Bachelor & Master Projects and Theses
Chair for Software and Systems Engineering

Prof. Dr. Stefan Leue

University of Konstanz
Chair for Software Engineering

Stefan.Leue@uni-konstanz.de
http://www.se.inf.uni-konstanz.de

Winter Term 2017/2018
We offer

- ... integration in state of the art research projects
- ... if possible collaboration with our industrial partners
- ... interesting real-life applications and problems
Scope of Projects/Theses

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

- **Thesis (Bachelor)**
  - 3-6 Months ("1 Semester")
  - 12 (Thesis) + 3 (Colloquium) ECTS (450h work)

- **Thesis (Master)**
  - 6 Months (1 Semester)
  - 30 (Thesis + Colloquium) ECTS (900h work)
Formal Requirements, Deliverables

- **Before you start your work**
  - Written proposal (≈ 1 page) containing (deadline: 2. November 2017)
    - the *topic* you want to choose
    - how well you *fit the prerequisites*
    - *why you are suited* for the task.
  - Schedule for the project/thesis
    - What should be achieved at which point in time.

- **During your preparation of the thesis or the project work**
  - Regular consultation with your supervisor.

- **Deliverables**
  - Any models/code/data/binaries you created for the project.
  - Report of ≥ 20 pages (Projects, Bachelor-Thesis) or ≥ 40 pages (Master-Thesis)
    - Discussing state of the art
    - Stating the problem
    - Presenting your approach and results
    - Critical discussion
Our Team

♦ Staff
  ‣ Prof. Dr. Stefan Leue
    – Email: Stefan.Leue@uni.kn, Room: PZ 902
  ‣ Dr. Georgiana Caltais
    – Email: Georgiana.Caltais@uni.kn, Room: PZ 913
  ‣ Martin Kölbl
    – Email Martin.Koelbl@uni.kn, Room: PZ 912
Safety-Critical Cyber-Physical Systems

♦ Our Research Focus
  ‣ design methods in support of the development of dependable systems
Verification of Safety-Critical Systems

- A Railroad Crossing
  - safety goal
    "It shall always be the case that there is never a car and a train in crossing at the same time."
  - can hazards happen? What is the cause?

Credits: Hanser / Südkurier
Possible Approach – Model Checking

- Model of The Real World – Transition Systems:

\[
\begin{align*}
\text{Train} & \quad \text{Ta} \quad \text{approaching} \\
\text{Train} & \quad \text{Ti} \quad \text{inside} \\
\text{Train} & \quad \text{Tc} \\
\text{Gate} & \quad \text{Gc} \\
\text{Gate} & \quad \text{Go} \\
\text{Car} & \quad \text{Ca} \quad \text{approaching} \\
\text{Car} & \quad \text{Cl} \quad \text{inside} \\
\text{Car} & \quad \text{Cc} \\
\end{align*}
\]
Possible Approach – Model Checking

- Model Checking

Model Checking

\[ M \models S \]

**model of the software**

- (transition system, Kripke structure)

**model checking algorithm**

**requirement specification**

- (assertions, temporal logic)

- **“car and train never in crossing at the same time”**

\[ \varphi = \Box \neg (Tc \land Cc) \]
Model Checking

- **Explicit State Model Checking**
  - most common: automatic search of all reachable system states to find property violations using **Depth First Search**
  - the path into a property violating state is called an **error path** or **counterexample**
Causality Checking

“car and train never in crossing at the same time”

- Model checking – (finite) counterexamples:

  \[
  \begin{align*}
  & [Ta, Gf, Tc, Ca, Cc] \\
  & [Ca, Ta, Gf, Tc, Cc] \\
  & [Ta, Gf, Ca, Cc, Tc] \\
  & [Ta, Ca, Gf, Cc, Tc] \\
  & [Ca, Ta, Gf, Cc, Tc] \\
  & [Ta, Ca, Cc, Gf, Tc] \\
  & [Ca, Ta, Cc, Gf, Tc] \\
  & [Ca, Cc, Ta, Gf, Tc] \\
  & [Ta, Ca, Cc, Gc, Tc] \\
  & [Ca, Ta, Cc, Gc, Tc] \\
  & [Ca, Cc, Ta, Gc, Tc] \\
  \end{align*}
  \]

  ... 

  - representation well-known to engineers...

- but, what is a cause?
  - algorithmic solution
“Engineer Friendly”

- Model of The Real World
  - SysML / UML specification
The QuantUM Approach

(Failure Mode) Specification → QuantUM Model

(Safety-) Requirements → automatic

QuantUM Model → automatic

Fault Trees
“Real World” Applications

- Experiments for **safety** properties include:
  - Airbag (with ZF/TRW Radolfzell)
  - Automotive Body Controller (AECU, with a German OEM)
  - Airspace Surveillance Radar (ASR), 3 variants (with Airbus Defense and Space)
Pattern Mining for Fault Localization [BP/BT/MP/MT]

- **Counterexamples in Model Checking**
  - there can be thousands of counterexamples with hundreds of events

- **Fault Localization**
  - use sequential pattern mining to find patterns in counterexamples that
    - occur frequently in "good" counterex.
    - occur infrequently in "bad" counterex.

- **Goals**
  - some existant seq. pattern mining algorithms
  - implementation and link to model checking
  - case studies

- **Collaboration**
  - with **PD Dr. Christian Borgelt**, Bio-Informatics and Information Mining
Real-Time Systems
- correctness depends on relative timing of computation steps
- timed automata
- UPPAAL model checker

Research Question
- how can causality checking be extended to real-time properties?
- what is a meaningful causality concept in this setting?

Project/Thesis Tasks
- how do counterexamples in real-time model checking look like?
- development of a causality concept for real-time properties
- implementation of this concept, case studies

Prerequisites
- Advanced Model Checking
Causality Checking for Programs [MP/MT]

- 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 for Deadlock Properties [BP/BT] [MP/MT]

- **Deadlocks**
  - circular wait, no more progress

- **Causality Checking for Deadlocks**
  - deadlock is reachability [BP/BT]
    - adopt causality checking to deadlock
    - implement in SpinJa
  - consider different deadlocks
    - extend the algorithmics and implementation of causality checking to multiple deadlocked states
  - implementation and case studies

- **Prerequisites**
  - preferably, one of the model checking courses
  - good programming skills
Run-Time Causality Checking [MP/MT]

- **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

- **Prequeuisites**
  - one model checking course
  - advantageous: machine learning, data mining
Model Transformation

- UML-RT models edited in Papyrus Real Time (EMF)
- 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
- interest in Papyrus community

Prerequisites

- Software Engineering
- interest in semantics and model transformation
Model Transformation
- SysML models edited in Papyrus Real Time (EMF)
- 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
- interest in Papyrus community

Prerequisites
- Software Engineering
- interest in semantics and model transformation
Causality Checking Concurrent Code [MP / MT]

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
Develop a graphical representation of EOL (event order logic)

- EOL (event order logic) is a subset of LTL (linear temporal logic) including ordering constraints.
- Currently, EOL is represented by formulae.
  \[(T_a \land (C_a \land C_c)) \land_t (\neg C_l \land_r T_c)\]
- Graphical representations are better suited for non-specialists, e.g.:

Prerequisite:

- Knowledge of LTL, e.g., course: Model Checking of Software and Systems, is an advantage but not strictly required
Interested? Contact...

... either one of us any time!

- Prof. Dr. Stefan Leue
  - Email: Stefan.Leue@uni.kn, Room: PZ 902
- Dr. Georgiana Caltais
  - Email: Georgiana.Caltais@uni.kn, Room: PZ 913
- Martin Kölbl
  - Email Martin.Koelbl@uni.kn, Room: PZ 912
Combine QuantUM with SAML

- The XXX (SAML) ...
- The goal of this project is to develop a translation from QuantUM to SAML and to integrate the SAML analysis tools into the QuantUM tool-chain.

Prerequisites:
- Either Model Checking of Software and Systems or Advanced Model Checking
- (Java) Programming