Link: https://udue.de/dcon2020prog

	Thursday, March 12	
	Room 1 (ground floor)	Room 2 (first floor)
8:30-8:50	Registration	
8:50-9:00	Welcoming Speech	
9:00-10:00	**Sebastian Wolff (TU Braunschweig)** Pointer Life Cycle Types for Lock-Free Data Structures with Memory Reclamation	
10:00-10:30	**Coffee Break**	
10:30-11:15	**Kim Völlinger (TU Berlin)** Certifying Distributed Algorithms	**Christina Mika-Michalski (Universität Duisburg-Essen)** Explaining Non-Bisimilarity in a Coalgebraic Approach: Games and Distinguishing Formula
11:15-12:00	**Sebastian Kenter (WWU Münster)** Generation of graphs expressing lock-sensitive reachability	**Thorsten Wißmann (FAU Erlangen-Nürnberg)** Generic and Efficient Partition Refinement
12:00-13:00	**Lunch**	
13:00-13:30	**Benjamin Bisping (TU Berlin)** Counterfactuals, Dispositions, and Hyperproperties	**Peter Chini (TU Braunschweig)** Parameterized Complexity of Program Verification Tasks
13:30-14:15	**Jens Gutsfeld (WWU Münster)** A Fixpoint Calculus for Asynchronous Hyperproperties	**Rebecca Bernemann (Universität Duisburg-Essen)** Modeling Probabilistic Transitions on Condition/Event Nets using Bayesian Networks
14:15-15:00	**Christoph Ohrem (WWU Münster)** Propositional Dynamic Logic for Hyperproperties	**Lars Stoltenow (Universität Duisburg-Essen)** Conditional Bisimilarity for Reactive Systems
15:00-15:30	**Coffee Break**	
15:30-15:45	**Edgar Arndt (TU Berlin)** Race Conditions in Population Protocols	
15:45-16:15	**Joshua Moerman (RWTH Aachen University)** Query Learning for Register Automata	
16:15-17:00	**Seyed Hossein Haeri (UCLouvain)** Piecewise relative observational purity	**Richard Eggert (Universität Duisburg-Essen)** Fixpoint Theory - Upside-Down

	Friday, March 13
	Room 1 (ground floor)
9:00-10:00	**Joost-Pieter Katoen (RWTH Aachen)** On The Automated Synthesis of Probabilistic Models
10:00-10:30	**Coffee Break**
10:30-11:15	**Sören van der Wall (TU Braunschweig)** Synthesis of concurrent, recursive Programs: Complexity of Multi-Pushdown Games
11:15-11:30	**Tobias Heindel (TU Berlin)** What if Nash equilibria are too expensive?
11:30-12:45	**Lunch**
12:45-13:30	**Business Meeting**

Sebastian Wolff (TU Braunschweig)
Pointer Life Cycle Types for Lock-Free Data Structures with Memory Reclamation

We consider the verification of lock-free data structures that manually manage their memory with the help of a safe memory reclamation (SMR) algorithm. Our first contribution is a type system that checks whether a program properly manages its memory. If the type check succeeds, it is safe to ignore the SMR algorithm and consider the program under garbage collection. Intuitively, our types track the protection of pointers as guaranteed by the SMR algorithm. There are two design decisions. The type system does not track any shape information, which makes it extremely lightweight. Instead, we rely on invariant annotations that postulate a protection by the SMR. To this end, we introduce angels, ghost variables with an angelic semantics. Moreover, the SMR algorithm is not hard-coded but a parameter of the type system definition. To achieve this, we rely on a recent specification language for SMR algorithms. Our second contribution is to automate the type inference and the invariant check. For the type inference, we show a quadratic-time algorithm. For the invariant check, we give a source-to-source translation that links our programs to off-the-shelf verification tools. It compiles away the angelic semantics. This allows us to infer appropriate annotations automatically in a guess-and-check manner. To demonstrate the effectiveness of our type-based verification approach, we check linearizability for various list and set implementations from the literature with both hazard pointers and epoch-based memory reclamation. For many of the examples, this is the first time they are verified automatically. For the ones where there is a competitor, we obtain a speed-up of up to two orders of magnitude.

Joost-Pieter Katoen (RWTH Aachen)
On The Automated Synthesis of Probabilistic Models

Probabilistic model checking focuses on checking whether a given (mostly finite) probabilistic model satisfies a given specification. Efficient algorithms and powerful tools such as PRISM and STORM exist and have been applied to various case studies. These techniques do require however the existence of a probabilistic model.
In this talk, I will focus on the automatic synthesis of Markov chains that satisfy a specification. This includes program sketching: given a probabilistic program with "holes", fill these "holes" such that the resulting program satisfies the specification.
I will outline two iterative approaches: counterexample-guided abstraction refinement (CEGAR) and counterexample-guided inductive synthesis (CEGIS). The "top-down" CEGAR approach partitions the design space starting from an abstraction of this space and re fines this by a light-weight analysis of veri fication results. The "bottom-up" CEGIS technique exploits critical subsystems as counterexamples to prune all programs behaving incorrectly on that input.
We apply these techniques to sketching of probabilistic programs, controller synthesis of POMDPs, and analysis of software product lines.

Kim Völlinger (TU Berlin)
Certifying Distributed Algorithms

While testing is important in practice, it offers no mathematical correctness. Formal verification guarantees complete correctness but is often too costly. Runtime verification stands between the two methods as a often less costly method that offers mathematical correctness. Runtime verification answers the question whether an input-output pair is correct. A certifying algorithm convinces its user at runtime by offering a correctness argument. This is appealing for outsourced computations where the user has no insight on the algorithm and does not want to trust the developers blindly. Certifying sequential algorithms are well-established. Distributed algorithms, designed to run on distributed systems, behave fundamentally different from sequential algorithms: their output is distributed the system or they even run continuously. We investigate certifying distributed algorithms.

Sebastian Kenter (WWU Münster)
Generation of graphs expressing lock-sensitive reachability

Reachability problems for parallel programs in the presence of locks are close to the border of decidability. We consider a variant of dynamic pushdown networks as a novel and flexible model for parallel programs with recursion and unboundedly many locks. Exploiting the decidability of monadic second-order satisfiability on a set of graphs generated by a hyperedge replacement grammar (HRG), we show that reachability is decidable for this model. To this end, we use augmented execution trees that capture the locking behaviour, and state an HRG that generates those graphs. While our approach offers an easy access to the problem, it also poses some challenges, and we will explain in this talk how to address them.

Christina Mika-Michalski (Universität Duisburg-Essen)
Explaining Non-Bisimilarity in a Coalgebraic Approach: Games and Distinguishing Formula

Behavioural equivalences can be characterized via bisimulation, modal logics, and spoiler-duplicator games. In this paper we work in the general setting of coalgebra and focus on generic algorithms for computing the winning strategies of both players in a bisimulation game. The winning strategy of the spoiler (if it exists) is then transformed into a modal formula that distinguishes the given non-bisimilar states. The modalities required for the formula are also synthesized on-the-fly, and we present a recipe for re-coding the formula with different modalities, given by a separating set of predicate liftings. Both the game and the generation of the distinguishing formulas have been implemented in a tool called T-BEG.

Thorsten Wißmann (FAU Erlangen-Nürnberg)
Generic and Efficient Partition Refinement

I present a generic partition refinement algorithm that quotients coalgebraic systems by behavioural equivalence, an important task in system analysis and verification. Coalgebraic generality allows to cover not only classical relational systems but also, e.g. various forms of weighted systems and furthermore to flexibly combine existing system types. Under assumptions on the type functor that allow representing its coalgebras in terms of nodes and edges, the algorithm runs in time $O(m \cdot log\ n)$ for input coalgebras with n states and m edges. The generic complexity result and the possibility of combining system types yields a toolbox for efficient partition refinement algorithms. Instances of our generic algorithm match the run-time of the best known algorithms for unlabelled transition systems, Markov chains, deterministic automata, Segala systems, and for color refinement. For weighted tree automata, the instance of the generic algorithm even improves the best run-time known. The algorithm is implemented in a tool (CoPaR) in full generality and allows the composition of existing systems types at run-time.

Benjamin Bisping (TU Berlin)
Counterfactuals, Dispositions, and Hyperproperties

We give an introduction to David Lewis' modal logic of counterfactual conditionals and its links to concurrency theory. Since the 1970s, the theory of counterfactuals interpreted over "spheres" of possible worlds has been extremely influential with analytic philosophers trying to explain causality and reactive properties of objecs (so-called "dispositions"). The foundations of software engineering and concurrency theory are often concerned with similar topics. So, it is worthwile to have a look at links between the fields, for example, in the research on temporal hyperproperties.

Jens Gutsfeld (WWU Münster)
A Fixpoint Calculus for Asynchronous Hyperproperties

Hyperproperties have received increasing attention in the last decade due to their importance e.g. for security analyses. Recently, several temporal logics were introduced to verify these hyperproperties via model checking. However, previous logics have in common that they focus only on synchronous hyperproperties, i.e. properties which relate paths to each other stepwise. In this paper, we introduce the logic $H\mu$, an extension of the linear time μ-calculus for hyperproperties without this restriction. While past μ-calculi for hyperproperties have focussed on extending the modal μ-calculus by the ability to consider multiple states simultaneously, we take a different approach and extend the linear time μ-calculus by quantification over named paths. This way we obtain a calculus that subsumes several previous temporal logics for hyperproperties. We show that our calculus can express a variety of interesting properties not expressible by other hyperlogics. Indeed, the high expressivity of $H\mu$ is evidenced by the fact that the satisfiability problem and even the model checking problem for it are undecidable. Accordingly, we exhibit fragments of $H\mu$ with decidable model checking and satisfiability problems and use the algorithms thus developed for underapproximation analyses of the full logic. We also establish a deep connection between $H\mu$ and a certain type of alternating asynchronous automata, which are of independent interest.

Christoph Ohrem (WWU Münster)
Propositional Dynamic Logic for Hyperproperties

Information security properties of reactive systems like non-interference often require relating different executions of the system to each other and following them simultaneously. Since common logics like LTL, CTL, or the modal μ-calculus cannot express such hyperproperties, the hyperlogics HyperLTL and HyperCTL* were developed to cure this defect. However, these logics are not able to express arbitrary ω-regular properties. In this paper, we introduce HyperPDL-Delta, an adaptation of the Propositional Dynamic Logic of Fischer and Ladner for hyperproperties, in order to remove this limitation. Using an elegant automata-theoretic framework, we show that HyperPDL-Delta model checking is asymptotically not more expensive than HyperCTL* model checking, despite its vastly increased expressive power. We further investigate,fragments of HyperPDL-Delta with regard to satisfiability checking.

Peter Cini (TU Braunschweig)
Parameterized Complexity of Program Verification Tasks

Decision problems arising in the context of program verification are usually hard. Results on worst-case complexities show that these problems are computationally involved and therefore expensive to solve. However, modern verification tools still perform well even on large instances. This shows a discrepancy between theory and practice which occurs for a particular reason. Worst-case complexity is measured in the size of the input without making further assumptions on the structure of the instance. Tools optimize their computation along the structure.
In order to resolve the discrepancy, we follow a recent trend in complexity theory: Parameterized Complexity. It allows for measuring the complexity of a problem in certain parameters that describe the structure of the instance in more detail. This offers a more fine-grained view into the complexity, giving us insights about which parameters make a problem hard or tractable. Of particular interest are fixed parameter tractable (FPT) problems. These are almost solvable in polynomial time. The only non-exponential part in the complexity depends on a parameter that is assumed to be small in practical instances.
We conducted several parameterized complexity analyses of verification problems. In the talk, we give an overview of the results obtained from this line of research. The main findings are: (1) A separation of tractable and intractable parameters for the considered problems. (2) New verification algorithms showing that various parameterizations of these problems are FPT. (3) New lower bounds, based on the exponential time hypothesis, that prove optimality of the found algorithms. Moreover, we observed that FPT results can be obtained by following a rough procedure: First, one needs to find a suitable normalform to compactly represent computations of a complex system. A common trick in verification. Then, algorithmic techniques from Parameterized Complexity are applied to test for the existence of a computation in normalform. These techniques are usually related to dynamic programming and determine the time consumption of the algorithm. Altogether, it shows that techniques from both fields, Parameterized Complexity and verification, work very well together.

Rebecca Bernemann (Universität Duisburg-Essen)
Modeling Probabilistic Transitions on Condition/Event Nets using Bayesian Networks

This talk presents extended Bayesian networks (BNs) that allow for dynamic changes to the probability distribution represented by BNs. In the proposed context they are utilized to model the observer's knowledge about the token distribution within a condition/event net in order to detect the current marking. The observer can study the net by monitoring successful transitions.
A mechanism is described enabling probabilistic transitions, i.e. making it possible to model the observation of a successful transition with respect to a probability distribution on transitions, thus introducing an additional source of uncertainty. This makes it possible to apply this model to a wider range of applications.
We use the notion of PROPs to obtain a compositional semantics which provides a convenient means to describe structural updates of networks. In order to demonstrate the performance improvement of our approach, we developed a C++ implementation that models the theoretical concepts.

Lars Stoltenow (Universität Duisburg-Essen)
Conditional Bisimilarity for Reactive Systems

Reactive systems à la Leifer and Milner, an abstract categorical framework for rewriting, provide a suitable framework for deriving bisimulation congruences. This is done by synthesizing interactions with the environment in order to obtain a compositional semantics.
We enrich the notion of reactive systems by conditions on two levels: first, as in earlier work, we consider rules enriched with application conditions and second, we investigate the notion of conditional bisimilarity. Conditional bisimilarity allows us to say that two system states are bisimilar provided that the environment satisfies a given condition.
We present several equivalent definitions of conditional bisimilarity, including one that is useful for concrete proofs and that employs an up-to-context technique, and we compare with related behavioural equivalences. We instantiate reactive systems in order to obtain DPO graph rewriting and consider a case study in this setting.

Edgar Arndt (TU Berlin)
Race Conditions in Population Protocols

We analyse the effect of race conditions on population protocols. A race condition in this context is a computational error that may occur when agent interactions are not atomic and may be interleaved with interactions involving other agents instead. We develop a formal model to extend population protocols that we conjecture to model race conditions in the underlying population protocol. Using extended protocols, we begin to examine well-known protocols in the presence of race conditions.

Joshua Moerman (RWTH Aachen University)
Query Learning for Register Automata

In this talk, I present results on learning register automata (RAs). These automata accept languages over an infinite alphabet, and they are strictly more powerful than ordinary automata. Learnability was motivated by verification of models for TCP, for which we need sequence numbers. Consequently, models for TCP have been learned, and some bugs have been uncovered, even in the closed-source Windows implementation.
There are different techniques for learning RAs, such as abstraction, logic, and symmetries... Each technique will have different advantages. I will discuss these methods and show an equivalent model, the so-called nominal automata. I will also present some new results on learnability of nondeterministic register automata.

Seyed Hossein Haeri (UCLouvain)
Piecewise Relative Observational Purity: A New Paradigm for Distributed Systems Programming

We promote a new paradigm for distributed systems programming: PROP. The execution of code written for distributed systems is "mostly functional" (i.e., the effectful proportion is indeed small and it's pure otherwise). Distributed systems programming, on the extreme contrary, is based on the assumption that it's all effectful. Our aim is to raise awareness of the programmer so they code in a mostly functional way too. To that end, we propose pure blocks as a discipline: a syntactic construct for the programmer to nominate the pieces of their code that are (im)pure as well as the nodes w.r.t. which they are (im)pure. In the talk, I'll present few benefits of such a paradigm and give some sample codes developed that way. I will also scratch the surface of our related formal semantics.

Richard Eggert (Universität Duisburg-Essen)
Fixpoint Theory - Upside-Down

Given a monotone mapping in a complete lattice of the form $\mathbb{L}^X$, where X is a finite set, we have developed a method which solves the following two problems in this (specific) setting: (1) Is a fixpoint the greatest fixpoint? (2) Is a prefixpoint above the greatest fixpoint?
We found why a fixpoint is not the greatest fixpoints; the culprits are so called "vicious cicles". In these "cycles", values convince each other to be too low, but increasing all values in this "cycle" at once would yield a postfixpoint.
Therefore we introduce a Galois connection which connects our lattice to the (finite) powerset-lattice $\mathcal{P}(X)$ to determine these "vicious cycles". This defines a function, whose greatest fixpoint gives the set of states in such a "vicious cycle".
This function also works for question (2). There are many applications for this method, i.e. markov-chains, bisimilarity, behavioural metrics, etc.

Sören van der Wall (TU Braunschweig)
Synthesis of concurrent, recursive Programs: Complexity of Multi-Pushdown Games

We study the complexity of context-bounded and phase-bounded multi-pushdown games (MPDG) with parity and reachability winning condition. Our first result shows that k-context parity MPDG are b-EXPTIME-complete, where $b = \max\{k-2, 1\}$. This means that up to three contexts do not increase the complexity over an analysis for pushdown games. Our second result shows that k-phase parity MPDG are k-EXPTIME-complete. Our upper bounds are obtained by revising Seth's reduction from MPDG to finite games. It was designed for phase-bounded computations. We give an optimization for the context-bounded case that saves two exponents. We also identify and fix a bug in Seth's work. Our lower bounds are based on a reduction from space-bounded alternating Turing machines. The novelty are so called first-order relations, relations between words that are defined in a fragment of first-order logic. We show how to decide first-order relations between words of $(k-1)$ fold exponential length by means of suitably efficient MPDG. Interestingly, we only need two stacks and reachability as the winning condition.

Tobias Heindel (TU Berlin)
What if Nash equilibria are too expensive?

Nash equilibria are hard to compute and more often than not far from social welfare - the price of anarchy is often high. So, why don't we have a quick look at correlated equilibria of multiplayer games (à la Shapley)?