ERC Advanced Grant 2015

Research Proposal (Part B1)

Principal investigator: Helmut Veith

Host institution: TU Wien (Technische Universität Wien), Austria

Proposal full title: Harnessing Model Checking for Distributed Algorithms

Proposal short name: HYDRA

Project duration: 60 months

Proposal summary

Distributed algorithms have numerous mission-critical applications in embedded avionic and automotive systems, cloud computing, computer networks, hardware design, and the internet of things. Although distributed algorithms exhibit complex interactions with their computing environment and are difficult to understand for human engineers, computer science has developed only very limited tool support to catch logical errors in distributed algorithms at design time.

The HYDRA project aims to build the theory and the practical tools for the formal verification of distributed algorithms by model checking. Recent work by the applicant's team has demonstrated that the progress of the last two decades in abstract model checking, automated theorem proving, and partial order reduction gives leverage for parameterized model checking techniques that are able to verify, for the first time, non-trivial fault tolerant distributed algorithms.

INDEFESSA GERENS REDIVIVIS BELLA COLVBRIS ARGOLIS AD LERNÆ TVNDITVR HYDRA VADVM

Hydra – Middle English Ydra, from Latin Hydra, from Greek $'\upsilon\delta\rho\alpha$
1. a many-headed serpent or monster in Greek mythology each head of which when cut off was replaced by two others
2. a multifarious evil not to be overcome by a single effort

Section a: Extended Synopsis

Project introduction: criticality and complexity of distributed algorithms

This project is concerned with the automated verification of distributed algorithms by model checking at design time. Before we discuss our vision for model checking, we introduce distributed algorithms by example.

Safety critical systems such as airplanes are highly dependent on sensor data. To avoid disasters caused by faulty data or processing, they replicate their components to minimize the probability of failure. When the replication mechanisms have logical design errors themselves, the consequences can be dramatic: In 2008, the failure of a single computing component in an Airbus operated by Qantas caused a dangerous altitude drop which resulted in serious injuries of twelve people. Although the component was triply replicated, the control system did not recognize the deviation of one component, and operated on wrong data.

This accident illustrates the consensus problem, a central challenge in distributed computing theory: *How can we enable n replicated components to agree on a value v although $t \leq n$ of the components may be faulty?* It is intuitively clear that the fraction $\frac{t}{n}$ describes the cost-reliability trade-off: Investing into n components which outnumber the t potentially faulty components is costly (and, in case of airplanes, heavy), but allows to tolerate more severe faults. Thus, we need a precise understanding about the minimum n necessary to tolerate a required t. In cloud computing and data centers, n is so large that faults are the norm rather than the exception, and t is empirically known. While failures are not as safety-critical as in airplanes, the economic consequences of downtimes and data losses have raised the awareness for fault tolerant computing in the cloud industry [25].

Algorithm 1: Bosco: one-step asynchronous Byzantine consensus [30]

```
  Input: $v_p$
1 broadcast $\langle$VOTE, $v_p\rangle$ to all processors;
2 wait until $n - t$ VOTE messages have been received;
3 if more than $\frac{n+3t}{2}$ VOTE messages contain the same value $v$ then
4     DECIDE($v$);
5 if more than $\frac{n-t}{2}$ VOTE messages contain the same value $v$,
6   and there is only one such value $v$ then
7     $v_p \longleftarrow v$;
8 Underlying-Consensus($v_p$);
```

Let us illustrate the consensus problem with the well-known BOSCO algorithm [30]. The algorithm runs on each of the n components, and communicates its own (Boolean) value v_p via a broadcast message to the other components. Since it is aware of the t possible faults, it is able to count the messages and values received from other components, and to take conservative decisions.

For correct functioning, the algorithm assumes the "resilience condition" $n > 3t$. Moreover, if $n > 5t$, then the algorithm guarantees very fast consensus in the absence of faults; if $n > 7t$, the algorithm is fast even in the presence of faults. The correctness of the BOSCO algorithm is non-trivial and proven in [30].

Although the pseudocode of BOSCO may look simple at first sight (and is simple by the standards of distributed algorithms), it relies on a list of explicit and implicit requirements: (1) it uses a message passing mechanism where messages are delivered with arbitrary delay, (2) faults are Byzantine, i.e., processes may fail arbitrarily, (3) each process can directly broadcast to all other processes without relaying, (4) the process interleaving is asynchronous except for the function called in line 8 which requires "partial synchrony", i.e., assumptions on the fairness of the scheduler, and (5) the time to reach the consensus is important. Moreover, the algorithm implicitly uses (6) process IDs to distinguish senders of messages, (7) loops and data structures to collect incoming messages and evaluate the conditions in lines 3 and 5, and (8) cryptographic signatures to prevent faulty processes from pretending to be someone else.

Other distr. algorithms, including [4, 32] by our collaborator U. Schmid, show a similar pattern of challenges:

	TORA [27]	XTC [33]	SCHyb [4]	TopCon [32]	BOSCO [30]
Task	*DAG for routing*	*Overlay graph*	*Consensus*	*FT Overlay*	*Consensus*
1. Message Passing	yes	yes	yes	yes	yes
2. Fault Tolerance	crash / links	crash	crash to Byzantine	crash	Byzantine
3. Com. Graph	yes	yes	clique	yes	clique
4. Partial Synchrony	no	no	synchronous	failure detector	line 8
5. Liveness	yes	yes	yes	complicated proof	fast decision
6. Process IDs	yes	yes	yes	yes	yes
7. Data Structures	tuples	ordered sets	sets, lists, trees	sets	sets
8. Signatures	no	no	yes*	no	implicit
9. Real-time & Hybrid	uses GPS time	geo routing	no	no	no
10. Probabilities	no	no	no	no	no

* Paper [4] contains several algorithms: one with signatures, one with restricted forms of signatures, one without.

Cloud infrastructure, many-core computation, hardware design, the internet of things and autonomous embedded systems have a trade-off between criticality, complexity, risks, and human engineering effort that makes a strong case for the automated verification of distributed algorithms.

Project motivation: a tool chain under construction

In a recent CACM paper [25], a team of engineers at Amazon describes their use of formal methods for the design and verification of distributed algorithms:

"In order to find subtle bugs in a system design, it is necessary to have a precise description of that design. There are at least two major benefits to writing a precise design: the author is forced to think more clearly, helping eliminate 'plausible hand waving,' and tools can be applied to check for errors in the design, even while it is being written. [. . .] As our designs are unavoidably complex, we needed a highly expressive language, far above the level of code, but with precise semantics. That expressivity must cover real-world concurrency and fault tolerance."

The reality of distributed algorithms research (as represented by conferences such as PODC, DISC, SSS) is quite different. The mainstream of the research community is focusing on new algorithmic solutions, complexity analysis of algorithms, and impossibility results. For many papers and textbooks, pseudocode, typically accompanied by informally stated mathematical assumptions on the environment, is the natural mode of presentation for the algorithms. As discussed in textbooks such as [1], distributed algorithms are "*notoriously difficult*" and "*notorious for their surplus of models*". In the presence of nondeterminism, asynchronicity, as well as communication and fault models in large variety, the pseudocode makes it possible to focus on the mathematical essence of the algorithm. In particular, the pseudocode reflects the generality of an algorithm which is disguised in the details of an implementation for a specific target platform and domain.

Pseudocode and natural language have obvious disadvantages, too: Without a rigorously defined syntax and semantics, the distributed algorithm cannot be understood by a computer. Thus, it is impossible to (i) simulate an algorithm, (ii) verify, debug and test the algorithm, (iii) synthesize code in a standard programming language, (iv) optimize / transform / modify the algorithm, (v) synthesize missing parts, (vi) maintain a repository of distributed algorithms, and (vii) to compare and benchmark computer-aided tools. In the HYDRA project, we focus on (ii), (vi), (vii), but are concerned, in varying depth, with all topics, and will enable other research groups to contribute to them.

The need for a formalization on the abstraction level of pseudocode is of course not a new insight. Already in the 1980s, Nancy Lynch and 2013 Turing Award winner Leslie Lamport, the top distributed algorithms researchers of their generation, initiated **I/O automata (IOA)**[1] and the **Temporal Logic of Actions, TLA+**[2]. Importantly, IOA and TLA+ do not describe implementations, but high-level models that are able to capture the essence and complexity of an algorithm. In fact, the quote by Amazon above is a laudatio of TLA+.

IOA and TLA+ have had enormous intellectual impact. IOA are frequently used in research papers, yet more as a paper-and-pencil model than a framework for computer-aided analysis. To the best of our knowledge, the rich IOA tool landscape (including a simulator, code generator, connection to the Larch theorem prover and to Coq, translator to TLA+) has no model checker, and has not been actively developed since 2005. Similarly, TLA+ is enjoying popularity as a proof technique, but model checking tool support is confined to the explicit-state model checker TLC. Current work is focusing on connecting the TLA+ Proof System to the proof assistant Isabelle and SMT solvers [9]. Thus, in spite of their methodological merit, the state of tool support for IOA and TLA+ has stalled out with explicit-state model checking and proof assistants. Explicit-state model checking supports debugging only for a fixed (small) number of (simple) processes, and suffers from capacity limitations that make verification of anything but the most simplified models impractical. Proof assistants suffer from capability limitations of the user, as higher order theorem proving remains a domain for experts only.

We believe that the lack of good model checkers in the 1990s is one main reason for the current state of the tool chain. In the early days of CAV and PODC – the premier research conferences in model checking and distributed algorithms – there was frequent interaction [20, 29, 8] – but after that, the communities, unfortunately, grew apart. In recent years, CAV invited outstanding PODC researchers to close this gap [34, 21].

When IOA and TLA+ were designed, model checking technology was not sufficiently mature to verify interesting distributed algorithms: even the verification of systems with 10 processes was a research challenge. Consequently, the potential of systematic formalization was not fully realized, and the perception of model checking in the distributed algorithms community was fairly negative.

As we argue below, the situation has changed: Model checking is ready for the challenge. The availability of reliable verification tools for distributed algorithms will increase the interest in formal modeling.

The vision of HYDRA is to develop, for the first time, an automated verification framework that is able to verify a significant class of realistic distributed algorithms.

[1]**IOA Language and Toolset** `http://groups.csail.mit.edu/tds/ioa/` includes more references

[2]**The TLA Home Page** `http://research.microsoft.com/en-us/um/people/lamport/tla/tla.html`

Project background: from textbook model checking to parameterized model checking

Model checking is a method for static verification of systems at design time. Invented in 1981 and recognized with a Turing award in 2007 [7], model checking found its way from theoretical computer science to practical applications in hardware, software, and embedded systems [B4]. In the original formulation of model checking, a system is represented as a finite state machine S, and a specification is written as a temporal logic formula φ. A model checker is an algorithmic tool to establish $S \models \varphi$, i.e., the truth of the specification on the system S. The main algorithmic challenge is *state explosion:* Since the states of S are global system states, n bits of memory result in 2^n states, and two threads of n consecutive steps have more than 2^n possible interleavings.

We can naturally distinguish three waves of model checking research that are most relevant for HYDRA:

(I) The model checkers SPIN and SMV are the paradigmatic representatives of the first generation of model checkers [22, 16]. SPIN's input language PROMELA is geared towards concurrent processes and SPIN uses partial order reductions to address state explosion. The input language of SMV is geared towards hardware, and internally it uses binary decision diagrams to fight state explosion. Both tools have much in common: they are sound and complete, they have an input language with a simple clear semantics, and they have been widely used ever since. Their main drawback is scalability: without further preprocessing, their built-in algorithms can verify only moderately sized systems. The TLA+ model checker TLC belongs to the same generation.

(II) The second generation model checkers are verifying source code, most often in C. These tools are characterized by an influx of techniques from static analysis, abstract interpretation and theorem proving, in particular SAT and SMT solving. Counterexample-guided abstraction refinement (CEGAR) is a new paradigm which uses the power of logic and theorem proving to create overapproximated abstract system models, model check them, and refine the abstraction if necessary [19, 14, 2] [M1, M2, M5]. Early example tools of this generation include Microsoft's SLAM [2], NEC's VARVEL [17], and academic tools such as BLAST [15] and MAGIC [R4]. While the second generation tools are much more powerful than their predecessors, and can handle infinite models with the help of theorem provers, they are trading soundness and completeness against practical success in debugging. This is not a voluntary choice: difficult research questions concerning the analysis of loops, dynamic data structures, and libraries make it necessary to verify source code under "optimistic assumptions" (sic). To summarize, software model checking for source code is still in the making.

(III) Parameterized model checking is concerned with models directly relevant to distributed algorithms. Let S_n denote a system composed of $n \geq 2$ processes, $S_{n,t}$ a fault-tolerant system with n processes where at most $t < n/3$ are faulty, and S_G a distributed algorithm over a finite network topology graph G. Then, the three verification questions can be stated as

$$\boxed{\forall n \geq 2 \,.\, S_n \models \varphi \qquad \text{and} \qquad \forall n \geq 2 \,.\, \forall t < \frac{n}{3} \,.\, S_{n,t} \models \varphi \qquad \text{and} \qquad \forall\, G.\, S_G \models \varphi}$$

respectively. Hence, parameterized model checking is, in a mathematically precise sense, strictly more difficult than ordinary model checking, and needs advanced techniques from logic and automata theory. Note that the model checkers above can verify such systems only for small concrete instances, e.g. $S_{10} \models \varphi$.

The state of the art in parameterized model checking combines the best and the worst of the model checking world: On the one hand, the community has developed a wealth of deep techniques including cut-offs, counter abstraction, environment abstraction, compositional methods, regular model checking, monotonic abstraction, and proof spaces [12, 11, 5, 28, 23, 24] [R6]. On the other hand, the overwhelming majority of the research is dealing with safety properties of shared memory concurrent programs; mutual exclusion and weak memory models are the preeminent research topics. Thus, they address mainly features 6 and 7 in the table on page 2. The most detrimental shortcoming is the lack of a joint input language and the lack of available tools: Since problems from distributed algorithms were too hard for model checking a generation ago, IOA and TLA+ were not on the agenda. This has led to a quite scattered research community. In recent years, initiatives such as the PV, EC2 and FRIDA workshop series (the latter coorganized by our group) helped to unify the community. In a research monograph [B3], we made an attempt to summarize the scattered results on decidability of parameterized model checking. The biggest success story in practice are cache coherence protocols: Due to urgent industrial need, hardware companies developed tools for cache coherence verification [23, 6, 31].

Main Hypothesis: **The theoretical foundations and tool support for model checking have acquired sufficient critical mass to make the parameterized verification of realistic distributed algorithms feasible.**

To achieve this goal, we will combine the soundness and completeness from the first generation model checkers, the powerful techniques developed for software model checkers, the modeling languages from distributed algorithms and application domains, and the new paradigms from parameterized model checking.

Project feasibility: verification of FTDAs

We have given a proof of concept for the main hypothesis in our recent work on **threshold-based fault-tolerant distributed algorithms (FTDAs)**. In a series of papers[3], we

(a) demonstrated correct modeling of the algorithms for small parameter values in SPIN (SPIN 2013),
(b) developed an automata model and a parametric data and counter abstraction for FTDAs (FMCAD 2013),
(c) developed an offline partial order (p.o.) reduction to reduce the abundant nondeterminism (CONCUR 2014),
(d) used SMT solving and a refined p.o. reduction for verification of several realistic FTDAs [R3] (CAV 2015).

In particular, we verified fast consensus of the BOSCO algorithm discussed on page 2. In our CAV 2015 paper, BOSCO is the most challenging benchmark: it has (i) the most severe class of faults (Byzantine), (ii) multiple resilience conditions, (iii) complex threshold expressions, and (iv) many local states (i.e. combined data and control states). Our tool verified BOSCO in under 4 hours. Since the partial order reduction is offline, it gives rise to a natural parallelization technique that will result in further dramatic improvement.

Our case study also demonstrates the importance of human guidance: Abstraction and refinement can substantially profit from auxiliary human input *if soundness is guaranteed by construction.* (Think of "hints" for the model checker.) Human guidance has been common practice in the verification of critical systems such as cache coherence protocols for a long time. We refer to this pragmatic approach as *Almost Automated Verification*.

BOSCO in fact is a good roadmark for the state of the art: In accordance with the BOSCO paper [30], our verification only concerned lines 1-7 of the algorithm, and not the procedure `Underlying-Consensus` called in line 8. Here, BOSCO transfers control to a standard (slower) consensus algorithm. As we know from [13], consensus is impossible in asynchronous systems. Thus, to verify `Underlying-Consensus`, we need to capture partially synchronous schedules in our verification tools. Like partial synchrony in row 4 of the table, *each of the 10 features in the table constitutes a research question addressed in HYDRA.*

Our main hypothesis states that the power of modern model checking methods, partial order reduction, theorem proving and abstraction methods make similar success stories for other, broader classes of distributed algorithms possible. Hence, our work on FTDAs serves as our main motivation for the HYDRA proposal.

Project objective: verification of realistic distributed algorithms
We summarize the research agenda in four objectives that will structure and guide our work:

The ***high-level objective*** is to improve the quality of critical distributed systems through rigorous algorithm design that is tightly coupled with automated verification methods. Our vision are formally verified fault-tolerant distributed algorithms which are immune against both faults and logical design errors. IOA, TLA+, and domain specific languages will bridge the gap between algorithm design and tool-supported analysis.

The ***theoretical objective***, and main challenge, is to permeate and exploit the mathematical structure inherent in distributed algorithms to make them amenable to verification. Informed by the manual proof and construction principles from distributed algorithm theory, we will develop specific automata models, partial order reductions, abstractions, decomposition and quantitative methods for use in verification algorithms.

The ***practical objective*** is to develop a model-checking based tool chain that is able to analyze a large class of distributed algorithms. To harness the full potential of state-of-the-art model checking, we will combine powerful existing tools (for instance, SAT and SMT solving) with the rich theoretical knowledge in parameterized verification. In the spirit of almost automated verification, the tools should be fully automated wherever possible, and allow human guidance where necessary.

The ***social objective*** is to foster systematic collaboration and exchange between the computer-aided verification community and the distributed algorithms community. Through the establishment of an interdisciplinary team, competitions, an algorithms repository, and strategic collaborations with researchers from both communities we will stimulate a culture of collaboration.

Technically, we will address verification in five verification themes that cover features 1–8 described in the table above. In contrast to most previous work, we will focus on message-passing, fault-tolerance and complex communication topologies. *Thus, we will use the ERC grant to explore entirely new areas of verification.*

Verification Themes

VT1	Communication in the presence of faults
VT2	Partial synchrony
VT3	Liveness and quantitative bounds
VT4	Communication graphs and process IDs
VT5	Complex data structures

[3]**Byzantine Model Checker** `http://forsyte.at/software/bymc/`

Why do we believe that HYDRA will be a game changer?

The first and obvious answer is the scale of ERC funding: The focused 5-year effort and the large team made possible by an Advanced Grant give us the leverage to foster parameterized model checking of distributed algorithms as a field of research in its own right. But why should we succeed where others have failed before?

1. It's the right time, for several reasons: (1) We have demonstrated that model checking has the critical mass to address realistic distributed algorithms. (2) Reliability of distributed algorithms is a growing industrial concern. Byron Cook (Amazon), Wilfried Steiner (TTTech), Sagar Chaki (CMU Software Engineering Institute) will collaborate with us on challenge problems from cloud computing, avionic and automotive systems, and adaptive cyberphysical systems. Mark Tuttle (Intel), a coinventor of IOA and user of TLA+, explicitly supports our plan to advance the state of the art. (3) There is a strong push towards software synthesis – exemplified by the Excape expedition Grant[4] and the Austrian Research network RiSE[4] where the applicant is the co-speaker. System synthesis frameworks such as BIP [3], Mace [18], and P [10] are an ideal match for HYDRA: To synthesise a correct implementation, we need a verified model of the distributed algorithm to start from. We expect that synthesis from verified models will also be a key technology for network verification [26].

2. We are the right team. Although the ERC Advanced Grant is awarded to individuals, verification is a team effort. (1) In the core team, we will continue our successful work with U Schmid and J Widder (distributed algorithms), I Konnov (model checking), T Kotek (logic), F Zuleger (liveness). In 2016, L Kovacs (theorem proving) will start a full professor position and an ERC Starting Grant in our group. (2) Through RiSE, HYDRA will utilize research synergies on *real time and hybrid systems* (E Bartocci, R Grosu), *probabilistic systems* (A Sokolova), and *synthesis* (T Henzinger, R Bloem). (3) Our track record demonstrates our enthusiasm for difficult interdisciplinary work and our ability to coordinate large efforts. (4) The LogiCS doctoral college will fund 2 more PhD students, and the university will finance workshops and other activities through VCLA[4].

3. It's our job. Distributed algorithms researchers started the formalization toolchain in the 1980s, but the insufficiency of model checkers hindered the effort [20]. It is our job to finish the tool chain.

Budget. The PI will dedicate 40% of his time to HYDRA. HYDRA will fund 3 postdocs and 3 PhD students.

Literature (abridged due to page limit)

[1] H. Attiya and J. Welch. *Distributed Computing*. Wiley, 2004.

[2] T. Ball, B. Cook, V. Levin, and S. K. Rajamani. SLAM and static driver verifier: Technology transfer of formal methods inside microsoft. In *IFM*, volume 2999, pages 1–20. Springer, 2004.

[3] A. Basu, S. Bensalem, M. Bozga, J. Combaz, M. Jaber, T.-H. Nguyen, and J. Sifakis. Rigorous component-based system design using the bip framework. *IEEE software*, 28:41–48, 2011.

[4] M. Biely, U. Schmid, and B. Weiss. Synchronous consensus under hybrid process and link failures. *Theor. Comput. Sci.*, 412(40):5602–5630, 2011.

[5] A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *CAV*, pages 403–418, 2000.

[6] C.-T. Chou, P. Mannava, and S. Park. A simple method for parameterized verification of cache coherence protocols. In *FMCAD*, volume 3312 of *LNCS*, pages 382–398. Springer Verlag, 2004.

[7] E. M. Clarke, E. A. Emerson, and J. Sifakis. Model checking: algorithmic verification and debugging. *CACM*, 52(11):74–84, 2009.

[8] E. M. Clarke and O. Grumberg. Avoiding the state explosion problem in temporal logic model checking. In *PODC*, pages 294–303. ACM, 1987.

[9] D. Cousineau, D. Doligez, L. Lamport, S. Merz, D. Ricketts, and H. Vanzetto. TLA + proofs. In *FM*, 2012.

[10] A. Desai, V. Gupta, E. K. Jackson, S. Qadeer, S. K. Rajamani, and D. Zufferey. P: safe asynchronous event-driven programming. In *PLDI*, pages 321–332, 2013.

[11] J. Esparza, P. Ganty, and R. Majumdar. Parameterized verification of asynchronous shared-memory systems. In *CAV*, pages 124–140. Springer, 2013.

[12] A. Farzan, Z. Kincaid, and A. Podelski. Proof spaces for unbounded parallelism. In *POPL*, 2015.

[13] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[14] S. Graf and H. Saïdi. Construction of abstract state graphs with pvs. In *CAV*, volume 1254 of *LNCS*, pages 72–83, 1997.

[15] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Lazy abstraction. In *POPL*, pages 58–70. ACM, 2002.

[16] G. J. Holzmann. *The SPIN Model Checker - primer and reference manual*. Addison-Wesley, 2004.

[17] V. Kahlon, S. Sankaranarayanan, and A. Gupta. Static analysis for concurrent programs with applications to data race detection. *STTT*, 15(4):321–336, 2013.

[18] C. E. Killian, J. W. Anderson, R. Braud, R. Jhala, and A. M. Vahdat. Mace: language support for building distributed systems. In *ACM SIGPLAN Notices*, volume 42, pages 179–188. ACM, 2007.

[19] R. Kurshan. *Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach*. Princ. Univ. Press, 1995.

[20] L. Lamport. Computer-hindered verification (humans can do it too). In *CAV*, volume 663 of *LNCS*, page 1. Springer, 1992.

[21] L. Lamport. What should math have to do with building complex distributed systems? In *CAV*, LNCS. Springer, 2015.

[22] K. L. McMillan. *Symbolic model checking*. Kluwer, 1993.

[23] K. L. McMillan. Parameterized verification of the flash cache coherence protocol by compositional model checking. In *CHARME*, volume 2144 of *LNCS*, pages 179–195, 2001.

[24] K. S. Namjoshi. Symmetry and completeness in the analysis of parameterized systems. VMCAI, pages 299–313, 2007.

[25] C. Newcombe, T. Rath, F. Zhang, B. Munteanu, M. Brooker, and M. Deardeuff. How Amazon web services uses formal methods. *Communications of the ACM*, 58(4):66–73, 2015.

[26] A. Panda, K. J. Argyraki, M. Sagiv, M. Schapira, and S. Shenker. New directions for network verification. In *SNAPL*, volume 32 of *LIPIcs*, pages 209–220, 2015.

[27] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM*, pages 1405–1413, 1997.

[28] A. Pnueli, J. Xu, and L. D. Zuck. Liveness with (0, 1, ∞)-counter abstraction. In *CAV*, pages 107–122, 2002.

[29] J. F. Søgaard-Andersen, S. J. Garland, J. V. Guttag, N. A. Lynch, and A. Pogosyants. Computer-assisted simulation proofs. volume 697 of *LNCS*, pages 305–319. Springer, 1993.

[30] Y. J. Song and R. Renesse. Bosco: One-step byzantine asynchronous consensus. In *DISC*, pages 438–450, 2008.

[31] M. Talupur and M. R. Tuttle. Going with the flow: Parameterized verification using message flows. In *FMCAD*, pages 1–8. IEEE, 2008.

[32] B. Thallner, H. Moser, and U. Schmid. Topology control for fault-tolerant communication in wireless ad hoc networks. *Wireless Networks*, 16(2):387–404, 2010.

[33] R. Wattenhofer and A. Zollinger. XTC: A practical topology control algorithm for ad-hoc networks. In *IPDPS*, 2004.

[34] J. Welch. Challenges for formal methods in distributed computing. In *CAV*, 2013. Invited talk.

Note: References such as [M1], [R1], [B1] refer to pages 7 and 10 of the CV in Sections B1.b and B1.c.

[4]https://excape.cis.upenn.edu/ http://arise.or.at/ http://vcla.at

Section b: Curriculum Vitae

Personal Information

Born on February 5, 1971 (44 years) — Male — Austrian
Married to Dr. Anna Prianichnikova, one son Nikita (born 2001)
veith@forsyte.at — http://www.forsyte.at — +43 664 60 588 18441

Employment History

2010 –	Full Professor, TU Wien, Austria
2008-2009	Full Professor (W3), TU Darmstadt, Germany
2007	Visiting Scientist, Microsoft Research Cambridge, UK (2 months)
2003-2007	Associate Professor (C3), TU Munich, Germany
2001-2003	Associate Professor, TU Wien, Austria
1999-2000	Visiting Scholar, Carnegie Mellon University, Pittsburgh, USA
1995-2001	Teaching and Research Assistant (Universitätsassistent), TU Wien

Education

2001	Habilitation in Applied and Theoretical Computer Science, TU Wien
1998	Doctor of Technical Sciences, TU Wien (Advisor: G. Gottlob).
1994	Diplom-Ingenieur (approx. M.Sc.) in Computational Logic, TU Wien

University Leadership Activities

2011 –	Founder and co-chair, *Vienna Center for Logic and Algorithms*, TU Wien
2011 –	Vice speaker of the professors, *Academic Senate of TU Wien*
2011 –	Coordinator of Research Focus *Logic and Computation*, Faculty of Informatics, TU Wien
2011 –	Member of the *Faculty Council* of the Faculty of Informatics, TU Wien

Educational Activities

2004–	Graduated 8 PhD students and 2 habilitations, 7 PhD students and 4 habilitations ongoing
2014 –	Speaker, Doctoral College *Logical Methods in Computer Science*, cf. below
2014 –	Member, Committee on Logic Education, Association for Symbolic Logic
2011 –	Member of the Curriculum Committee, Faculty of Informatics, TU Wien
1990	Established an individual diploma curriculum in Computational Logic

Publication Record

More than 120 publications in leading conferences for *computer-aided verification* (CAV, FMCAD, TACAS, CONCUR, SAS, VMCAI), *logic in computer science* (LICS, CSL), *software engineering* (ICSE, ASE, FASE), *computer security* (Security and Privacy, CCS), *theory of computation* (ICALP, MFCS, CCC, PODC), and journals such as J.ACM, ACM TOCL, ACM TOSEM, ACM TOIT, IEEE TDSC, IEEE TSE, Information and Computation, JCSS, IPL, Annals of Pure and Applied Logic etc.

More than 5100 citations according to google scholar

Most Cited Papers Omitted in Section c

1418 cit.	[M1] E Clarke, O Grumberg, S Jha, Y Lu, H Veith. **Counterexample-guided abstraction refinement.** Computer-Aided Verification (CAV), Springer LNCS 1855, 2000
618 cit.	[M2] E Clarke, O Grumberg, S Jha, Y Lu, H Veith. **Counterexample-guided abstraction refinement for symbolic model checking.** J. ACM 50(5), 2003
182 cit.	[M3] A Campailla, S Chaki, E Clarke, S Jha, H Veith. **Efficient filtering in publish-subscribe systems using binary decision diagrams.** Int. Conf. on Software Engineering (ICSE), 2001
148 cit.	[M4] P Chauhan, E Clarke, J Kukula, S Sapra, H Veith, D Wang. **Automated abstraction refinement for model checking large state spaces using SAT based conflict analysis.** Formal Methods in Computer-Aided Design (FMCAD), 2002
135 cit.	[M5] E Clarke, S Jha, Y Lu, H Veith. **Tree-like counterexamples in model checking.** Logic in Computer Science (LICS), 2002

Funding Record (exluding current grants)

Basic research projects funded by DFG (German research fund) and WWTF (Vienna research fund).
Industrial projects funded by EADS, Diehl Aerospace, Microsoft Research as PI, and by BMW as co-PI.
co-PI in large strategic grants: doctoral college PUMA in Munich and security center CASED in Darmstadt, doctoral college *Mathematical Logic in Computer Science* and doctoral college *Adaptive Systems* in Vienna.
co-PI in two TEMPUS EU grants with Uzbekistan and Kazakhstan for e-Learning in developing countries.

Further Evidence of Recognition (excluding Section c)

2015	Nominated for CAV Award (to be decided in July 2015)
2014 –	Editor of Acta Informatica, Springer
2014 –	Editor of Modeling and Analysis of Information Systems, Russia
2010 –	Guest Editor for FMSD, LMCS, JCSS
2005 –	Adjunct Full Professor, Carnegie Mellon University, Pittsburgh, USA
2003	ACM SIGSOFT Distinguished Paper Award
2002/03	Invited speaker at Computer Science Logic (CSL) and ASL Annual Meeting
1999	Winner of a Max Kade Fellowship to visit Carnegie Mellon University
1999	Dissertation award *sub auspiciis praesidentis* by the Austrian president
1989	Winner of International Programming Award, Ukrainan Academy of Science, Kiev

Research Record: Interdisciplinary, Curiosity-Driven and Application-Driven

The use of mathematical logic in computer science was at the core of my research and education efforts since my undergraduate days. Following theoretical contributions to finite model theory, database theory, and complexity theory in my PhD thesis, I made model checking my main research area, and focused my interest in logic on applications in software engineering, hardware design, embedded systems, computer security and distributed algorithms. In the course of my career, I always attempted to combine rigor of methodology with breadth of interest. I believe that my track record demonstrates my ability to build interdisciplinary teams, to foster the creativity of my students, and to transfer expertise and knowledge across established disciplines and schools of computer science – which is crucial for the HYDRA proposal. My interdisciplinary contributions include binary decision diagrams for publish-subscribe systems (ICSE 2001), descriptive complexity theory for hardware data structures such as bit-vector logics and binary decision diagrams (Comp. Complexity 1998, MFCS 2013), the first ANSI C compiler for secure two-party computation (CCS 2012), security protocols for networked games (IEEE Security & Privacy 2007), database query languages based on algorithms for temporal logic (ACM TOCL 2002), verification and testing methods based on query answering (ICALP 2004, ACM TOCL 2010, ASE 2010, CAV 2008), white box verification for embedded systems code with intellectual property concerns (CAV 2007, ACM TOSEM), description logic for shape analysis (LICS 2015, IFM 2014), machine learning for verification (CAV 2015), and model checking for malware analysis (IEEE TDSC 2010, FMCAD 2010). The HYDRA project will summarize my research experience in an exciting and important high impact challenge.

Scientific Leadership: Creation of a Logic and Verification Hotspot in Austria

When I returned to my native Vienna from faculty positions abroad in 2010, the traditional strength of TU Wien in logic for databases, artificial intelligence and automated deduction was being complemented by a series of strong hires in computer-aided verification all over Austria. Recognizing the enormous potential of this situation, I took a leading role in joint efforts to develop Austria as a research hotspot for logic in CS:

The **National Research Network Rigorous Systems Engineering** (RiSE – www.arise.or.at) is an expedition grant 2011-2018 with a total budget of 7.4 M EUR aimed at computer-aided verification and synthesis. As the vice speaker, I am sharing responsibilities with speaker Roderick Bloem (Graz). Further PIs include T. Henzinger (ERC Advanced), A. Biere, K. Chatterjee (ERC Starting), L. Kovacs (ERC Starting), and U. Schmid, whose participation resulted in the breakthrough papers described in Section a. In 4 years, RiSE resulted in 200 visitors to Vienna, dozens of joint papers, and a vast extension of faculty (R. Grosu and L. Kovacs as senior hires, E. Bartocci, M. Seidl, A. Sokolova, G. Weissenbacher, F. Zuleger as tenure track faculty).

The **Doctoral College Logical Methods in Computer Science** (LogiCS — www.logic-cs.at) is extending this collaboration to the logic in CS community in Austria. With a budget of 2.7 M EUR for 2014-2018 and a perspective until 2022, LogiCS provides funding and a graduate program for up to 30 PhD students in databases, verification, and logic. LogiCS has significantly improved our attractiveness for international graduate students (and attracted 50% female students). LogiCS is coordinated by myself and Stefan Szeider as vice speaker.

The **Vienna Summer of Logic** 2014 (VSL - www.vsl2014.at) was the largest conference on logic in computer science in history, including FLOC, the Logic Colloquium, and KR. Chaired and initiated by M. Baaz, T. Eiter and myself, VSL helped to position Vienna as a research hotspot and attract faculty and students.

The **Vienna Center for Logic and Algorithms** (VCLA - www.vcla.at), founded by Stefan Szeider and myself is supporting workshops, visitors, and outreach programs. The activities of VSL and VCLA have resulted in extensive media coverage – more than 80 news pieces in newspapers and public radio and TV – that helped to improve our profile vis-a-vis the public, colleagues, and the government.

Appendix: All ongoing and submitted grants and funding of the PI (Funding ID)

On-going Grants

Project Title	*Funding source*	*Amount (Euros)*	*Period*	*Role of the PI*	*Relation to current ERC proposal*
RiSE "Rigorous Systems Engineering" Currently in the 2nd (final) 4 year funding phase	FWF Austrian Science Foundation	7.4 M EUR 1207 K EUR for our group	1.3.2011-31.3.2019	vice speaker	RiSE is an Austrian wide research network. The RiSE research topics are disjoint to the ERC proposal, but have synergies concerning synthesis, real time, and probabilistic systems.
LogiCS "Logical Methods in Computer Science" 1st Funding Phase	FWF Austrian Science Foundation	2.7 M EUR approx.. 500 K EUR for our group	1.3.2014-31.3.2018	speaker (= main PI)	LogiCS is a doctoral college with funding for PhD students. LogiCS will fund 2 PhD students who will contribute to HYDRA.

Applications

Project Title	*Funding source*	*Amount (Euros)*	*Period*	*Role of the PI*	*Relation to current ERC proposal*
APALACHE "Abstraction-based parameterized TLA+ Checker"	WWTF Vienna Science and Technology Fund	556 K EUR	2016-2018 (36 months)	Core team member	Igor Konnov is the PI of the project. The project is concerned with predicate abstraction and parameterized verification of TLA+ models. If the application is successful, it will increase the impact of HYDRA on the TLA+ community.
EduProf "A transnational framework for digital and professional capacity building in Uzbekistan."	EU Erasmus+ Structural Projects	974 K EUR 122 K EUR for TU Wien	2016-2018 (36 months)	Consortium member	No relation

Section c: Ten Years Track Record

10 Representative Papers

CAV 2015	[R2] **Empirical Software Metrics for Benchmarking of Verification Tools.** Y Demyanova, T Pani, H Veith, F Zuleger.
CAV 2015	[R3] **SMT and POR beat Counter Abstraction: Parameterized Model Checking of Threshold-Based Distributed Algorithms.** I Konnov, H Veith and J Widder.
LICS 2015	[R1] **Extending ALCQIO with Reachability: Between Finite Model Theory and Description Logic.** T Kotek, M Simkus, H Veith, F Zuleger.
525 cit.	[R4] **Modular verification of software components in C.** S Chaki, E Clarke, A Groce, S Jha, H Veith. IEEE Trans. SE, 30(6):388–402, 2004.
110 cit.	[R5] **Detecting malicious code by model checking.** J Kinder, S Katzenbeisser, C Schallhart, H Veith. Proc. DIMVA, 174-187, 2005
89 cit.	[R6] **Environment abstraction for parameterized verification.** E. Clarke, M. Talupur, H. Veith. Proc. VMCAI, 126-141, 2006.
76 cit.	[R7] **An abstract interpretation-based framework for control flow reconstruction from binaries.** J Kinder, F Zuleger, H Veith. Proc. VMCAI, 214-228, 2009
48 cit.	[R8] **Verification by Network Decomposition.** E Clarke, M Talupur, T Touili, J Veith. Proc. CONCUR, 276-291, 2004
45 cit.	[R9] **Bound analysis of imperative programs with the size-change abstraction.** F Zuleger, S Gulwani, M Sinn, H Veith. Proc. SAS, 280-297, 2011
28 cit.	[R10] **Secure Two-party Computations in ANSI C.** A. Holzer, M. Franz, S. Katzenbeisser, H. Veith. Proc. ACM CCS, 772–783, 2012

[R7] was the most cited, [R5] the 2nd most, [R6], [R9] the 3rd most cited papers of their conferences. [R4] presents the first generation SW checker MAGIC. [R6], [R8] have become classic papers on parameterized verification. [R5] and [R7] represent our project on analysis of executables and malware detection, [R9] our research thread on loop bounds, and [R10] our recent ANSI-C compiler for secure two-party computation. [R1], [R2], [R3] demonstrate the agility of our current group: [R3] uses advanced methods from finite model theory, [R2] machine learning for the analysis of a tool competition, and [R1] is the last paper in the series leading to HYDRA.

Research Monographs

2016	[B1] **Model Checking.** E Clarke, O Grumberg, D Kroening, D Peled, H Veith. MIT Press. *2nd revised edition of the classic monograph on model checking.*
2015	[B2] **Handbook of Model Checking.** Edited by E Clarke, T Henzinger , H Veith. Springer. *With 32 chapters, the handbook will represent the state of the art in model checking.*
2015	[B3] **Decidability of Parameterized Verification.** R Bloem, S Jacobs, A Khalimov, I Konnov, S Rubin, H Veith, J Widder. Synthesis Lectures on Distributed Computing Theory, Morgan & Claypool Publishers.
94 cit.	[B4] **25 Years of Model Checking - History, Achievements, Perspectives.** Edited by O Grumberg, H Veith. Springer LNCS Vol 5000, 2008
2008-2013	[B5-7] Proceedings Co-Editor for **CAV 2013, CSL 2010, LPAR 2008**.

Invited Presentations (Keynote KN, Invited Speaker IS, Invited Tutorialist IT)

Sep 2015	**Verification of Distributed Algorithms.** IS, Parameterized Verification Workshop, Madrid.
Aug 2015	**Who is Afraid of Verifying Distributed Algorithms?** KN, PSI: 10th Ershov Informatics Conference, Kazan, Russia.
Apr 2015	**Perspectives on White-Box Testing: Coverage, Concurrency, and Concolic Execution.** KN, IEEE Int. Conference on Software Testing, Verification and Validation, Graz.
Sep 2014	**Shape and Content - A Database-Theoretic Perspective on the Analysis of Data Structures.** Joint IN, 11th Int. Conf. on Integrated Formal Methods (IFM) and 11th Int. Symp. on Formal Aspects of Component Software (FACS). Bertinoro.
Oct 2014	**Model Checking of Fault-Tolerant Distributed Algorithms.** IT, 7th Summer School VTSE (Verification Technology, Systems & Applications), Luxembourg.
Sep 2010	**(How) Did You Specify Your Test Suite?** IS, Haifa Verification Conference.
Sep 2009	**Embedding Formal Methods into Systems Engineering.** IS, 11th Int. Symp. on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara.
Sep 2008	**From Manual Proofs to Model Checking.** IT, Dagstuhl Workshop Fault-Tolerant Distributed Algorithms on VLSI Chips

Aug 2008	**Danger Inside: Computer Errors in the Infrastucture.** KN, European Forum Alpbach.
Oct 2007	**Ptolemaic Software Analysis.** Invited Speaker. IS, Symposium on Software Quality - State of the Art, ETH Zürich.
Oct 2007	**On the Notion of Vacuous Truth.** IS, Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR), Erevan.

Research expeditions that the applicant Principal Investigator has led (Details in Section b)

2011 - 2018	National Research Network **RiSE**, deputy speaker. (7.4 Mio EUR)
2014 –	Doctoral College **LogiCS**, speaker. (2.7 Mio EUR for 2014-2018)
2012 – 2018	Vienna Center for Logic and Algorithms **VCLA**, co-chair (600 K EUR)

Organisation of international conferences in the field of the applicant

FMCAD'16	PC Co-Chair. (*International Conference in Computer-Aided Verification*)
VSL'14	Vice chair of the Vienna Summer of Logic 2014, cf. Section b
FLOC'14	Chair of FLOC (Federated Logic Conference) 2014. *Included CAV, CSF, CSL-LICS, ICLP, IJCAR, ITP, RTA-TLCA, SAT + 70 workshops. Part of Vienna Summer of Logic.*
CAV'13	PC Co-Chair, St. Petersburg. (*International Top conference in Computer-Aided Verification*)
CSL'10	PC Co-Chair, Brno. (*European Top Conference for Logic in Computer Science*)
FMCAD'10	Tutorial Chair, Lugano.
LPAR'08	PC Co-Chair, Qatar. (*International Conference in Logic in Computer Science*)
PCs	CAV (5 times), FMCAD (5 times), CSL (3 times), Haifa Verification Conference (3 times), LICS (2 times), Computer Science Russia (4 times) and more than 20 other conferences.
Steering Committees	EACSL (Europ. Assoc. CS Logic), Kurt Gödel Society, Workshop Series EC^2 (Exploiting Concurrency Efficiently and Correctly), Austrian Computer Science Day

Major contributions to the early careers of excellent researchers

S Katzenbeisser	PhD 2004, Philips Research, now Professor (W3) at TU Darmstadt, Germany. (7 graduated PhD students, one of them a faculty member at Twente). 149 pub.
M Samer	PhD 2004, Postdoc Leicester, † 2010. *Zemanek Award* (Best Austrian PhD Thesis). 27 pub.
C Schallhart	PhD 2007, Postdoc Oxford, now Google London. *Kurt Gödel Scholarship* to visit Columbia University. *Award for best Austrian Master Thesis.* 59 pub.
J Kinder	PhD 2011, Postdoc EPFL, now Lecturer at Royal Holloway, UK. *General Electric Silver Award* for Master Thesis, *Internship at MSR*. 18 pub.
M Tautschnig	PhD 2011, Postdoc Oxford, now Lecturer at Queen Mary, UK. 41 pub.
F Zuleger	PhD 2011, now Tenure Track Assistant Professor, TU Wien. *Microsoft European PhD Scholarship*, *Internship at MSR*, *TopMath PhD program* for gifted students. 20 pub.
S Kugele	PhD 2012, now Postdoc TU München. 13 pub.
A Holzer	PhD 2014, now Postdoc at Univ. Toronto. 24 pub.
J Birgmeier	Master 2014, now PhD student at Stanford. 1 pub.
D Pötzl	Master 2013, now PhD student at Oxford. 4 pub.
T Pani	Master 2014, now PhD student in my group. *EPILOG Master Thesis Award.* 1 pub.
S Rubin	Postdoc 2012-14, now Postdoc at Naples, Italy. 27 pub.
G Kovasznai	Postdoc 2013-14, now Associate professor at Eszterházy Károly College, HU. 9 pub.
J Widder	Postdoc 2011-14, now Assistant professor at TU Wien. 38 pub.
Current students	5 female and 2 male PhD students: Annu Gmeiner (IN, graduation 2015), Yulia Demyanova (RU, graduation 2015), Marijana Lazic (SRB, new student), Nadia Labai (ISR, new student), Thomas Pani (AT, new student), Soodeh Farokhi (IR, graduation 2015), Moritz Sinn (DE, graduation 2015).

The following further young researchers will confirm major impact on their career

Ivona Brandic (TU Wien Science Award, 500 K EUR, now Tenure Track at TU Wien), Sagar Chaki (CMU, now CMU Software Engineering Institute) Agata Ciabattoni (FWF START grant, 1 M EUR, now Professor at TU Wien), Azadeh Farzan (Toronto, 1 year sabbatical in Wien), Laura Kovacs (ERC Starting Grant, 1.5 M EUR, Professor at TU Wien in my group from 2016), Magdalena Ortiz (now Tenure Track at TU Wien), Muralidhar Talupur (CMU, then Intel), Georg Weissenbacher (WWTF Young Researcher Award, 1.5 M EUR, now Tenure Track at TU Wien).

All citation counts for this proposal were obtained from google.scholar *and* publish or perish