Bachelor & Master Projects and Thesis

Prof. Dr. Stefan Leue

Software and Systems Engineering

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

Winter Term 2021/22
Members

- Prof. Dr. Stefan Leue {SL}
  - Email: Stefan.Leue@uni.kn, Room: PZ 902
- Dr. Georgiana Caltais {GC}
  - Email: Georgiana.Caltais@uni.kn, Room: PZ 913
- Martin Kölbl {MK}
  - Email: Martin.Koelbl@uni.kn, Room: PZ 912
- Fabian Bauer-Marquart {FBM}
  - Email: Fabian.Marquart@uni.kn, Room: PZ 912
Projects at our Chair

- Safety Analysis, Fault Localization and Causality
  - causality checking, Functional Safety of Automotive Systems
- Analysis and Automated Repair of Timed Traces
  - synthesis of repairs using SMT technology
- QuantUM and QuantUM+
  - Model Based System Engineering, implementation of Causality Checking
- Architectures for Automotive Systems
  - HW/SW Architectures for Autonomous Driving
- Formal Verification for Machine Learning
  - quality assurance for ML-based systems
- Formal Verification for Quantum Computing
  - Automated quantum program correctness proofs
- Computational Methods in Systems Biology
  - formal explanatory modeling of collective behavior
- Legal Tech
  - logical modeling and analysis of legal artefacts
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
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)

› **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)
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 18, 2021)

♦ 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

Topic Areas

- Topic I: System Safety and Analysis
- Topic II: Formal Modeling and Analysis
- Topic III: Formal Verification of Machine Learning
- Topic IV: Quantum Computing and Formal Verification
- Topic V: Applications
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

Prerequisites

- programming, discrete structures
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
Causality Checking in HyperLTL [MP/MT] {SL}

**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

\[
\varphi ::= \forall \pi. \varphi \mid \exists \pi. \varphi \\
\psi ::= a_\pi \mid \neg \psi \mid \psi \lor \psi \\
\phi ::= (\psi \triangleright \psi')
\]

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 \(\land, \lor, \rightarrow\) as well as the derived LTL operators release

\[
\varphi \triangleright \psi = (\neg \psi \land \neg \psi)
\]

eventually \(\Diamond \varphi = \text{true} \lor \varphi\), globally \(\Box \varphi = \neg \Diamond \neg \varphi\), and weak until

\[
\varphi \text{ until } \psi = \Box \varphi \lor (\varphi \triangleright \psi)
\]

---

Causal Explanations of Accidents [MA]

♦ Motivation
  ‣ What can we learn from real system experience?

♦ Experience
  ‣ QuantUM computes causes for property violations
  ‣ model F1/10 of self driving car ([https://f1tenth.org](https://f1tenth.org))
  ‣ process idea
    1. QuantUM computes causes for system failures
    2. monitor real system execution until system fails
    3. determine the actual cause for system failure

♦ What Kind of a Cause Explains an Accident?
  ‣ Which system details are necessary for cause analysis?
  ‣ How to monitor execution to determine the actual cause?
Case Study on Fail-Operation Mode [BA]

- **Preceeding Work**
  - architecture of autonomous car (ADS) [FMICS]
  - a *safe* self driving car is fail-operational
    - in case a component fails
      the car still stops on shoulder

- **Task**
  1. realize implementation of architecture on car F1/10 ([https://f1tenth.org/](https://f1tenth.org/))
  2. execute cause to test fail-operational mode

- **Questions**
  - How realistic is the ADS architecture?
  - What data flow is in the system?
Formal Modeling and Analysis
Causality in Hybrid Systems [MP/MT] {MK, SL}

♦ 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...
Multiple Constraint Relaxation for Timed Systems [MP/MT]

♦ Motivation
  ‣ real-time constraints can be unnecessarily tight

♦ Analysis to Relax Time Constraints
  ‣ encode timed system (trace) in logic formalism
    – e.g., SMT2
  ‣ encode analysis candidates \((\Delta_1, \Delta_2, \Delta_3\ldots)\)
  ‣ check for weaker constraints
    – multi-objective optimization
    – solve pareto optimization problem
  ‣ is the functionality of the system changed?

♦ Related Work
  ‣ logical modeling of real-time systems
  ‣ optimization

♦ Requirements
  ‣ interest in exploring multi-objective optimization and related tools
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

♦ Prequequisites
  ‣ one model checking course
  ‣ advantageous: machine learning, data mining
Formal Verification of Machine Learning
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
Counterfactual Causality in Neural Networks [M] {FBM, SL}

♦ Motivation
  ‣ neural networks lack the ability to recognize or react to new circumstances they have not been specifically programmed or trained for.

♦ In interpretable machine learning, counterfactual explanations can be used to explain predictions of individual instances.
  ‣ What is the causality of prediction outcomes?
  ‣ for an unexpected prediction, identify and explain the causality for this prediction.
  ‣ define causality explanation in terms of:
    – Network structure
    – Input data
    – Training data

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

Mothilal, 2020
Verification of a Self-Driving Car AI [M] {FBM, 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
Formal Verification of Quantum Computing
Verification & Repair of Quantum Programs [M] {FBM, SL}

- **Motivation**
  - Quantum algorithms follow a different paradigm than classical algorithms
  - it is more difficult to debug them or prove their correctness

- **Make our Specification Verification Technique More Scalable**
  - Quantum computing and qubits are mathematically grounded in linear algebra
  - quantum algorithm is encoded as an SMT-formula using vector, matrix and tensor logic
    - the state space of a qubit is infinite
    - state vectors grow exponentially with the number of qubits
  - goal: Design a more abstract encoding

- **Prerequisites**
  - interest in quantum computing
  - interest in verification
Motivation
- Machine Learning is applied to Quantum Computers
- Quantum convolutional neural networks (QCNNs) can classify classical image and quantum data

Safety Verification
- define meaningful safety specifications for quantum classifiers
- case study:
  - implement in *PyTorch* and *Qiskit*
  - evaluate technique for different types of data and model

Prerequisites
- interest in quantum computing
- some knowledge in machine learning
- interest in verification
Quantum Approach to Software Verification [M] {FBM, SL}

♦ **Motivation**
  ‣ Quantum Computers are more powerful than classical computers
  ‣ they solve SAT problems more efficiently
  ‣ software verification is a complex problem that can be mapped to SAT

♦ **Quantum-based Software Verification**
  ‣ identify benchmarks for bounded model checking (BMC)
  ‣ design a quantum BMC algorithm (based on SAT solving)
  ‣ evaluate its complexity

♦ **Prerequisites**
  ‣ interest in quantum computing
  ‣ some knowledge in machine learning
  ‣ interest in verification
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...
Logical Analysis of Sales Contracts [MP/MT]

Joint work with Prof. RüdigerWilhelmi, Dept. of Law, Univ. KN

♦ Motivation
  ‣ sale contracts for companies need to be self-consistent
  ‣ vulnerable to inconsistencies because of references and dependencies
  ‣ aim: find inconsistencies automatically

♦ Approach (LegalTech)
  ‣ transform contract to a logical model
  ‣ check for semantic and syntactic inconsistencies using logic reasoning tools

♦ Requirements
  ‣ basic logic
  ‣ good command of German language

A. Object of Agreement
Person A sells Person B the company Bakery Gmbh.

B Sale Conditions
§1 Purchase Price
Person B pays 50.000€ within 4 weeks after the sale date.

…

§5 Warranties
Person A ensures the company has a capacity to bake 10.000 pretzels a day.

§6 Claims
In case the company has not the capacity ensured in §5, Person B has to send a written claim within 2 weeks after the sale date.

§7 Legal Consequences
Person A has to remove any claim within 4 weeks else the price defined in §1 is reduced.
Explaining Faults with Machine Learning [BP/BT][MP/MT] {SL, FBM}

♦ 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
For BA 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></td>
<td>15.10. – 15.11.</td>
<td>15.01.</td>
</tr>
<tr>
<td></td>
<td>15.01. – 15.02.</td>
<td>15.04.</td>
</tr>
<tr>
<td></td>
<td>01.04. – 01.05.</td>
<td>30.06.</td>
</tr>
<tr>
<td></td>
<td>01.07. – 01.08.</td>
<td>30.09.</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></td>
<td>01.02. – 15.02.</td>
<td>01.03.</td>
<td>01.06.</td>
</tr>
<tr>
<td></td>
<td>01.05. – 15.05.</td>
<td>01.06.</td>
<td>01.09.</td>
</tr>
<tr>
<td></td>
<td>15.07. – 01.08.</td>
<td>15.08.</td>
<td>15.11.</td>
</tr>
<tr>
<td></td>
<td>15.10. – 02.11.</td>
<td>15.11.</td>
<td>01.03.</td>
</tr>
</tbody>
</table>

* ungefähre Angabe; der genaue Zeitpunkt wird vom ZPA festgelegt
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 at any time!

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

or: sen@uni-konstanz.de
Questions