Bachelor & Master Projects and Thesis

Prof. Dr. Stefan Leue

Software and Systems Engineering

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

WS 2020/2021
Members

- Prof. Dr. Stefan Leue {SL}
  - Email: Stefan.Leue@uni.kn.de, Room: PZ 902
- Dr. Georgiana Caltais {GC}
  - Email: Georgiana.Caltais@uni.kn.de, Room: PZ 913
- Martin Kölbl {MK}
  - Email: Martin.Koelbl@uni.kn.de, Room: PZ 912
- Fabian Marquart {FM}
  - Email: Fabian.Marquart@uni.kn.de, Room: PZ 912
Areas of Research Interest

♦ Formal Modeling and Analysis
  ‣ model checking
  ‣ SMT solving

♦ System Safety and Analysis
  ‣ causality in systems
  ‣ causality checking / QuantUM
  ‣ analysis and repair for timed system models
  ‣ safety and verification of machine-learning based systems

♦ Applications
  ‣ automotive software architectures
  ‣ functional safety of automotive software (ISO 26262)
  ‣ safety of autonomous driving (SOTIF ISO PAS 21448)
  ‣ legal tech
  ‣ modeling of collective behavior
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 industrial partners
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)
Project Report / Thesis Structure

♦ 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
        (proposal for this term ideally submitted by November 17, 2020)

♦ 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
Projects and Theses
System Safety and Analysis
Problem Statement

- Causality Checking in QuantUM relies on trace computation
  - bottleneck: memory
- LTSmin / PINS is interface to symbolic model checking engine
  - hope: more memory efficient

Project Tasks

- integrate QuantUM/Causality Checking into LTSmin / PINs
- direct logic encoding of causality checking in symbolic BDD data structures [MP/MT]

Prerequisites

- programming, discrete structures
- for [MP/MT]: (symbolic) model checking an advantage
Online SysML Checker [BP/BT] {MK}

♦ Motivation
  ‣ QuantUM and TarTar are complex
    – installation process, interoperability SysML and tools
    – correct input, interpretation of errors
  ‣ trial models and easily changeable models
  ‣ collaboration with tool Artest of Mindmotiv

♦ Aims
  ‣ integrate checking tools (QuantUM/TarTar)
    – in online SysML/Marte editor
    – use (existing) automatic model transformation by ATL
    – use microprocess architecture (RestFull_Api)
  ‣ depict results and error messages
  ‣ (informal) arguments that the tool chain preserves semantics (correctness)
Causality Checking for Programs [MP/MT] {SL}

◆ **Programs**
  - the assignment of certain values to variables can cause a program to crash
  - which variable assignments and which values are causal for a program failure?

◆ **Tasks**
  - selection of a program analysis framework (other than testing)
    - for instance, symbolic execution, static analysis
  - development of a causality notion for program executions
  - prototype implementation and case study

◆ **Prerequisites**
  - good understanding of logic, program semantics, foundations of computing
Motivation

- Causality Checking currently defined based on Event Order Logic
- HyperLTL is a logic that allows for quantification over traces
  - applications in security
- how can the definition of Causality Checking benefit from a definition using HyperLTL

Tasks

- definition of Causality Checking using Hyper LTL
- use of this definition for
  - use of HyperLTL model checking for detecting causalities
  - repair synthesis using HyperLTL synthesis

Prerequisites

- background in model checking and temporal logic, or willingness to acquire this
- general interest in logics and their algorithmic mechanization

---

HyperLTL

HyperLTL [8] is a temporal logic for specifying hyperproperties. It extends LTL by quantification over trace variables $\pi$ and a method to link atomic propositions to specific traces. The set of trace variables is $\mathcal{V}$. Formulas in HyperLTL are given by the grammar

$$\psi ::= \forall \pi. \psi \mid \exists \pi. \psi \mid \psi \land \psi \mid \neg \psi \mid P \psi \mid \psi \Rightarrow \psi \mid \psi \land \neg \psi \mid \psi \land \psi \mid \psi \land \psi \mid \psi \land \psi .$$

where $a \in \mathcal{AP}$ and $\pi \in \mathcal{V}$. The alphabet of a HyperLTL formula is $2^{\mathcal{AP}}$. We allow the standard boolean connectives $\land, \neg, \Rightarrow$ as well as the derived LTL operators release $\psi \Rightarrow \psi \equiv \neg (\neg \psi \land \neg \psi)$, eventually $\chi \psi \equiv true \psi \land \chi \psi$, globally $\Box \psi \equiv \neg \neg \psi$, and weak until $\psi \chi \psi \equiv \Box \psi \lor (\psi \land \chi \psi)$.  

---

Formal Modeling and Analysis
Robustness in Timed Systems [MP/MT] {MK, SL}

♦ Motivation
  ‣ clocks in realistic systems deviate from ideal clocks
    – what is the robustness of a timed system?

♦ Compute Possible Deviation ($\epsilon$)
  ‣ what kinds of deviation exist?
    – limited, additive, multiplier
  ‣ encode timed system
  ‣ find (semi-decidable) algorithm to compute $\epsilon$

♦ Related Work
  ‣ timed causes (→ paper)
  ‣ robustness in real-time systems

♦ Requirements
  ‣ real-time model checking
  ‣ advantageous: course on SMT solving
Motivation

- hybrid systems have continuous and discrete dynamic behavior
  - e.g. physical law
- causes in hybrid systems are not single-events but continuous effects
  - e.g. accident since a car broke too short, not hard enough or both?

Apply Causality on Hybrid Systems

- formally define causality for hybrid systems
- aim: an algorithm to compute causes

Related Work

- causality in timed systems
- hybrid systems models

Prerequisites

- advantage: advances model checking
Repair for Parametric Timed Automata [BP/BT] [MP/MT]

♦ Parametric Timed Automata
  ‣ literature survey
    – starting points
      • Minimal-Time Synthesis for Parametric Timed Automata, É. André et al., 2019
      • What’s decidable about parametric timed automata? É. André, 2015
    – tools? case studies?
  ‣ relationship to repair analysis
    – based on SMT solving
    – tool TarTar @UniKN
    – paper

♦ Prerequisites
  ‣ ideally, Advanced Model Checking, but not mandatory...
Model Transformation to LTSmin / PINS

- SysML models edited in Papyrus Real Time (EMF) or Modelio
- understand semantics of state machine diagrams, inter-object communication and meta-models
- define transformation rules in ATL model transformation framework
- implementation and case studies

Benefits

- exposure to practically very relevant model based design language

Prerequisites

- Software Engineering
- interest in semantics and model transformation
Run-Time Causality Checking [MP/MT] {SL}

- **Runtime Verification**
  - observe and assess running system
  - often: monitoring

- **Run-Time Causality Checking**
  - observe system behavior
  - detect occurrence of events at run-time as causal for undesired system behavior
  - learn for the future

- **Tasks**
  - study various run-time verification approaches, in particular run-time model checking
  - analyze, what causality can mean in this context
  - adapt causality checking

- **Prequelisites**
  - one model checking course
  - advantageous: machine learning, data mining
Causality Checking Concurrent Code [MP / MT] {SL}

◈ CHESS
   - tool for finding and producing bugs in concurrent programs (C,C++,C#)

◈ Task
   - transfer the idea of Causality Checking to testing / code analysis
   - apply it to the CHESS tool / environment
   - implement prototype
   - perform case studies
Formal Verification of Machine Learning
Robustness Analysis of Neural Networks [BP/BT] [MP/MT] {FM, SL}

♦ Motivation

   › Does the network perform robustly (i.e. is the classification the same for similar images?)

♦ Develop a measurement technique for evaluating a neural network‘s robustness based on SHGO.

   › Come up with a method on how to modify DeepOpt’s specification parameter.
   › Use the MNIST dataset and the pre-trained networks from the DeepPoly paper as input to your algorithm.
   › First, determine general robustness concerning different epsilon environments, and consequently, robustness concerning perturbed input.
   › For both experiments, give accurate percentages of robustness for the different values of epsilon.

♦ Prerequisites

   › Machine learning models and training algorithms
   › Interest in verification

<table>
<thead>
<tr>
<th>Attack</th>
<th>Original</th>
<th>Lower</th>
<th>Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_\infty$</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>Rotation</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

Singh, 2018
Automatic Repair of Neural Networks [MP / MT] {FM, SL}

♦ Motivation
  ‣ Can a neural network that has errors be repaired?
  ♦ Develop an algorithm that can repair a neural network with regards to specification violations.
    ‣ After DeepOpt puts out a set of counterexamples for a given network and a specification, your algorithm should use these counterexamples to improve or repair the network
    ‣ This can be done by either retraining or by embedding the network inside a system with input filters
    ‣ Evaluate the improvement using a complete verification algorithm such as DeepPoly
    ‣ Use the MNIST or HorizontalCAS datasets for evaluation.

♦ Prerequisites
  ‣ Machine learning models and training algorithms
  ‣ Interest in verification
Causality Checking of Neural Network Errors [MP / MT] {FM, SL}

♦ Motivation
  ‣ Can errors in a neural network be explained?
  ♦ When a specification is written that a neural network has to conform to, it is important to know which samples in the training data are responsible if such a specification is violated.
    ‣ Develop an algorithm that uses DeepOpt’s output which is a set of counterexamples to detect their cause in the training data
    ‣ By removing the points close to the counterexamples from the training data and training a new network, determine if the error persists by checking the new network with DeepOpt and DeepPoly.

♦ Prerequisites
  ‣ Machine learning models and training algorithms
  ‣ Interest in verification

Ranjan, 2019
Topic Area

Applications
Modeling Collective Behavior [MP/MT] {SL}

♦ Objective
  ‣ assess the suitability of formal description techniques to model emergent behavior of biological collectives
    – look at stochastic model checking
    – look at spatio-temporal logics
  ‣ context
    – cluster Center for the Advanced Study of Collective Behavior
  ‣ perform concrete case study
    – cooperate with the Jordan Lab @ Uni KN (biology / Max-Planck)
      • behavior of schools of fish
    – select method and tool

♦ Prerequisites
  ‣ willingness to perform inter-disciplinary research
  ‣ readiness to familiarize oneself with formal modeling and analysis techniques / tools
  ‣ model checking courses are an advantage, but not a must...
Analysis Methods for SOTIF [MP/MT] {SL, FM}

- Safety of the Intended Function
  - ISO PAS 21448 SOTIF
  - how to ensure safety of an autonomous vehicle in the presence of malfunctioning machine learning components

- Objective
  - analysis, where semi-formal and formal modeling and analysis can be used in SOTIF
  - approaches to be considered – e.g., probabilistic contracts

- Prerequisites
  - interest in machine learning
  - ideally combined with seminar "Formal Verification and Testing of Deep Neural Networks"
Explaining Faults with Machine Learning [BP/BT][MP/MT] {SL, FM}

- Explaining Faults with Machine Learning
  - Model Checking: thousands of counterexamples with hundreds of events
  - Use Machine Learning (ML) to learn & explain counterexamples
    - Learn compact representation of counterexamples
    - Extract patterns

- Goals
  - Tool to explain a set of counterexamples of Model Checker (SPIN)
    - Implementation in Python
    - Decision Tree Learning (scikit learn)
  - Case study on selected Promela Models

- Requirements
  - Strong programming skills (e.g. C++, Java, Python)
  - Ability to work independently
  - Basic Machine Learning knowledge
Non-Formal Applications
Survey on Sustainable Software (BP/BA) {MK, FM}

♦ Motivation
  ‣ problem: software becomes a throwaway product (outdates fast)
    – cost resources, e.g., man-power
    – how green (e.g., CO₂ footprint) is software?

♦ Project
  ‣ Survey on Sustainability and Greenness of Software
    – definitions of sustainability and greenness
    – how to quantify these properties?
    – is sustainability/green performances bad? If so, are there reasons?
      • e.g. dependency on libraries, security updates, inefficient algorithms, missing development principles, time bound of development, …

♦ Thesis
  ‣ possibilities to improve software (development)?
    – how to detect low sustainability and green performance
    – how can software engineering improve development?
    are new principles necessary?
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!
... either one of us any time!

- Prof. Dr. Stefan Leue  
  - Email: Stefan.Leue@uni.kn
- Martin Kölbl  
  - Email: Martin.Koelbl@uni.kn
- Fabian Marquart  
  - Email: Fabian.Marquart@uni.kn.de