar to a symbolic representation of the TA (k-normalized Zonegraph)
- Experimentally implement the approach and compare it with state of the art methods

Literature