1 General description (<= 1000 characters) <—— 985

Moderne Internetdienste, wie die von Amazon, Google, Facebook, oder Netflix, laufen in der Cloud. Benutzeranfragen werden von zehntausenden Computern bearbeitet, die sich in weltweit verteilten Datenzentren befinden. Bei so vielen involvierten Computern werden Fehler von der Ausnahme zur Regel. Daher wird es immer wichtiger Fehlertoleranzmechanismen auf rigorose Art zu entwerfen und zu verifizieren, dass diese Mechanismen tatsächlich ihre Aufgaben erfüllen.

TLA+ ist ein Formalismus, den Turing-Preisträger Leslie Lamport erfunden hat, um solche Fehlertoleranzmechanismen zu entwerfen. Das APALACHE Projekt widmet sich automatischen Verifikationsmethoden für TLA+. Wir entwickeln ein Model-Checking-Programm namens Apalache, das modernste automatische Verifikationsmethoden auf TLA+ anwendet. In späteren Projektphasen werden wir Apalache mit neu-entwickelten parameterisierten Model-Checking-Methoden erweitern um automatische Verifikation großer Systeme zu ermöglichen.

2 Scientific abstract (<= 1500 characters) <—— 1491

Critical distributed systems are designed to tolerate faults of individual components: they must work even if some of their components fail. Leslie Lamport, the pioneer of fault-tolerant distributed computing and 2014 Turing award winner, envisioned rigorous engineering of distributed systems with formal specification languages and proofs. His language TLA+ [Lam02] is used by engineers at Intel, Microsoft [BL03], and Amazon [NR+15].

When TLA+ was invented, automatic verification was technically limited to toy examples or manual abstractions. Thus, Lamport devised TLA+ for manual proofs. In practice, engineers prefer automated tools to manual proofs [BL03,NR+15]. TLC, the only model checker for TLA+, was developed in 1999, i.e., before widespread use of automatic abstraction and SMT solvers. This limits its practical use: one can only check systems with very few components [NR+15], and must hope that bugs do not appear in the deployed systems with significantly more components.

Since 1999, software model checking made revolutionary progress. Automatic abstraction methods, e.g., [CG+03], helped to prove software (e.g., Windows device drivers) to be free of critical bugs. Our goal is to achieve a similar success in the area of fault-tolerant distributed systems.

We will address two technical and methodological challenges:

- Domain-specific abstraction for fault-tolerant TLA+ designs.
- Parameterized model checking techniques for an unbounded number of components.

State of the Art and Scientific Challenge [1]

As many modern applications are running in the cloud, the user requests are processed by tens of thousands of commodity computers in data centers operated by, e.g., Amazon, Google, Facebook, and Netflix. When such a large number of computers is involved, faults become a norm rather than an exception[2]. As demonstrated by the recent outage of the Amazon Elastic Compute Cloud (EC2) in the eastern region of the US, which brought down about 70 sites[3], an unlucky sequence of faults and software bugs can make a whole data center unavailable. Hence, to make systems more reliable, we have to deal with two kinds of undesired phenomena: *faults and bugs*.

By a *fault* we understand a situation that is not controllable by a system designer, e.g., a computer crash or reboot, disk corruption, network disconnect, or power outage. A *bug* is a manifestation of an incorrect protocol design or software implementation, e.g., too small timeout, race condition, deadlock, buffer overflow, or memory leak. Historically, faults are dealt with by fault-tolerant distributed algorithms, which are mainly designed for safety-critical systems [KG94], e.g., plant or vehicle control systems. As indicated by the mentioned examples, more and more mainstream IT applications adopt fault-tolerance for economic reasons.

The existing methods to show the correctness — i.e., freedom of bugs — of fault-tolerant systems require deep mathematical background and are labor-intensive, e.g., paper-and-pencil proofs or computer-assisted proofs. In mainstream applications, the implementations and designs change at much shorter intervals than the ones in safety-critical systems. Hence, the labor-intensive verification methods are of limited use. For these new applications of fault tolerance, we need domain-specific verification methods that have a large degree of automation.

Why TLA$^+$? The standard approach to implement fault-tolerant systems is to take a fault-tolerant distributed algorithm from the literature and try to implement its algorithmic idea faithfully. Transferring a distributed algorithm, which is typically given in pseudo code, to a running system can be ambiguous and ad-hoc. The Paxos [Lam98] approach to replicated state machines has been transferred into implementations such as PBFT [CL+99], Chubby [CGR07], Zyzzyva [KA+07], ZAB [JRS11], EPaxos [MAK13], RAFT [OO14]. As admitted by the specialists who did such implementations, "*Converting the algorithm into a practical, production-ready system involved implementing many features and optimizations — some published in the literature and some not*." [CGR07]. Even if a fault-tolerant distributed algorithm has been shown to be correct in theory, due to this ad-hoc approach, the correctness of the theoretical algorithm does not imply the correctness of the *implementation*. For instance, the developers of Apache ZooKeeper — a distributed coordination service for cloud computing produced by Yahoo — introduced conceptual bugs to the ZAB protocol while "optimizing" it [Med12].

Anticipating these problems, Lamport introduced TLA in the 80ies (and later TLA$^+$). Its precise semantics replaces the pseudo code that is predominant in distributed algorithm litera-

[1] **As our multidisciplinary project spans several computer science communities, we cannot include all relevant references in two pages (as required by WWTF). We omit references to some protocols, tools, etc.**

[2] `http://techblog.netflix.com/2010/12/5-lessons-weve-learned-using-aws.html`

[3] `http://aws.amazon.com/message/65648/`

ture. Refinement — i.e., adding more and more details to the TLA$^+$ specifications — has been introduced as a rigorous method to transfer a high-level specification into a running system. In principle, this allows to verify systems in two steps: (i) verify the high level design, and (ii) verify that the implementation refines the high-level design. Both steps are hard, however. In general even (i) is undecidable as TLA$^+$ includes set theory. The only existing tool for automatic verification, the TLC model checker, is used to check *simplified and finite-state* designs [NR+15]: one has to fix the number of components in a system (to a typically very small number), one has to fix the domain of variables, etc. Hence, TLC does not represent the state-of-the-art in model checking. *This project is devoted to cutting edge model checking techniques for the verification of fault-tolerant system designs specified in* TLA$^+$*. Thus,* APALACHE *contributes to goal (i), providing confidence in correctness of complex real-world software at a natural level of abstraction [NR+15].* Goal (ii) is not in the scope of this proposal.

Predicate Abstraction and Parameterization in Model Checking. One of the cornerstone methods to deal with complex systems is predicate abstraction [GS97]. It efficiently leverages propositional satisfiability (SAT) and satisfiability modulo theories (SMT) solvers in model checking. Recent technological leaps in SAT and SMT in connection with counterexample-guided abstraction refinement (CEGAR [CG+03]) resulted in successful software model checkers, e.g., SLAM [BLR11], BLAST [HJ+02], IC3 [BBW14]. Our own recent work on parametric interval abstraction (see below) can be seen as a form of predicate abstraction tailored for fault-tolerant distributed algorithms.

Fault-tolerant designs are parameterized in the number of components and thus need parameterized verification. Parameterized model checking differs from textbook model checking in that it addresses an *infinite* family of systems, each system having different number of components. Four major research lines are most relevant to our project: cut-off methods [EN95; EK03; CT+04], techniques for well-structured transition systems [Abd10; FKP15], compositional techniques [McM01; CMP04; OTT09], and techniques based on abstraction [GS92; PXZ02; CTV08]. As parameterized model checking is decidable for simple cases such as token rings, concurrent protocols (e.g., mutual exclusion), and cache coherence protocols, we need to combine them with abstraction. Our own work and [AG+12] are rare examples of parameterized techniques for fault-tolerant distributed algorithms.

Specification frameworks. TLA$^+$ by Leslie Lamport [Lam02] and I/O Automata (IOA) by Nancy Lynch [Lyn96] are specification and refinement frameworks that are intended to span the design and implementation process. Both were devised when automated verification was out of reach, and were thus designed with manual proofs in mind. Still, industry is starting to adopt these frameworks: TLA$^+$ was used at Microsoft [BL03] and Amazon (in "*10 large complex real-world systems*" [NR+15]), and IOA was used at Oracle [LLM12]. Researchers are using TLA$^+$ and IOA to document fault-tolerant protocols: PBFT [CL+99] is specified in IOA, and DiskPaxos [GL03], EPaxos [MAK13], and RAFT [OO14] are specified in TLA$^+$.

Both IOA and TLA$^+$ offer basic tool support for verification: The Tempo toolset [AL+08] for IOA interfaces with the proof assistant PVS and the model checker UPPAAL, and the TLA$^+$ Toolbox [Lam02] offers the proof assistant TLAPS and the model checker TLC [YML99]. While development of the Tempo toolset seems to be frozen, there is a constant improvement of

TLA$^+$. As we have learned in detail during a visit at our collaboration partner Stephan Merz (he and his group are the primary TLA$^+$ developers who have strong collaboration with a small corresponding team at Microsoft), TLAPS was recently integrated with SMT solvers [MV12a; MV12b]. Hence, the user can automatically prove simple proof steps with SMT. Invariant checking in TLC was also extended with distributed state exploration, so TLC can be used in a cluster. Yet, the core algorithm of TLC uses explicit state enumeration, and thus does not offer parameterized model checking or other state-of-the-art model checking techniques.

Our own recent related work. In the last years, we achieved a breakthrough in parameterized model checking of fault-tolerant distributed algorithms (FTDAs). We were the first to automatically verify non-trivial FTDAs from the literature. Our work is based on an interdisciplinary effort to ensure that we adequately model FTDAs without simplifications. To do so we introduced two modeling frameworks, namely, control flow automata [JK+13b] and threshold automata [KVW14]. Based on them we generalized and combined data and counter abstraction into our *parametric interval abstraction* [JK+13a], which allowed us to model check the first FTDAs. To verify more involved FTDAs, we introduced an offline partial order reduction in [KVW14]. In our most recent work [KVW15], we introduced an even more aggressive partial order reduction and showed that it allows, in combination with SMT solvers, to check sophisticated FTDAs such as condition-based consensus and one-step consensus. These techniques are implemented in our freely available tool ByMC [`http://forsyte.at/software/bymc`] that is continuously extended. We also presented our results as brief announcement at ACM PODC and received very positive feedback from the distributed algorithm community. Moreover, the members of the core team read lectures about these results at the SFM14:ESM summer school [GK+14] in Bertinoro/Italy, the summer school VTSA'14 in Luxembourg, and the conference on Tools & Methods for Program Analysis'14 in Kostroma/Russia. Besides, in [DH+14] we introduced a logic-based approach for the verification of consensus algorithms.

The scientific goal of APALACHE *is the development of state-of-the-art model checking tools for* TLA$^+$*. Due to rising industrial interest in* TLA$^+$ *and the availability of advanced model checking techniques, we believe in the timeliness of the project.*

Other approaches to system correctness. There are many attempts to debug distributed systems at the implementation level. The main drawback of many existing bug-finding tools is that they are neither *sound* (every "reported bug" is reproducible) nor *complete* (every actual bug is reported). Testing techniques exercise the system by randomly or systematically injecting faults in the components, e.g., [GD+11]. Recent debugging tools for distributed systems use model checking state exploration techniques. Such *distributed system model checkers* periodically take a snapshot of the system and explore the states reachable from the snapshot by normal system code or faults, e.g., [LY+09; GY11; LH+14]. These tools do not guarantee completeness. To cope with combinatorial explosion of faults and states, some recent tools allow the user to guide the search by writing reduction rules, e.g., SAMC [LH+14], which is however neither complete, nor sound. As Amazon engineers note: "*...testing the code is inadequate as a method for finding subtle errors in design, as the number of reachable states of the code is astronomical*" [NR+15]. Thus, implementation-level debugging techniques miss bugs that occur only in the large scale systems.

Project Summary

Distributed systems are inherently difficult to analyze. Concurrency and faults make reasoning much harder than for conventional sequential programs. To address this, Lamport introduced TLA$^+$ as a precise logic-based specification framework [Lam02]. In TLA$^+$, system behavior is expressed in first order logic and set theory. TLA$^+$ is a very expressive language, which is beneficial for system designers, but it constitutes a challenge for automated verification. Indeed, the current automated tool support is quite limited. It consists of an interactive proof assistant TLAPS for safety specifications and the explicit state model checker TLC. The goal of this project is to (i) bring state-of-the-art model checking techniques to the realm of TLA$^+$, and (ii) to develop new TLA$^+$ verification techniques for industrial needs.

Research questions. Our goal is to verify fault-tolerant distributed systems expressed in TLA$^+$ using advanced parameterized and abstraction-based techniques. To this end, we have to address the following questions:

Q1. *Rich language*. Specifications in TLA$^+$ are considerably more expressive than standard software: TLA$^+$ is untyped, it allows quantification over sets, comparison of cardinalities, and comparison and updates of the states of concurrent components.

Q2. *Parameterized systems*. TLA$^+$ specifications should be checked for an unknown number of components (in order to verify systems of practical scales, such as in clouds).

Q3. *Verification beyond toy examples*. Our tool should be able to verify with state-of-the-art protocols such as DiskPaxos [GL03], ZAB [JRS11], and RAFT [OO14], encoded in TLA$^+$.

Hypotheses and Approaches. Not surprisingly, we face theoretical limits. Both, satisfiability of a TLA$^+$ formula and parameterized model checking are known to be undecidable [AK86]. Consequently, there is no general algorithm to address (Q1) and (Q2). Deep understanding of our application domain is the key to the goal:

H1. *Code patterns.* TLA$^+$ designs use concepts from the distributed algorithms literature. Usually, TLA$^+$ code maintains messages sent by the processes in a monotonically growing set, compares process identifiers to choose a leader, compares ballot numbers, or checks majorities to ensure a quorum, etc.

Hence, although TLA$^+$ is designed to be extremely expressive (in order not to limit the designer), its users follow many coding idioms. Thus, our domain has good prerequisites for automated verification.

H2. *Communication patterns*. Distributed algorithms have only few mechanisms to achieve fault tolerance, e.g., process replication, where replicas send messages to *all* other replicas to keep the system in a good state (e.g., in reliable broadcast). In another mechanism, replicas use a coordinator and send messages only to the coordinator (e.g., in Paxos). In [JK+13b; JK+13a; KVW14; KVW15], we already used such domain knowledge for efficient verification.

H3. *Not push-button*. Due to the inherent hardness of the problem, we cannot expect 100% automation. The experience at Intel [TT08] shows that highly-skilled engineers are willing to help the tool by giving hints, e.g., by manually annotating the code, providing abstractions or writing invariant candidates.

To address questions (Q1)–(Q3), we will develop new techniques based on model checking as a central paradigm for verification of TLA$^+$ designs. We will develop new decision

procedures and improve predicate abstraction [GS97] and counter-example guided abstraction refinement CEGAR [CG+03] to deal with the patterns of fault-tolerant TLA$^+$ designs (see H1–H3).

Objectives and Research plan. Our goal is to extend the functionality of the TLC model checker along two axes to arrive at the following tools:

O1. APALACHE-FIN. Similar to TLC, this tool verifies finite state systems, that is, given a TLA$^+$ design, one has to fix, e.g., parameters or variable domains before verification. In contrast to TLC, our tool APALACHE-FIN adapts state-of-the-art model checking techniques to TLA$^+$. Similar to the effect of predicate abstraction in software model checking, we expect our tool to achieve dramatic performance improvements over TLC.

O2. APALACHE-PAR. This tool will be able to deal with parameterized designs, that is, without fixing, e.g., the number of processes. To do so, we will develop domain specific parameterized model checking techniques for an unbounded number of components.

We divide the required work towards these objectives in the following work packages.

Work Package A. *Repository of* TLA$^+$ *patterns*. TLA$^+$ neither has explicit control flow, nor standard language primitives to express fault-tolerant designs. To perform abstraction and verification, we have to use the domain knowledge that comes in the form of code idioms and design patterns. We will analyze the published TLA$^+$ specifications of the available fault-tolerant protocols and create an open repository of TLA$^+$ patterns.

Work Package B. *Predicate abstraction of* TLA$^+$ *patterns.* We will develop a predicate abstraction framework for TLA$^+$ specifications of finite-state systems, i.e., with all parameters fixed. The key challenge is to construct the predicate abstraction of a system step encoded in TLA$^+$. Software model checkers use logic decision procedures to do this for program statements. Similarly, to verify FTDAs in TLA$^+$, we need decision procedures for message sets, counting arguments, faults, resilience conditions, etc. [JK+13a]. Such specialized decision procedures are an active research area [MV12b; DH+14].

Work Package C. *Parameterized verification of* TLA$^+$ *designs.* To verify fault-tolerant designs parameterized in the number of processes and faults, we will develop techniques based on our recent results on parametric abstractions and partial order reduction [JK+13a; KVW14; KVW15] as well as the framework of invisible invariants [PRZ01]. Further, we will develop a framework for combining the verification results from a portfolio of parameterized model checking techniques such as data and counter abstraction, environment abstraction, and well-structured transition systems. The framework also includes new abstraction refinement techniques.

Work Package D. *Evaluation of the approach on fault-tolerant* TLA$^+$ *designs*. The relevance of abstraction-based methods can only be confirmed in empirical studies. In a continuing task, we will evaluate the tools APALACHE-FIN and APALACHE-PAR on challenging case studies coming from three sources: the published specifications in TLA$^+$, e.g., DiskPaxos [GL03], RAFT [OO14], EPaxos [MAK13], GFS, Niobe, and Chain [GBM08]; the benchmarks of our own more specialized tools [JK+13a; DH+14; KVW15]; and designs from systems oriented research, e.g., PBFT [CL+99], Zyzzyva [KA+07], ZAB [JRS11].

3 Measuring Project Success (<= 750 characters) <—— 729

The most direct outcome will be the freely available APALACHE tool set. The work packages are designed in a way that allows us to build on a stable model checker developed in WP B that we will upgrade with the new techniques from WP C. In particular, our goal is to verify TLA+ designs that have never been automatically verified before.

We will present our results at the top venues on computer-aided verification (CAV, FMCAD, TACAS, CONCUR, VMCAI), distributed computing (PODC, DISC), systems research (NSDI, OSDI), and software engineering (ICSE, ASE). As we aim at fundamental results, we will submit articles to journals such as Distributed Computing, Formal Methods in Systems Design, and Information and Computation.

Outline of Approach

As motivated in the project summary, our goal is to transfer state-of-the-art model checking methodology to TLA$^+$. Due to the expressiveness of TLA$^+$ and parameterization of the realistic TLA$^+$ specifications (cf. Q1 and Q2), most of our verification problems are undecidable. Model checking research, including our own work, shows that abstraction is a pragmatic approach to tackle practical instances of undecidable problems. To circumvent undecidability, abstraction exploits domain knowledge and features of the problem instances, e.g., counter abstraction [PXZ02; JK+13a] uses symmetry.

Our own research in the last years was a first step towards this agenda. As explained in the state of the art section, we introduced new techniques for parameterized model checking of fault-tolerant distributed algorithms [JK+13a; KVW14; KVW15]. These techniques apply to algorithms that are parameterized in the number n of identical (symmetric) processes, among which at most t processes are faulty, and whose process code contains threshold guards like "`if received (ping) from at least n-t distinct processes`". Testing whether the number of messages is above the threshold $n - t$ is an algorithmic pattern, which is omnipresent in the literature. For example, in [KVW15] we verified subtle fault-tolerant algorithms that exclusively use this pattern: asynchronous reliable broadcast and Byzantine agreement, non-blocking atomic commit, condition-based consensus, one-step consensus.

Industrial TLA$^+$ designs are instantiations of many algorithmic patterns from distributed algorithms (e.g., threshold guards, leader election, heartbeats). In APALACHE, we exploit such patterns to automatically build abstractions of TLA$^+$ code and thus make the verification problem amenable for model checking. To this end, we will collect TLA$^+$ patterns from the published TLA$^+$ code of fault-tolerant designs in WP A.

In WP B, we focus on predicate abstraction [GS97] specific to code patterns in TLA$^+$. By abstracting TLA$^+$ designs with a constant number of processes, we will be able to verify the same code as the TLC model checker, but for larger state spaces.

To check parameterized TLA$^+$ designs — unbounded in the number of processes and faults — we lift the techniques of WP B to parameterized model checking in WP C. There, we extend our techniques based on counter abstraction [JK+13a] and partial order reduction [KVW14; KVW15] to deal with a range of TLA$^+$ patterns. As there is no single parameterized verification technique that fits all features of TLA$^+$ designs, we will develop a framework to

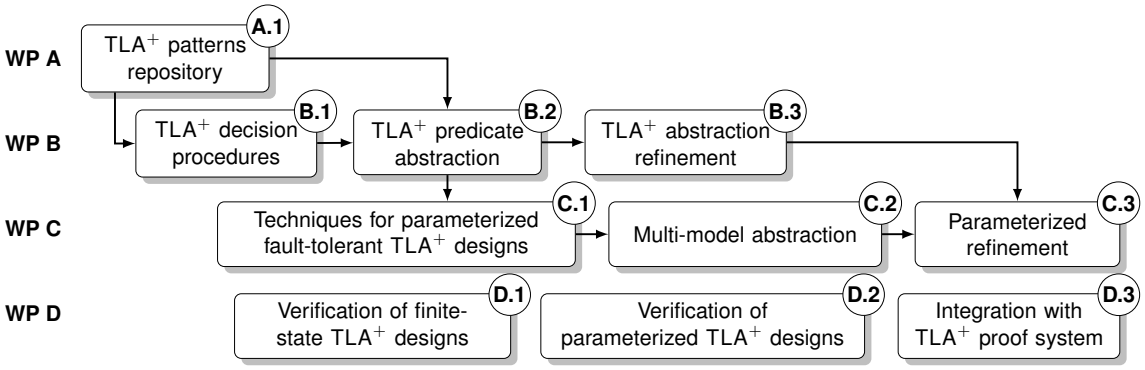

Figure 1: The work packages and the tasks. (WP D is run in parallel with WP B and C.)

decompose the verification problem into several subproblems to be addressed with dedicated parameterized model checking techniques.

Finally, to make the results of APALACHE applicable to realistic TLA$^+$ code, we will continuously evaluate our techniques and tools on the published TLA$^+$ designs in WP D. We will start evaluation, as soon as we have the first version of the tools, to get the feedback early on.

Collaborations. We have established collaborations that cover all of APALACHE work packages. For WP A, we have received a support letter from Byron Cook (expert in predicate abstraction, Senior Principal at Amazon), who is inviting Igor Konnov (PI) for a three-month research visit at Amazon, before the start of APALACHE. Stephan Merz (expert in TLA$^+$, LORIA Nancy) will collaborate with us on WP A, B, and D. Know-how on decision procedures for WP B comes from our continuing work [DH+14] with Cezara Dragoi (INRIA Paris). We work with Azadeh Farzan (U. Toronto) on parameterized verification in WP C. We continue collaboration with Laura Kovacs (TU Chalmers) and Georg Weissenbacher (Forsyte, TU Wien) on decision procedures and interpolation for WP B.

Work Package A: Repository of TLA$^+$ patterns

Key Contributors: J. Widder, I. Konnov, PhD students NN1, NN2

Collaborations. Byron Cook (Amazon), Stephan Merz (LORIA Nancy)

Summary. As a general specification language, TLA$^+$ does not offer special syntax for any particular application domain. In his book [Lam02], Lamport specifies sequential, concurrent, real-time, and distributed systems with TLA$^+$. Moreover, in contrast to programming languages, TLA$^+$ is flat: it does not enforce the user to define process boundaries, structure the control flow or restrict variable scope. In essence, a system step is specified as a logical formula over state variables, sets, and functions. Although this feature of TLA$^+$ is no problem for manually constructed proofs, it hinders automatic analysis tools. Indeed, software model checkers, e.g., SLAM [BLR11], BLAST [HJ+02], CTIGAR [BBW14], heavily use program control-flow and scoping to efficiently compute an abstraction.

In addition to TLA$^+$, Lamport has introduced Pluscal, a language for sequential and concurrent algorithms. This language resembles programming languages in that it has explicit control flow, processes, global and local variables, and concurrency operators. As noted by the engineers at Amazon [NR+15], whether a design is done in Pluscal or TLA$^+$ depends to a certain degree on the personal taste of the engineer. Hence, we should support both languages. Fortunately, the semantics of Pluscal is defined by translation to TLA$^+$. Thus, we can extract typical Pluscal translation patterns from the documentation.

Research Contribution. Following hypotheses (H1)–(H2), we collect TLA$^+$ patterns from several sources into a publicly available repository. Similar to the collection of driver API rules in SLAM [BLR11], our repository plays a critical role in APALACHE. Moreover, the availability of the repository will boost research on verification tools for TLA$^+$.

Lamport follows certain idioms in his book, e.g., the next-state action is defined as a disjunction of formulas specifying alternative steps, while a single step is specified as a conjunction of constraints. As the book [Lam02] is an authoritative source on TLA$^+$, we expect the users to follow Lamport's coding style, and do not write specifications in an arbitrary form.

Task A.1: Repository of TLA$^+$ Patterns. As the first steps, we collect the patterns from the TLA$^+$ book [Lam02] and Pluscal to TLA$^+$ translation rules. Further, we analyze the published TLA$^+$ code of the following fault-tolerant protocols: various versions of Paxos by Lamport (Paxos made simple, Disk Paxos [GL03], Byzantine Paxos, Stopping Paxos, Fast Paxos, Vertical Paxos), RAFT [OO14], EPaxos [MAK13], GFS, Niobe, and Chain [GBM08]. As the starting point of the analysis, we consider how the following typical features of distributed algorithms are encoded as TLA$^+$ expressions: message transmission, message receive, faults, timeouts, heartbeats, majorities and quorums, failure detectors, coordinators, and view changes. The repository will contain the identified patterns and templates of TLA$^+$ expressions that can be used to generate predicates in the following tasks.
Deliverables. An open repository of TLA$^+$ patterns as input to the other work packages.

Work Package B: Predicate abstraction of TLA$^+$ patterns

Key Contributors: I. Konnov, PhD student NN1, A. Laarman, H. Veith
Collaborations. Georg Weissenbacher, Stephan Merz (LORIA), Cezara Dragoi (INRIA).
Summary. Here, we address the research question (Q1). Software model checkers such as SLAM [BLR11], BLAST [HJ+02], CTIGAR [BBW14] focus on efficient predicate abstraction of source code that performs long sequential computations. To compute the effect of program statements on predicates, software model checkers use logic decision procedures for quantifier-free formulas over, e.g., linear integer arithmetics or uninterpreted functions. TLA$^+$ specifications are very different from source code: they represent highly concurrent, immensely non-deterministic, and relatively short computations (e.g., in [NR+15] the longest counterexample had 35 steps, which is orders of magnitude smaller than in software model checking); they employ quantified formulas over set operations, functional operations, and arithmetics; they are inevitably parameterized; finally, they contain few boilerplate code.

Though WP B focuses on non-parameterized systems, the parameterized verification methods of WP C can use the machinery of WP B to, for instance, construct predicate abstraction of a single process. Hence, our goal is not only to build a finite-state model checker, but also to interface general TLA$^+$ code with highly-specialized techniques of parameterized verification.
Research Contribution. Based on the TLA$^+$ patterns from WP A, we develop an approach to generate predicates that represent an abstract global state of a given TLA$^+$ specification. The key challenge here is to construct a transition relation between the abstract states, which constitutes a sound abstraction of a system step encoded in TLA$^+$. For this, we develop new decision procedures for, e.g., message sets, message counting arguments, faults, and resilience conditions [JK+13b]. Such decision procedures are an active research area, where we have recently contributed [DH+14].

We investigate two approaches to constructing an abstract transition relation of TLA$^+$ specification: In Task B.1 we construct a predicate abstraction for a parameterized TLA$^+$ specification using special decision procedures; in Task B.2 we fix the parameters of a TLA$^+$ specification, and construct its predicate abstraction using off-the-shelf SAT and SMT solvers.

When a model checker reports a property violation, the counterexample is either *feasible* (a real bug in the system) or *spurious* (an abstraction artifact). Thus, to close the abstraction-

refinement loop, we have to deal with spurious counterexamples. In Task B.3, we develop specialized (semi-)decision procedures for refinement that focus on the patterns from WP A.

Task B.1: (Semi-)decision Procedures for TLA$^+$. Given the patterns from WP A, we develop specific (semi-)decision procedures for the key TLA$^+$ constructs that are needed to build an abstract transition relation of a TLA$^+$ specification. In the context of the Heard-Of model [CS09], our recent work [DH+14] gives such a semi-decision procedure for Consensus Logic, which focuses on universally quantified formulas with syntactically restricted set comprehensions and set cardinalities. Although TLA$^+$ is very expressive, we believe that this task is feasible for practically important fragments of TLA$^+$. We start with the SMT theory of finite sets, lists, and maps [KWR09] and encoding of TLA$^+$ in SMT [MV12b] and focus on the patterns from WP A.

Deliverables. Publication of scientific results and prototypical implementation that contributes to the model checkers APALACHE-FIN and APALACHE-PAR.

Task B.2: Predicate Abstraction for TLA$^+$. This is a low-risk counterpart of Task B.1. Similarly to TLC, we fix parameters (e.g., the number of processes, the set of initial values) of a TLA$^+$ specification. However, instead of enumerating reachable states, we first construct a predicate abstraction, which abstracts away unnecessary details, and then apply either our model checker ByMC, or an off-the-shelf model checker (e.g. nuXmv, IC3) to the finite-state transition system. In this task, having the TLA$^+$ patterns on the input, we investigate how to efficiently compute an abstract transition relation.

Propositional satisfiability (SAT) and satisfiability-modulo-theory (SMT) solvers have made tremendous progress in the last decade. As TLA$^+$ is untyped, we first have to apply the type inference algorithms for TLA$^+$ [MV12a]. Then, given an abstract state as a set of predicates, we compute the values of predicates in the successor states by querying the solver:

1. Encoding the TLA$^+$ transition relation as a SAT formula. Once all parameters are fixed, the domains of all variables become finite, and we can use bit-blasting techniques [KBS08] to express a TLA$^+$ specification as a propositional formula.
2. Encoding the TLA$^+$ transition relation as an SMT formula. The results [MV12a; MV12b] on checking TLA$^+$ proof obligations with SMT is our starting point. However, we expect that we can find more efficient solutions to our problem, because we have to reason about local transitions of one process, instead of inductive invariants.

Deliverables. Publication of scientific results and prototypical implementation that contributes to the finite-state model checker APALACHE-FIN.

Task B.3: Abstraction refinement for TLA$^+$. Similar to Task B.2, we first focus on the case when the parameters are fixed and variable domains are bounded. In this case, refinement is in principle decidable [CG+03]. By applying the SAT and SMT encodings of TLA$^+$ specifications from Task B.2, we can automatically check, whether a counterexample is spurious (the parameterized case is addressed in Task B.3). Nowadays, interpolation is the dominating technique for abstraction refinement [McM03; McM14; BBW14].

In a nutshell, interpolation works as follows. Let φ and ψ be two formulas that model a set of system states and its abstraction respectively. When φ and ψ are inconsistent (i.e.,

ψ is a spurious abstraction), then there is a formula ρ (interpolant) that is finer than ψ, and ρ is still less detailed than φ. Since TLA$^+$ is extremely expressive, we cannot expect the interpolation property for the whole logic. Hence, we identify useful fragments of TLA$^+$ that have the interpolation property or variations of it, and thus, allow us to perform refinement.

Deliverables. Publication of scientific results and prototypical implementation that contributes to the finite-state model checker APALACHE-FIN.

Relation to other work packages. This work package constitutes a *bootstrapping phase* for work packages C and D. First, it allows us to build frameworks and prototypes in the early stage of APALACHE and thus to quickly evaluate the feasibility of our solutions (Task D.1). Second, it provides us with necessary skills and tools to proceed with the work package C.

Work Package C: Parameterized verification of TLA$^+$ designs

Key Contributors: J. Widder, PhD student NN2, M. Lazic, I. Konnov, T. Kotek, H. Veith

Collaborations. Laura Kovacs (Chalmers), Azadeh Farzan (U. Toronto)

Summary. In this work package we address the research question (Q2). Research on parameterized model checking has generated a wealth of techniques for specialized and often simplified computation models. On one hand, there are decidability results for token passing systems [EN95; CT+04], systems with rendezvous communication [GS92], and systems with disjunctive and conjunctive guards [EK03]. On the other hand, there are semi-decision procedures that put less restrictions on the computational model, but do not guarantee completeness, e.g., counter abstraction [PXZ02] and environment abstraction [CTV08], techniques for well-structured transition systems [Abd10], and flat counter automata [BF+08].

Realistic fault-tolerant designs demand for new parameterized verification techniques. As no single technique fits all systems, there is a need for a framework that allows the user to apply parameterized verification techniques for the *orthogonal* abstractions. For instance, in [JK+13a], we manually combined (and generalized) data abstraction and counter abstraction to deal with message counters, process counters, and resilience conditions.

Research Contribution. In Task C.1, we systematically integrate our own techniques for parameterized model checking and develop new techniques to deal with the features of TLA$^+$ fault-tolerant designs. To apply multiple parameterized model checking techniques to the same TLA$^+$ specification, in Task C.2, we develop a framework that splits a TLA$^+$ specification into multiple abstractions and checks each abstraction with a dedicated procedure. This allows us to decompose the parameterized verification problem into several orthogonal subproblems. Finally, in Task C.3, we investigate the new type of spurious counterexamples, which are caused by parameterization in the number of processes.

Task C.1: Techniques for parameterized fault-tolerant TLA$^+$ designs. Our recent parameterized model checking techniques [JK+13a; KVW14; KVW15] have been experimentally proven successful for sophisticated fault-tolerant distributed algorithms. As a first step in this task, we systematically integrate them into the TLA$^+$ environment.

Further, in our latest work [KVW15], we have discovered that reasoning about partial orders can be incorporated into SMT-based bounded model checking, which gives us a complete and efficient algorithm for a special class of fault-tolerant distributed algorithms. This result

applies to threshold-based algorithms, which are based on one pattern found in fault-tolerant algorithms: counting how many messages were sent by other processes. We extend this technique to deal with the new TLA$^+$ patterns found in work package A.

In another research line, we investigate extensions of the technique of invisible invariants [PRZ01; MZ11] to TLA$^+$ specifications. The idea of invisible invariants is to first automatically construct an inductive invariant of the system with a fixed number of processes, and then to generalize it to an arbitrary number of processes. Based on the patterns (Task A.1), we investigate generalization rules for our domain, as the specifications of fault-tolerant distributed algorithms are quite different from specifications of concurrent systems (e.g., mutual exclusion algorithms that the technique of invisible invariants was designed for). Such rules will involve cardinalities and quantified formulas over set elements.

Finally, we integrate new domain-specific abstractions that will be developed by co-PI Josef Widder in his FWF standalone project P-27722 "PRAVDA: Parametrized Verification of Fault-tolerant Distributed Algorithms" that was awarded in February 2015.

Deliverables. Publication of scientific results and prototypical implementation. This task constitutes the core of the parameterized model checker APALACHE-PAR.

Task C.2: Multi-model Abstraction. So far, integration of parameterized model checking techniques has been done by stacking up abstractions, as in [ZP04] or in our own work [JK+13a]. This approach, however, does not apply, when there is no natural order between the abstractions, and the techniques need feedback from each other. For instance, the fault-tolerant algorithm from [CS09] requires us to reason both about rotating coordinators and message counting. While our work [JK+13a] deals with message counting, and the work on token rings deals with round-robin scheduling [EN95], there is no obvious way to compose these two techniques. Therefore, we need a framework that integrates multiple abstractions.

We organize the multi-abstraction framework as follows. Given two orthogonal verification techniques X and Y and a TLA$^+$ specification S, we construct sound abstractions S_X and S_Y of S using the predicates supported by techniques X and Y. The abstraction S_X is then verified with technique X: if X does not report a counterexample on S_X, then the specification is correct w.r.t. X; otherwise, X returns an automaton A that captures counterexamples found by X. In the latter case, we apply compositional reasoning between X and Y by analyzing counterexamples. We do not decompose a system into modules and do not use modular reasoning in the classical sense, but reason about different features of the same execution by applying different techniques.

Deliverables. Publication of scientific results and contributions to the tool APALACHE-PAR.

Task C.3: Refinement of Parameterized Counterexamples. In [JK+13a], we have found a new type of spurious counterexamples, where each step occurs in some concrete system, but no single concrete system demonstrates the complete counterexample. We investigate in which cases feasibility checking of such parameterized counterexamples is decidable. Further, we develop refinement semi-algorithms for such counterexamples.

Deliverables. Publication of scientific results.

Work Package D: Evaluation of the Approach on Fault-tolerant TLA$^+$ Designs

Key Contributors: J. Widder, I. Konnov, PhD Student NN1, T. Pani

Collaborations. Stephan Merz (LORIA), and the joint Microsoft/INRIA Lab (Paris).

Summary. In this work package we address the research question (Q3). In work packages B–C, we design and implement abstraction methods. Even theoretically sound abstractions do not necessarily work well in practice. An impractical abstraction is either too coarse, which results in many spurious counterexamples, or too fine, which results in prohibitively large state space. Consequently, the relevance of abstraction-based methods can only be confirmed with empirical studies, and experiments are crucial for our approach. This work package is a continuing task that focuses on evaluation of the tools developed in APALACHE.

Research Contribution. We select challenging case studies coming from three sources. First, we check the TLA$^+$ specifications that we used in WP A to collect patterns. Second, we translate into TLA$^+$ our case studies encoded in Promela, which were used in specialized tools [JK+13b; KVW14; KVW15]. Third, we pick several fault-tolerant protocols from systems oriented research (e.g., PBFT [CL+99], Zyzzyva [KA+07], ZAB [JRS11]) and specify them in TLA$^+$. We then use these case studies to evaluate our tools in Tasks D.1–D.2.

Task D.1: Verification of finite-state TLA$^+$ designs with APALACHE-FIN. We evaluate the prototype tool APALACHE-FIN developed in work package B and the TLC model checker (state of the art for TLA$^+$) on the selected benchmarks. We also compare performance of TLA$^+$ model checkers with off-the-shelf model checkers such as nuXmv, UPPAAL, IC3, FASTer. As these model checkers restrict input to their specific format, we use only those TLA$^+$ specifications that can be translated to the tool-specific format without oversimplification.

Deliverables. Scientific publications in the form of tool papers and case studies.

Task D.2: Verification of parameterized TLA$^+$ designs with APALACHE-PAR. We evaluate the prototype tool APALACHE-PAR developed in work package C on the selected benchmarks. We also compare performance of APALACHE-PAR with our tool ByMC, which is tailored to the special domain of threshold-based fault-tolerant algorithms.

Deliverables. Scientific publications in the form of tool papers and case studies.

Task D.3: Integration with the TLA$^+$ Proof System. In parameterized model checking, parameters define the number of processes, correct or faulty. Fault-tolerant distributed algorithms and thus TLA$^+$ specifications in addition exhibit another kind of parameters, for instance: the number of input values in consensus, or the set of potential ballots in Paxos. Usually model checking is limited to some fixed "reasonable" parameter values, e.g., the set of inputs is restricted to two values in consensus (binary consensus). This reduction is often accepted without a proof, though, in most cases, it can be shown by a mechanical proof of simulation.

Hence, we integrate our techniques with the TLA$^+$ proof system by proving certain lemmas with model checking. This approach uses the best of the two worlds: model checking reasons about non-determinism and concurrency, which humans find hard to grasp; the users generalize model checking results with simple inductive proofs, which need human ingenuity.

Deliverables. Scientific publications, tool integration between TLAPS and APALACHE.

Relation to other work packages. This WP closes the practical side of the abstraction-refinement loop and provides feedback for the methods developed in the other packages.

4 Innovative aspects (up to 2000 characters) <—— 1705

State-of-the-art model checking tools are not adapted to the domain of fault-tolerant distributed systems:

1. While existing software and hardware verification tools focus on systems with a relatively small degree of non-determinism, distributed systems have immense non-determinism (even in comparison to concurrent software).
2. Software model checkers focus on implementation issues, such as race conditions, buffer overflows, deadlocks, etc. In sharp contrast, distributed systems exhibit difficult bugs on the design level, which involve multiple communication components [NR+15].

Igor Konnov (PI, model checking), Josef Widder (co-PI, distributed algorithms), and Helmut Veith (head of the hosting group and collaborator) already made significant progress in the interdisciplinary effort of model checking fault-tolerant distributed algorithms. In APALACHE, we put together distributed computing and model checking to an innovative application.

We will extend our interdisciplinary effort by collaboration with Stephan Merz (LORIA), who has rich experience in TLA+ and TLAPS. Additional know-how on semi-decision procedures comes from continuing interdisciplinary work [DH+14] with Cezara Dragoi (INRIA).

Igor Konnov will visit Byron Cook (Amazon) to learn more on the state-of-the-art industrial approach towards TLA+ designs. Cook heads a new international formal verification team that addresses problems in cloud computing at Amazon. He has expressed strong interest in the project, as the new verification methods we will develop will have high practical value in the cloud industry. We will bring novel model checking techniques to the design process of industrial distributed systems.

5 Comment <—— 159

Describe which aspects of the proposal are especially novel/innovative. How does the project differ from the ongoing research activities of each team member?

6 Prospective benefits (up to 1250 characters) <—— 1221

Our society depends on the correct operation of computer systems that are inevitably distributed. The distributed nature of the computations and faults introduce a new level of complexity. As one can see from the recent report by Amazon [NR+15], big enterprises, e.g., cloud providers, call for verification tools to tame this new level of complexity and to deliver high-quality products on time.

As we address specifications in TLA+ (used at Intel, Microsoft [BL03], and Amazon) we expect our results ready for industrial adoption by the end of the project. While we focus on TLA+, our techniques are not bound to a specific programming language or platform. Thus, the results can be later transferred to other development frameworks. This will require close collaboration with industry.

Thanks to the strategic national grant RiSE, Austria has become a center for automated verification: http://arise.or.at. Groups led by recognized researchers, e.g., in Graz (R Bloem), Linz (A. Biere), Vienna (H. Veith, U. Schmid), and Maria Gugging (T. Henzinger, K. Chatterjee) provide an exciting environment. The present project will enable us to associate our research and our students with this highly visible enterprise.

7 Comment <—— 227

Illustrate what the prospective benefits of your project are and when they are to be expected. Also outline what niche your project's results might fill, possibly referring to what the respective market or societal needs are.

8 Gender Management Policy (up to 500 characters) <—— 495

We commit ourselves to a family-friendly environment. As the PI is a young father himself, he understands the needs of parents for flexible work schedule. This will be reflected in our policy on meeting and work hours. We have positive experience of working with Annu Gmeiner, a student of ours who is finishing her PhD thesis after a maternity leave. Out of 14 students fully funded by the doctoral college LogiCS, seven are female, and 55% of the PhD students at Forsyte group are female.

9 Plan for Human Resources Development (up to 500 characters) <—— 488

We will support at least two dissertations. As the students will be affiliated with the doctoral college LogiCS, they will receive access to the lively academic environment that fosters scientific exchanges and timely career planning.

APALACHE is central in career development of Igor Konnov (PI). The project will contribute to Igor's habilitation. This project will enable Igor Konnov (PI) and Josef Widder (co-PI) to fully integrate into the RiSE Network as independent researchers.

10 Igor Konnov: Role and Responsibilities (<= 500) <—— 460

Igor Konnov is the principal investigator in the project. He will be responsible for the research agenda, project planning, coordination within the project, and student supervision. He is also a key researcher in the work packages B and C. In doing so, Igor will supervise the implementation of the APALACHE tools, and their evaluation.

Igor will lead the projects dissemination agenda by organizing workshops and visits of international experts at TU Wien.

11 Competencies (<= 500) <—— 457

Igor Konnov has developed groundbreaking techniques for parameterized model checking based on induction and abstraction. In addition to theoretical contributions, Igor proved strong software engineering skills by implementing these techniques in practical tools. During his time at Moscow State University and TU Wien he advised students at different levels, ranging from Bachelor to PhD.

He is the principal author of the tools CheAPS and ByMC (see CV).

12 Comment <—— 37

Specific competencies for the project

13 Josef Widder: Role and Responsibilities (<= 500) <—— 470

Josef will lead the effort in identifying patterns in TLA+ designs which requires domain specific knowledge, and he will work in parameterized model checking methods.

Josef serves as a link to the more theoretically leaning FWF project PRAVDA, which develops parameterized model checking methods for fault-tolerant distributed algorithms. In that, the projects will be able to exchange case studies, newly designed methods, etc. from which both projects will benefit.

14 Competencies (<= 500) <—— 499

Josef is an expert in distributed computing theory, especially in fault-tolerant distributed algorithms. In that, he has invaluable knowledge on the application domain of TLA+ tools. He has invented several fault-tolerant distributed algorithms and has analyzed many more.

Since 2011, Josef works on parameterized model checking. His combined background in distributed algorithms and model checking allowed him to develop verification techniques based on abstraction and partial order reduction.

15 Helmut Veith: Role and Responsibilities (<= 500) <—— 434

Helmut Veith is a collaborator and the leader of the hosting group. As the speaker of doctoral college Logical Methods in Computer Science (LogiCS) and FWF-funded National Research Network RiSE, and a co-chair of the Vienna Center for Logic and Algorithms, he will facilitate research collaborations. Helmut Veith will advise the PI and co-PI on the well-balanced scientific planning and resource management in the APALACHE project.

16 Competencies (<= 500) <—— 499

Helmut Veith is an international leader in computer-aided verification. He authored over 120 refereed publications in top venues on computer-aided verification, computer security, software engineering, database theory, distributed algorithms. He co-invented counterexample-guided abstraction refinement (the original paper is cited 1388 times).

He has led many successful research projects including WWTF project Proseed and the FWF national research network RiSE (recently extended until 2019).

17 Comments <—— 217

Helmut will coordinate activities and synergies between RiSE and APALACHE.

In the FWF-funded National Research Network RiSE, Helmut's group is working on synthesis methods for fault-tolerant distributed algorithms.

References (max. 2 pages as required by WWTF)

As required by WWTF, we highlight in bold ten key publications representing the state of the art

[Abd10] **P. A. Abdulla. "Well (and better) quasi-ordered transition systems". In: *Bulletin of Symbolic Logic* 16.4 (2010), pp. 457–515**

[AG+12] F. Alberti, S. Ghilardi, E. Pagani, S. Ranise, and G. P. Rossi. "Universal Guards, Relativization of Quantifiers, and Failure Models in Model Checking Modulo Theories". In: *JSAT* 8.1/2 (2012), pp. 29–61

[AK86] K. Apt and D. Kozen. "Limits for automatic verification of finite-state concurrent systems". In: *IPL* 15 (1986), pp. 307–309

[AL+08] M. Archer, H. Lim, N. A. Lynch, S. Mitra, and S. Umeno. "Specifying and proving properties of timed I/O Automata using Tempo". In: *Design Autom. for Emb. Sys.* 12.1–2 (2008), pp. 139–170

[BBW14] J. Birgmeier, A. R. Bradley, and G. Weissenbacher. "Counterexample to induction-guided abstraction-refinement (CTIGAR)". In: *CAV*. Springer. 2014, pp. 831–848

[BF+08] S. Bardin, A. Finkel, J. Leroux, and L. Petrucci. "FAST: acceleration from theory to practice". In: *STTT* 10.5 (2008), pp. 401–424

[BL03] B. Batson and L. Lamport. "High-level specifications: Lessons from industry". In: *Formal methods for components and objects*. Springer. 2003, pp. 242–261

[BLR11] T. Ball, V. Levin, and S. K. Rajamani. "A decade of software model checking with SLAM". In: *Communications of the ACM* 54.7 (2011), pp. 68–76

[CG+03] **E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. "Counterexample-guided abstraction refinement for symbolic model checking". In: *J. ACM* 50.5 (2003), pp. 752–794**

[CGR07] T. D. Chandra, R. Griesemer, and J. Redstone. "Paxos made live: an engineering perspective". In: *PODC*. ACM. 2007, pp. 398–407

[CL+99] M. Castro, B. Liskov, et al. "Practical Byzantine fault tolerance". In: *OSDI*. Vol. 99. 1999, pp. 173–186

[CMP04] C.-T. Chou, P. Mannava, and S. Park. "A Simple Method for Parameterized Verification of Cache Coherence Protocols". In: *FMCAD*. Vol. 3312. LNCS. Springer Verlag, 2004, pp. 382–398

[CS09] B. Charron-Bost and A. Schiper. "The Heard-Of model: computing in distributed systems with benign faults". In: *Distributed Computing* 22.1 (2009), pp. 49–71

[CT+04] E. Clarke, M. Talupur, T. Touili, and H. Veith. "Verification by Network Decomposition". In: *CONCUR 2004*. Vol. 3170. 2004, pp. 276–291

[CTV08] E. Clarke, M. Talupur, and H. Veith. "Proving Ptolemy right: the environment abstraction framework for model checking concurrent systems". In: *TACAS'08/ETAPS'08*. Springer, 2008, pp. 33–47

[DH+14] C. Dragoi, T. A. Henzinger, H. Veith, J. Widder, and D. Zufferey. "A Logic-based Framework for Verifying Consensus Algorithms". In: *VMCAI*. Vol. 8318. LNCS. 2014, pp. 161–181

[EK03] E. A. Emerson and V. Kahlon. "Exact and Efficient Verification of Parameterized Cache Coherence Protocols". In: *CHARME*. Vol. 2860. LNCS. 2003, pp. 247–262

[EN95] E. Emerson and K. Namjoshi. "Reasoning about rings". In: *POPL*. 1995, pp. 85–94

[FKP15] A. Farzan, Z. Kincaid, and A. Podelski. "Proof Spaces for Unbounded Parallelism". In: *POPL*. ACM. 2015, pp. 407–420

[GBM08] R. Geambasu, A. Birrell, and J. MacCormick. "Experiences with formal specification of fault-tolerant file systems". In: *DSN*. IEEE. 2008, pp. 96–101

[GD+11] H. S. Gunawi et al. "FATE and DESTINI: A Framework for Cloud Recovery Testing". In: *NSDI*. 2011

[GK+14] A. Gmeiner, I. Konnov, U. Schmid, H. Veith, and J. Widder. "Tutorial on Parameterized Model Checking of Fault-Tolerant Distributed Algorithms". In: *Formal Methods for Executable Software Models*. LNCS. Springer, 2014, pp. 122–171

[GL03] E. Gafni and L. Lamport. "Disk paxos". In: *Distributed Computing* 16.1 (2003), pp. 1–20

[GS92] **S. M. German and A. P. Sistla. "Reasoning about systems with many processes". In: *J. ACM* 39 (3 1992), pp. 675–735**

[GS97] S. Graf and H. Saïdi. "Construction of Abstract State Graphs with PVS". In: *CAV*. Vol. 1254. LNCS. 1997, pp. 72–83

[GY11] R. Guerraoui and M. Yabandeh. "Model Checking a Networked System Without the Network." In: *NSDI*. 2011

[HJ+02] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. "Lazy abstraction". In: *POPL*. 2002, pp. 58–70

[JK+13a] **A. John, I. Konnov, U. Schmid, H. Veith, and J. Widder. "Parameterized model checking of fault-tolerant distributed algorithms by abstraction". In: *FMCAD*. 2013, pp. 201–209**

[JK+13b] A. John, I. Konnov, U. Schmid, H. Veith, and J. Widder. “Towards Modeling and Model Checking Fault-Tolerant Distributed Algorithms”. In: *SPIN*. Vol. 7976. LNCS. 2013, pp. 209–226

[JRS11] F. P. Junqueira, B. C. Reed, and M. Serafini. “Zab: High-performance broadcast for primary-backup systems”. In: *DSN*. IEEE. 2011, pp. 245–256

[KA+07] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong. “Zyzzyva: speculative byzantine fault tolerance”. In: *ACM SIGOPS OS Review*. Vol. 41. 6. ACM. 2007, pp. 45–58

[KBS08] D. Kroening, R. Bryant, and O. Strichman. *Decision procedures: an algorithmic point of view*. Springer Science & Business Media, 2008

[KG94] H. Kopetz and G. Grünsteidl. “TTP – A Protocol for Fault-Tolerant Real-Time Systems”. In: *IEEE Computer* 27.1 (1994), pp. 14–23

[KVW14] I. Konnov, H. Veith, and J. Widder. “On the Completeness of Bounded Model Checking for Threshold-Based Distributed Algorithms: Reachability”. In: *CONCUR*. Vol. 8704. LNCS. 2014, pp. 125–140

[KVW15] **I. Konnov, H. Veith, and J. Widder. “SMT and POR beat Counter Abstraction: Parameterized Model Checking of Threshold-Based Distributed Algorithms”. In: *CAV*. [accepted as a long paper, the submission is available at:** `http://forsyte.at/download/konnov-cav15.pdf`**]. 2015**

[KWR09] D. Kroening, G. Weissenbacher, and P. Ruemmer. *A Proposal for a Theory of Finite Sets, Lists, and Maps for the SMT-Lib Standard*. 7th International Workshop on Satisfiability Modulo Theories. 2009

[Lam02] L. Lamport. *Specifying systems: The TLA+ language and tools for hardware and software engineers*. Addison-Wesley Longman Publishing Co., Inc., 2002

[Lam98] L. Lamport. “The part-time parliament”. In: *ACM TOCS* 16.2 (1998), pp. 133–169

[LH+14] T. Leesatapornwongsa, M. Hao, P. Joshi, J. F. Lukman, and H. S. Gunawi. “SAMC: Semantic-aware model checking for fast discovery of deep bugs in cloud systems”. In: *OSDI*. 2014, pp. 399–414

[LLM12] M. Lesani, V. Luchangco, and M. Moir. “A Framework for Formally Verifying Software Transactional Memory Algorithms”. In: *CONCUR*. Vol. 7454. LNCS. 2012, pp. 516–530

[LY+09] H. Lin, M. Yang, F. Long, L. Zhang, and L. Zhou. “MODIST: transparent model checking of unmodified distributed systems”. In: (2009)

[Lyn96] N. Lynch. *Distributed Algorithms*. Morgan Kaufman, 1996

[MAK13] I. Moraru, D. G. Andersen, and M. Kaminsky. “There is more consensus in egalitarian parliaments”. In: *ACM SOSP*. ACM. 2013, pp. 358–372

[McM01] **K. L. McMillan. “Parameterized Verification of the FLASH Cache Coherence Protocol by Compositional Model Checking”. In: *CHARME*. Vol. 2144. LNCS. 2001, pp. 179–195**

[McM03] K. L. McMillan. “Interpolation and SAT-based model checking”. In: *CAV*. Springer. 2003, pp. 1–13

[McM14] K. L. McMillan. “Lazy annotation revisited”. In: *CAV*. Springer. 2014, pp. 243–259

[Med12] A. Medeiros. *ZooKeeper’s atomic broadcast protocol: Theory and practice*. Tech. rep. 2012

[MV12a] S. Merz and H. Vanzetto. “Automatic Verification of TLA + Proof Obligations with SMT Solvers”. In: *LPAR*. 2012, pp. 289–303

[MV12b] S. Merz and H. Vanzetto. “Harnessing SMT Solvers for TLA+ Proofs”. In: *ECEASST* 53 (2012)

[MZ11] **K. McMillan and L. Zuck. “Invisible Invariants and Abstract Interpretation”. In: *SAS*. 2011, pp. 249–262**

[NR+15] **C. Newcombe et al. “How Amazon web services uses formal methods”. In: *Communications of the ACM* 58.4 (2015), pp. 66–73**

[OO14] D. Ongaro and J. K. Ousterhout. “In Search of an Understandable Consensus Algorithm”. In: *USENIX*. 2014, pp. 305–319

[OTT09] J. W. O’Leary, M. Talupur, and M. R. Tuttle. “Protocol verification using flows: An industrial experience”. In: *FMCAD*. 2009, pp. 172–179

[PRZ01] A. Pnueli, S. Ruah, and L. Zuck. “Automatic deductive verification with invisible invariants”. In: *TACAS*. Springer, 2001, pp. 82–97

[PXZ02] A. Pnueli, J. Xu, and L. Zuck. “Liveness with (0,1,∞)- Counter Abstraction”. In: *CAV*. Vol. 2404. LNCS. 2002, pp. 93–111

[TT08] **M. Talupur and M. R. Tuttle. “Going with the Flow: Parameterized Verification Using Message Flows”. In: *FMCAD*. 2008, pp. 1–8**

[YML99] **Y. Yu, P. Manolios, and L. Lamport. “Model checking TLA+ specifications”. In: *Correct Hardware Design and Verification Methods*. Springer, 1999, pp. 54–66**

[ZP04] L. Zuck and A. Pnueli. “Model checking and abstraction to the aid of parameterized systems (a survey)”. In: *Computer Languages, Systems & Structures* 30.3 (2004), pp. 139–169

18 Explanation of Cost Planning (<= 3000) <—— 2324

In this project, we request money only for personnel costs. As explained in the overhead policy, we will use overheads to cover non-personnel costs.

We request 100% funding for the PI Igor Konnov, as his university position is funded only until December 2015. The co-PI Josef Widder recently re-joined TU Vienna Embedded Computing Systems Group, and thus has a funded university position until January 2017. Further, in February 2015, FWF decided to fund his standalone project PRAVDA that provides three years of funding for him and one PhD student. Therefore, we do not request funding for Josef Widder, who will devote his research duty of the university position to APALACHE. Helmut Veith holds a university professor position, and thus we do not request funding for him.

As APALACHE requires significant contribution in research and implementation, we request funding for two PhD students, who will have a researcher position funded from the APALACHE budget for three years. Their work on the project will constitute major parts of their PhD dissertations.

We also request funds to support two master students for one year. Such research assistant positions will allow the master students to obtain experience and skills on research with a team of experienced colleagues within a large project. We will also use overhead funds to support extra research assistant positions for master students.

Forsyte, which is a group in the Institute of Information Systems, has a well-equipped infrastructure. Thus, we do not request funds for additional equipment. Shall a necessity for additional equipment arise, we will use the overhead funds to obtain it.

We add 5% in-kind contribution to the project by Tomer Kotek and Alfons Laarman, which means that they will support the core team in our parameterized verification effort. This will result in 1-2 joint papers in the course of the project.

The PI and co-PI are organizing a workshop series on Formal Reasoning in Distributed Algorithms (FRIDA). With this money we are going to cover the costs (approx. 2000 EUR) of one invited speaker per installment during the three years of the project.

We do not explicitly ask for travel money in the budget, since travel and research visits will be paid from the overheads. The details are given in the overhead policy.

19 Overhead policy <—— 475

The University does not charge overheads for WWTF projects. Thus, we explain how we can make use of overheads below.

We plan that PI, co-PI, and the PhD students visit at most two conferences per year each, 2000 EUR per conference. This amounts to 48,000 EUR, which we plan to pay from the overheads.

We will also use overheads to pay for summer schools visited by the PhD students, additional research positions for master students, and publication costs (when needed).

20 Disclosure <—— 945

Although APALACHE is well-embedded in the research strategy of the Forsyte group, no parts of the project are subject to ongoing or granted requests for funding. There are relationships to the FWF-funded National Research Network RiSE, which considers synthesis of fault-tolerant distributed algorithms, while we consider verification of TLA+ designs. Further, FWF standalone project PRAVDA focuses on mathematical techniques and the theoretical approach to parameterized model checking of very specific fault-tolerant distributed algorithms. In contrast, APALACHE leans towards practically-oriented model checking of TLA+ specifications that are not restricted to any kind of algorithms.

Further, we have applied for funding of an FFG Exploratory Project GridGames that lies in the intersection of model checking and techno-economical systems design with game-theoretic models. GridGames targets a specific application domain of Smart Grids.

21 Comment <—— 128

Disclosure of other applications for funding: Is the project (or parts of it) already subject to ongoing requests for funding?

Igor KONNOV

Vienna University of Technnology
Institute of Information Systems 184/4
Formal Methods in Systems Engineering
Favoritenstraße 9-11, 1040 Wien, Austria
Homepage: http://forsyte.at/konnov

Higher Education

Oct 2003– Nov 2008. Lomonosov Moscow State University (Russia): Ph.D. in Computer Science

Sep 1998– Jul 2003. Lomonosov Moscow State University (Russia): Specialist (approx. Master of Science)

Appointments

Vienna University of Technology, TU Wien (Austria):

Since Dec 2011. Postdoctoral assistant professor (Universitätsassistent)

Jul 2011 – Dec 2011. Postdoctoral research assistant (Projektassistent)

Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics (Russia):

Jan 2010 – Jun 2011. Junior research fellow

Dec 2006 – Jan 2010. Programmer with research and teaching duties

Sytech LLC (Russia): Part-time Systems Architect **(2006–2010)**, Software Developer **(2004–2006)**

Funded Projects

2011-2014. WWTF: Vienna Science and Technology Fund. Project PROSEED, 598.000 €
Role: researcher, **Coordinator**: H. Veith

2010-2014. FWF: Austrian National Research Network S11403-N23 RiSE
Role: researcher, **PI**: H. Veith (582.8k €), **Coordinator**: R. Bloem (3,736k €)

2010-2012. Russian Federal Special-Purpose Programme: Project 14.740.11.0399 Development of a Prototype for Computer Simulation of Real-Time Distributed Systems, 8.9 Mio. Rub. (approx. 200,000 €)
Role: responsible for coordination, research agenda, and report writing, **PI**: R.L. Smeliansky

Relevant Activities/Experience

Guest co-editor: Special issue on Computer Aided Verification'13, Formal Methods in System Design, Springer

Workshop chair of CAV'13: Conference on Computer Aided Verification (Jul 13-19, **2013**)

Workshop co-organizer: Formal Reasoning in Distributed Algorithms (FRIDA)
(FRIDA'15 co-located with FORTE'15 on Jun 5, **2015**, and FRIDA'14 co-located with CAV'14 on Jul 23-24, **2014**)

Tool author: Byzantine **M**odel **C**hecker [1], **Che**cker of **A**synchronous **P**arameterized **S**ystems [2]

Invited talks & lectures: Seminar on Distributed Cloud Computing, Dagstuhl/Germany (Feb **2015**); Tools & Methods of Program Analysis'14, Kostroma/Russia (Nov **2014**); Lecture at Summer School on Verification Technology, Systems & Applications, Luxembourg (Oct **2014**); Seminar on Formal Verification of Distributed Algorithms, Dagstuhl/Germany (Apr **2013**); Computing Laboratory, Oxford/UK (Feb **2011**)

Teaching: assisting in courses on Computer Aided Verification, Software and Systems Verification, Formal Methods of Informatics

Co-advising: 2 PhD (TU Wien), 5 master (TU Wien, MSU), and 3 bachelor students (TU Wien, MSU)

[1] URL: http://forsyte.tuwien.ac.at/software/bymc
[2] URL: http://lvk.cs.msu.ru/~konnov/cheaps

Ten peer-reviewed publications in the last five years

(An asterisk (*) highlights the publications most relevant to the proposal.)

Journal papers [5, 10, 9]

Book [1]

Book chapter [3]

Conference papers [2, 4, 6, 7, 8]

List of Publications

[1] *Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. *Decidability of Parameterized Verification*. Synthesis Lectures on Distributed Computing Theory. (accepted for publication). Morgan & Claypool Publishers.

[2] *Igor Konnov, Helmut Veith, and Josef Widder. "SMT and POR beat Counter Abstraction: Parameterized Model Checking of Threshold-Based Distributed Algorithms". In: *CAV*. [accepted as a long paper, the original submission is available at: http://forsyte.at/download/konnov-cav15.pdf]. 2015.

[3] *Annu Gmeiner, Igor Konnov, Ulrich Schmid, Helmut Veith, and Josef Widder. "Tutorial on Parameterized Model Checking of Fault-Tolerant Distributed Algorithms". In: *Formal Methods for Executable Software Models*. LNCS. Springer, 2014, pp. 122–171. DOI: 10.1007/978-3-319-07317-0_4.

[4] *Igor Konnov, Helmut Veith, and Josef Widder. "On the Completeness of Bounded Model Checking for Threshold-Based Distributed Algorithms: Reachability". In: *CONCUR 2014 — Concurrency Theory*. Ed. by Paolo Baldan and Daniele Gorla. Vol. 8704. Lecture Notes in Computer Science. 2014, pp. 125–140. ISBN: 978-3-662-44583-9. DOI: 10.1007/978-3-662-44584-6_10.

[5] I.V. Konnov, V.V. Podymov, D.Yu. Volkanov, V.A. Zakharov, and D.A. Zorin. "How to Make a Simple Tool for Verification of Real-Time Systems". In: *Automatic Control and Computer Sciences* 48.7 (2014), pp. 534–542. DOI: 10.3103/S0146411614070232.

[6] *Annu John, Igor Konnov, Ulrich Schmid, Helmut Veith, and Josef Widder. "Brief announcement: parameterized model checking of fault-tolerant distributed algorithms by abstraction". In: *PODC*. 2013, pp. 119–121. DOI: 10.1145/2484239.2484285.

[7] *Annu John, Igor Konnov, Ulrich Schmid, Helmut Veith, and Josef Widder. "Parameterized model checking of fault-tolerant distributed algorithms by abstraction". In: *FMCAD*. 2013, pp. 201–209. DOI: 10.1109/FMCAD.2013.6679411.

[8] *Annu John, Igor Konnov, Ulrich Schmid, Helmut Veith, and Josef Widder. "Towards Modeling and Model Checking Fault-Tolerant Distributed Algorithms". In: *SPIN*. Vol. 7976. LNCS. 2013, pp. 209–226. DOI: 10.1007/978-3-642-39176-7_14.

[9] *Igor V. Konnov and Vladimir A. Zakharov. "An invariant-based approach to the verification of asynchronous parameterized networks". In: *J. Symb. Comput.* 45.11 (2010). Elsevier, pp. 1144–1162. DOI: 10.1016/j.jsc.2008.11.006.

[10] I. V. Konnov. "On application of weaker simulations to parameterized model checking by network invariants technique". In: *Automatic Control and Computer Sciences* 44.7 (2010), pp. 378–386. DOI: 10.3103/S0146411610070035.

Josef WIDDER

Vienna University of Technnology
Institute of Computer Engineering 182/2
Embedded Computing Systems Group
Treitlstraße 3, 2nd floor, 1040 Wien, Austria
Homepage: http://forsyte.at/widder

Higher Education

2002 – 2004. Vienna University of Technology (TU Vienna): Ph.D. in Computer Science

1996 – 2002. TU Vienna: Dipl.-Ing. in Computer Science (Master of Science)

Appointments

since 2015 Assistant professor (Univ.-Ass.). TU Vienna. Embedded Computing Systems. Ulrich Schmid.

2011 – 2015 PostDoc. TU Vienna. Forsyte Group. Helmut Veith.

2010 – 2011 PostDoc. Texas A&M University. Parasol Lab. Jennifer L. Welch.

2009 – 2010 Assistant professor (Univ.-Ass.). TU Vienna. Embedded Computing Systems. Ulrich Schmid.

2007 – 2009 PostDoc. Ecole polytechnique. Laboratoire d'Informatique LIX. Bernadette Charron-Bost.

2003 – 2007 Research assistant and Assistant professor (Univ.-Ass.). TU Vienna. Automation and Embedded Computing Systems. Ulrich Schmid.

Funded Projects

starting 2015: FWF Stand-alone Project. "Parametrized Verification of Fault-tolerant Distributed Algorithms", 334k EUR
Role: PI

since 2013: TU Vienna, Innovative Projects, "Program Analysis for Automated Liveness verification of Distributed Algorithms", 100k EUR.
Role: co-PI

2004: FIT-IT PhD stipend, "Distributed Computing in the Presence of Bounded Asynchrony", 56k EUR.
Role: PI

Relevant Activities/Experience

Related Workshops co-organized: Dagstuhl Seminar 13141 "Formal Verification of Distributed Algorithms", and FRiDA workshop (Formal Reasoning in Distributed Algorithms) at FLoC 2014 and FORTE 2015.

Research Visits: In 2013 visiting Stephan Merz (LORIA Nancy), who is participating in the development of the TLA proof system. Other visits at Texas A&M University, IRISA Rennes, RWTH Aachen, INRIA Rocquencourt.

PC Chair: Vice Chair of SSS 2009 and Local Topic Chair at Euro-Par 2015.

PC membership: ICDCN 2015, SSS 2006, 2012, 2013, EDCC 2010, 2012.

Invited Talks at summer school SFM 2014, Journées Internationales sur l'auto-stabilisation 2002, International Workshop-Conference TMPA-2014, two Dagstuhl seminars, RWTH Aachen, LIP6 and 6 more.

Teaching: several courses at TU Vienna on topics including program and system verification, dependable and distributed systems, and real-time scheduling

10 Publications from the last Five Years

(An asterisk (*) highlights the publications most relevant to the proposal.)

Journal papers [1, 6, 10]

Book chapter [4]

Conference papers [2, 5, 3, 8, 7, 9]

List of Publications

[1] Bernadette Charron-Bost, Matthias Függer, Jennifer L. Welch, and Josef Widder. “Time Complexity of Link Reversal Routing”. In: *ACM Trans. Algorithms* 11.3 (Jan. 2015), 18:1–18:39.

[2] *Igor Konnov, Helmut Veith, and Josef Widder. “SMT and POR beat Counter Abstraction: Parameterized Model Checking of Threshold-Based Distributed Algorithms”. In: *CAV’15*. [accepted as a long paper, the original submission is available at: `http://forsyte.at/download/konnov-cav15.pdf`]. 2015.

[3] *Cezara Drăgoi, Thomas A. Henzinger, Helmut Veith, Josef Widder, and Damien Zufferey. “A Logic-based Framework for Verifying Consensus Algorithms”. In: *VMCAI*. Vol. 8318. LNCS. 2014, pp. 161–181.

[4] *Annu Gmeiner, Igor Konnov, Ulrich Schmid, Helmut Veith, and Josef Widder. “Tutorial on Parameterized Model Checking of Fault-Tolerant Distributed Algorithms”. In: *Formal Methods for Executable Software Models - 14th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2014, Advanced Lectures*. Vol. 8483. LNCS. Springer, 2014, pp. 122–171.

[5] *Igor Konnov, Helmut Veith, and Josef Widder. “On the Completeness of Bounded Model Checking for Threshold-Based Distributed Algorithms: Reachability”. In: *CONCUR*. Vol. 8704. LNCS. 2014, pp. 125–140.

[6] Bernadette Charron-Bost, Antoine Gaillard, Jennifer Welch, and Josef Widder. “Link Reversal Routing with Binary Link Labels: Work Complexity”. In: *SIAM Journal on Computing* 42.2 (2013), pp. 634–661.

[7] *Annu John, Igor Konnov, Ulrich Schmid, Helmut Veith, and Josef Widder. “Parameterized model checking of fault-tolerant distributed algorithms by abstraction”. In: *FMCAD*. 2013, pp. 201–209.

[8] *Annu John, Igor Konnov, Ulrich Schmid, Helmut Veith, and Josef Widder. “Towards Modeling and Model Checking Fault-Tolerant Distributed Algorithms”. In: *SPIN*. Vol. 7976. LNCS. 2013, pp. 209–226.

[9] *Matthias Függer and Josef Widder. “Efficient Checking of Link-Reversal-Based Concurrent Systems”. In: *CONCUR*. Vol. 7454. LNCS. 2012, pp. 486–499.

[10] Josef Widder, Martin Biely, Günther Gridling, Bettina Weiss, and Jean-Paul Blanquart. “Consensus in the presence of mortal Byzantine faulty processes”. In: *Distributed Computing* 24.6 (2012), pp. 299–321.

HELMUT VEITH

Formal Methods in Systems Engineering Group 184/4
Institute of Information Systems
Vienna University of Technology (TU Wien)
Favoritenstraße 9–11
A-1040 Vienna

(+43) 1 58801 18441
(+43) 1 58801 18493
http://www.forsyte.at/people/veith
veith@forsyte.at

APPOINTMENTS AND EDUCATION

2010-	Full professor, TU Wien, Austria
2008-2009	Professor (W3), TU Darmstadt, Germany
since 2005	Adjunct Full Professor, Carnegie Mellon University, Pittsburgh, USA
2003-2007	Professor (C3), TU Munich, Germany
2001-2003	Associate Professor, TU Wien, Austria
2001	Habilitation in Applied and Theoretical Computer Science, TU Wien
1999-2000	Visiting Scientist, Carnegie Mellon University, Pittsburgh, USA
1998	Doctoral defense *sub auspiciis praesidentis*, TU Wien
1995-2001	Teaching and Research Assistant (Universitätsassistent), TU Wien
1994	Diplom-Ingenieur (equivalent to M.Sc.) in Computational Logic, TU Wien
1989	High school graduation, BG/BRG Tulln, Austria

FUNDED PROJECTS (MAXIMUM 3)

Name	Running time	Funding source	funding sum	Role
Doctoral College *Logical Methods in Computer Science*	2014-2018	FWF	2.7 Mio EUR 300 K EUR for PI	speaker
National Research Network *Rigorous Systems Engineering*	2011-2019	FWF	7.5 Mio EUR 1.0 Mio EUR for PI	vice speaker
PROSEED (*Proof Seeding*)	2011-2014	WWTF	600 K EUR	PI

CAREER-RELATED ACTIVITIES

- Vice chair of Vienna Summer of Logic 2014, and chair of FLoC 2014
  Vienna Summer of Logic was the largest conference in the history of logic – vsl2014.at
- Program co-chair, FMCAD 2016, CAV 2013, St. Petersburg, CSL 2010, Brno, and LPAR 2008, Qatar; Track Chair for Formal Methods for SSS 2015
- Editor (with Ed Clarke and Tom Henzinger), *Handbook of Model Checking*, Springer 2015
- Editor (with Orna Grumberg), *25 Years of Model Checking*, Springer 2009
- Editorial Board Member of *Acta Informatica* and *Modeling and Analysis of Information Systems*
- Program committee member in more than 50 international conferences
- Invited speaker: PSI 2015, ICST 2015, iFM 2014/FACS 2014, VTSA 2014, Haifa Verification Conference 2009, LPAR 2008, European Forum Alpach 2008, CSL 2003, and at multiple workshops and symposia
- ACM SIGSOFT Distinguished Paper Award 2003
- More than 120 publications in computer-aided verification, logic in computer science, software engineering, computer security
- Seven graduated PhD students (four with faculty positions), five current PhD students.
- Co-chair, Vienna Center for Logic and Algorithms (VCLA, *www.vcla.at*)
- Co-Founder and Steering Committee Member, Austrian Computer Science Day
- Steering Committee Member, European Association for Computer Science Logic (EACSL)
- Steering Committee Member, International Kurt Gödel Society
- Long term involvement in TEMPUS projects for development of IT in Uzbekistan
- Member of the Academic Senate of TU Vienna (currently vice-speaker of the professors)
- Coordinator of the Research Focus on Logic and Computation, Faculty of Informatics, TU Vienna
- Member of the Curriculum Committee, Faculty of Informatics, TU Vienna
- Member of Faculty Council of the Faculty of Informatics, TU Vienna
- Board Member, PhD School for Informatics, TU Vienna

TEN IMPORTANT PUBLICATIONS

[1] Moritz Sinn, Florian Zuleger, and Helmut Veith. A Simple and Scalable Static Analysis for Bound Analysis and Amortized Complexity Analysis. In *CAV*, pp. 745-761, 2014.

[2] * Benjamin Aminof, Tomer Kotek, Sasha Rubin, Francesco Spegni, and Helmut Veith. Parameterized Model Checking of Rendezvous Systems. *CONCUR*, pp. 109–124, 2014.

[3] * Igor Konnov, Helmut Veith, and Josef Widder. On the Completeness of Bounded Model Checking for Threshold-Based Distributed Algorithms: Reachability. In *CONCUR*, pp. 125–140, 2014.

[4] * Cezara Dragoi, Thomas A. Henzinger, Helmut Veith, Josef Widder, and Damien Zufferey. A logic-based framework for verifying consensus algorithms. In *VMCAI*, pp. 161–181, 2014.

[5] * Igor Konnov, Helmut Veith, and Josef Widder. SMT and POR beat Counter Abstraction: Parameterized Model Checking of Threshold-Based Distributed Algorithms. In *CAV*, [accepted as a long paper, the submission is available at: *http://forsyte.at/download/konnov-cav15.pdf*], 2015.

[6] * Annu John, Igor Konnov, Ulrich Schmid, Helmut Veith, and Josef Widder. Parameterized model checking of fault-tolerant distributed algorithms by abstraction. In *FMCAD*, pp. 201–209, 2013.

[7] * Annu John, Igor Konnov, Ulrich Schmid, Helmut Veith, and Josef Widder. Towards Modeling and Model Checking Fault-Tolerant Distributed Algorithms. In *SPIN*, pp. 209–226, 2013.

[8] Azadeh Farzan, Andreas Holzer, Niloofar Razavi, and Helmut Veith. Con2colic testing. In *ESEC/FSE'13*, pp. 37–47, 2013.

[9] Tomer Kotek, Mantas Simkus, Helmut Veith, and Florian Zuleger. Extending ALCQIO with Trees: Between Finite Model Theory and Description Logic. In *LICS*, to appear, 2015.

[10] * Marko Samer and Helmut Veith. On the distributivity of LTL specifications. *ACM Trans. Comput. Log.*, 11(3), 2010.