DOSSIER DE CANDIDATURE
APPLICATION

Cochez le concours sur lequel vous candidatez
Check the position on which you are applying

☐ **CR2 - (Chargés de recherche de deuxième classe / *Young graduate scientist position*)**

☒ **CR1 - (Chargés de recherche de première classe / *Young experienced scientist position*)**

☐ **DR2 - (Directeurs de recherche de deuxième classe / *Senior researcher position*)**

Nom[1] **:** KONNOV
Last name

Prénom : Igor
First name

Sexe : ☐**F** ☒**M**
Sex

Nom utilisé pour vos publications (facultatif) :
Name used for your publications (optional): **Igor Konnov, Igor V. Konnov, I.V. Konnov**

[1] Il s'agit du nom usuel figurant sur vos pièces d'identité
It is the name appearing on your identity cards

SOMMAIRE / *SUMMARY*

Formulaire 1 — PARCOURS PROFESSIONNEL
Form 1 — PROFESSIONAL HISTORY

1) Parcours Professionnel / *Professionnal history*

1. Situation professionnelle actuelle / *Current professionnal status*

Statut et fonction / *Position and statute*: Postdoctoral researcher, principal investigator in the WWTF project "APALACHE"
Établissement (ville - pays) / *Institution (city -country)*: TU Wien (Vienna — Austria)
Date d'entrée en fonction / *Start*: 01.01.2016 as a PI (at TU Wien since 01.07.2011)
[ ] Sans emploi / *Without employment*

2. Expériences professionnelles antérieures / *Previous professionnal experiences*

Date début *Start*	Date fin *End*	Établissement *Institutions*	Fonction et statut *Positions and status*
12.12.2011	31.12.2015	TU Wien (Vienna – Austria)	Assistant professor (limited contract) (Universitätsassistent)
01.07.2011	11.12.2011	TU Wien (Vienna – Austria)	Postdoctoral researcher (Projektassistent)
01.01.2010	30.06.2011	Lomonosov Moscow State Univ. (Moscow — Russia)	Junior research fellow (m.n.s.)
01.12.2006	31.12.2009	Lomonosov Moscow State Univ. (Moscow — Russia)	Pre- and postdoctoral research and teaching assistant (officially hired as a "programmer")
2006	2010	Sytech LLC (Moscow — Russia)	part-time systems architect
2004	2006	Sytech LLC (Moscow — Russia)	part-time software developer
2002	2003	IFirst LLC (Moscow — Russia)	part-time programmer

Nombre d'années d'exercice des métiers de la recherche après la thèse / *Number of years of professionnal research experience after thesis*: 8 (eight years, PhD awarded on February, 13, 2009, defended on November, 28, 2008)

2) Interruptions de carrière/*Career breaks*

None

3) Prix et distinctions / *Prizes and awards*

1. **2009–2010**: Fellowship for young researchers. Faculty of Computational Mathematics and Cybernetics, the Moscow State University — Russia
2. **2003–2005**: PhD student scholarship by LSI Logic (Russian branch) — Russia
3. **2003** Graduated from the Moscow State University *with distinction*: — Russia
   97% of the best score: avg. score 4.87 (Russian scale, 5.0 is the best score)

4) Encadrement d'activités de recherche / *Supervision of research activities*

PhD students:

- **Nov 2016–now**. Jure Kukovec (TU Wien) is working on a new symbolic model checker for TLA$^+$. Main advisor.
- **Oct 2016–now**. Thanh Hai Tran (TU Wien) is working on a new symbolic model checker for TLA$^+$. Main advisor.
- **Feb 2015–now**. Marijana Lazić (TU Wien). I am co-advising Marijana together with Dr. Josef Widder on her thesis "Logic Modelling and Analysis of Distributed Algorithms". We have written a conference paper for POPL'17 [4] together and submitted a journal paper to FMSD [37].
- **2011–2014**. Annu Gmeiner (nee John, TU Wien). I advised Annu informally. The formal advisors were Prof. Helmut Veith and Dr. Josef Widder. Dr. Gmeiner defended her PhD thesis on "Parameterized model checking of fault-tolerant distributed algorithms" in 2015. We have written the conference papers for SPIN'13 [11], FMCAD'13 [10], PODC'13 [19], and the tutorial paper for SFM [13] together.

Internships:

- **Sep 2015–Nov 2015**. Qiang Wang (PhD student of Prof. Joseph Sifakis, EPFL, Switzerland). This internship resulted in a paper accepted at CONCUR'16 [6].

Master students:

- **2016**. Jure Kukovec (University of Ljubljana). Together with Prof. Andrej Bauer (Univ. Ljubljana), I co-advised Jure on his master thesis on "Extensions of Threshold Automata for Reachability in Parameterized Systems".
- **2016**. Thanh Hai Tran (TU Wien). I advised Thanh Hai Tran on his master thesis on "User-guided predicate abstraction of TLA$^+$ specifications".
- **2011**. Alexander Mischenko (Moscow State Univ.). I advised Alexander Mischenko on his specialist thesis (approx. MSc) on "Static Type Analysis of Python Programs on Bytecode Level".
- **2009**. Denis Sigaev (Moscow State Univ.). Together with Alexei Kachalin, I advised Denis Sigaev on his specialist thesis on "Detection of Programs Protected from Reverse Engineering".
- **2008**. Alexey Schevchenko (Moscow State Univ.). I advised Alexey Schevchenko on his specialist thesis on "Application of Regular Model Checking to Infinite-State Systems".
- **2007**. Peter Bulychev (Moscow State Univ.). Together with Prof. Vladimir Zakharov, I advised Peter Bulychev on his specialist thesis on "Game-Theoretic Methods of Protocol Verification".

5) Responsabilités collectives / *Responsibilities*

- **Workshop chair** of the Conference on Computer-Aided Verification (**CAV 2013**).
- **Co-organizer** of four workshops on Formal Reasoning in Distributed Algorithms (**FRIDA**):
  FRIDA'14 at CAV'14 in Vienna, FRIDA'15 at FORTE'15 in Grenoble, FRIDA'16 at NETYS'16 in Marrakech, and FRIDA'17 planned at DISC'17 in Vienna.
- **Guest editor** (with Helmut Veith and Natasha Sharygina) of the Special issue on Computer-Aided Verification 2013 in Formal Methods of System Design (**FMSD, Springer**).
- **Member of Program Committees**: Formal Methods in Computer-Aided Design (**FMCAD**) 2017; External Reviewer Committee member of Computer-Aided Verification (**CAV**) 2016; Symbolic and Numeric Algorithms for Scientific Computing (**SYNASC**) 2013 and 2016; Stabilization, Safety, and Security of Distributed Systems (**SSS**) 2015; Tools and Methods of Program Analysis (**TMPA**) 2015 and 2017; 8th Workshop on Program Semantics, Specification, and Verification (**PSSV**); Parallel, Distributed, and Network-based Processing (**PDP**) 2017.
- **External reviewer**: TACAS'17, STACS'17, VMCAI'17, MARS'17, ICFEM'16, CONCUR'16, IJCAR'16, LICS'16, EuroPar'16, AAMAS'16, CAV'15, FMCAD'15, TACAS'15, FoSSaCS'15, CAV'14, SAS'14, GandALF'14, ESOP'14, HVC'14, CAV'13, LATA'13, SSS'13, CAV'12, NFM'12, SPIN'12, VMCAI'12, FMICS'11, CSL'11.
- **Journal and book chapter reviews**: TIME in 2015, LMCS in 2017 (Logical Methods in Computer Science), Handbook of Model Checking (eds. E. Clarke, T. Henzinger, H. Veith).
- **Editorial Board** of Proceedings of the Institute for System Programming of the Russian Academy of Sciences (Moscow, Russia), member since 2016.
- **Student Award Committee** of Vienna Center for Logic and Algorithms 2014-2015.

6) Management / *Management*

Research projects:

- **2016–2018**. I am running the research project ICT15-103 "APALACHE: Abstraction-based Parameterized TLA Checker" at TU Wien, which currently involves three PhD students. The project was granted in 2015 by Vienna Science and Technology Fund (WWTF) after a competitive international evaluation: 10 out of 137 proposals were selected. The project was awarded 539,000 € for three years. I am the principal investigator in the project, and Josef Widder is the co-principal investigator.

- **2010-2011**. I coordinated the project 14.740.11.0399 at Moscow State University on "Developing a Prototype for Computer Simulation of Real-Time Distributed Systems" (PI: Prof. Smeliansky), which was supported by the Russian Federal Special-Purpose Programme. The project budget was approximately 200,000 €. In this project, I coordinated two teams: (1) the team developing a distributed simulation engine based on High-Level Architecture (HLA); and (2) the team integrating verification tools. These two teams included one teaching & research assistant, two PhD students, and three master students. I was responsible for coordination, research agenda, and report writing.

Research & development teams:

- **2009–2010**. R&D project "Obfuscation techniques on intermediate code representation" at Moscow State University. I led the team of one master student and one PhD student.
- **2007–2008**. R&D project "Obfuscation techniques for C++" at Moscow State University. I led the team of one master student and one PhD student.

7) Collaborations, mobilité / *Collaborations, visits*

1. Collaborations

(Sorted according to my understanding of intensivity of the collaboration)

- Helmut Veith (Formal Methods in Systems Engineering, TU Wien, Austria). Since 2011, I have been working as a postdoc at Helmut Veith's group. Together, we investigated techniques for verification of fault-tolerant distributed algorithms and parameterized model checking. Our joint work resulted in a number of high-quality publications [4, 1, 2, 6, 7, 12, 8, 13, 9, 19, 10, 11]. Together with Helmut Veith and Natasha Sharygina, I organized the Conference on Computer-Aided Verification (CAV 2013) in St. Petersburg, Russia.
- Josef Widder (Embedded Computing Systems Group, TU Wien, Austria). We started our collaboration in 2011, when Josef Widder joined Helmut Veith's group. Since then we have been working together on verification of fault-tolerant distributed algorithms. Josef Widder contributed his invaluable knowledge on distributed computing and his rigorous approach to mathematical proofs. Our joint work resulted in a number of high-quality publications [4, 1, 5, 37, 2, 7, 12, 8, 13, 9, 19, 10, 11]. After the tragic death of Helmut Veith in March 2016, we continued our joint research efforts, which resulted in recent publications [4, 5, 37]. We are collaborating in the project Apalache, in which we are developing a new model checker for TLA$^+$.
- Vladimir Zakharov (Moscow State University, Russia). Vladimir Zakharov was my main PhD advisor, and we wrote a number of papers on parameterized model checking [3, 28, 41, 42, 33, 35, 36] together. After I obtained PhD, we worked on symmetry reduction [32, 25] and model checking of UML statecharts [29, 30, 18] together.
- Ulrich Schmid (Embedded Computing Systems Group, TU Wien, Austria). Ulrich Schmid contributed his expertise in distributed computing in our early work on verification of fault-tolerant distributed algorithms. Together with Ulrich Schmid, Helmut Veith, Annu Gmeiner, and Josef Widder, we have published papers on our early techniques for model checking of threshold-guarded algorithms [13, 19, 10, 11].
- Roderick Bloem (TU Graz, Austria). For several years, we collaborated on the survey of parameterized model checking techniques. We published this survey as a book [12] in 2015 and an overview article [2] in 2016. Currently, we are working on synthesis of fault-tolerant distributed algorithms.
- Stephan Merz (VERIDIS, INRIA Nancy, France). We have established a collaboration in the project Apalache. Stephan Merz has been providing us with his expertise on TLA$^+$ and the TLA$^+$ toolset. Additionally, we have been co-organizing four workshops on Formal Reasoning in Distributed Algorithms.
- Joseph Sifakis (Verimag, France). In 2015, we started joint work on parameterized extensions of the Behavior-Interaction-Priority framework, which resulted in the paper at CONCUR'16 [6].
- Swen Jacobs (Reactive Systems Group, Universität des Saarlandes, Germany). We have been working on the survey of parameterized model checking techniques [2, 12]. Currently, Swen Jacobs, Andrey Kupriyanov (IST Austria), and I are collaborating on parameterized verification of leader election algorithms.
- Francesco Spegni (Univ. Politecnica Delle Marche, Italy). We have been working on abstractions of fault-tolerant distributed algorithms with message delays. Recently, we have published a paper at VMCAI'17 [5]. Francesco Spegni has extended my tool ByMC (cf. Form 3) with abstraction techniques for timed automata. Currently, we are preparing another submission.
- Byron Cook (Amazon, USA). We have established a collaboration with Byron Cook in the project Apalache. As Amazon is using TLA$^+$, this collaboration will help us to get an early feedback on our new model checker for TLA$^+$.

2. Short visits

- Thomas Wahl and Daniel Kroening. Computing Laboratory. Oxford/UK. One week in February 2011.
- Alexander Letichevsky. Glushkov Institute of Cybernetics. Kiev/Ukraine. A research visit in January–February 2009 as part of the EU INTAS project Nr. 05-1000008-8144.

8) Enseignement / *Teaching*

Vienna University of Technology (TU Wien)

- **2013–present.** Computer Aided Verification (Prof. Helmut Veith)
  Master students, compulsory, lectures & practicals, 3 ECTS
  Since 2017, I have been holding the course. Until 2017, reading parts of the lecture course, teaching assistance.
- **2017–present.** Lab Exercises in Computer Aided Verification Master students, compulsory, lab exercises, 3 ECTS
  Since 2017, I have been holding a lab course complementary to CAV, where the students implement a basic model checking technique, or verify a distributed algorithm using standard tools such as Spin, NuSMV, or TLC.
- **2013–present.** Program and Systems Verification (Assoc. Prof. Georg Weissenbacher)
  Teaching assistance Bachelor students, compulsory, lectures & practicals, 6 ECTS
- **2016–2017.** Introduction to Logical Methods in Computer Science PhD students of Logic in CS, 2h, 3 ECTS
  Each member of the Doctoral College LogiCS reads one introductory lecture. I read the lecture on Model Checking.
- **2011–2015.** Formal Methods of Informatics (Prof. Helmut Veith et al.)
  Teaching assistance Master students, compulsory, lectures & practicals, 6 ECTS

Moscow State University (MSU)

- **2008–2010.** Software model checking (Dr. Savenkov) 8th semester, compulsory, lectures & seminars, 32 hrs.
  Designed the course together with K. Savenkov, read parts of the lecture course, teaching assistance
- **2004.** Seminars on The C Programming Language and UNIX 3rd semester, compulsory, 32 hrs.
  Instructed at all seminars (approx. 20 students)
- **2005.** Seminars on Syntax Analysis and C++ 4th semester, compulsory, 32 hrs.
  Instructed at all seminars (approx. 20 students)
- **2004.** Operating Systems (Prof. Terekhov) 3rd semester, compulsory, lectures, 54 hrs.
  Teaching assistance
- **2003–2011.** Computer Networks (Prof. Smeliansky) 6th semester, compulsory, lectures, 64 hrs.
  Teaching assistance
- **2003–2004.** The Java Programming Language optional, lectures, 32 hrs.
  Read parts of the lecture course, teaching assistance
- **2003–2004.** MSU *entrance examinations* in mathematics compulsory
  Corrected written math exams, assisted in the oral math exams

Kazakhstan branch of Moscow State Univ., Astana/Kazakhstan

- **2011.** Software model checking 8th semester, compulsory, lectures & seminars, 32 hrs.
  Held the lecture course and the seminars

Tashkent University, Tashkent/Uzbekistan

- **2011–2013.** Participated in the EU project CANDI: Teaching Competency and Infrastructure for e-Learning and Retraining

9) Diffusion de l'information scientifique / *Dissemination of scientific knowledge*

10) Eléments divers / *Other relevant information*

Tutorials

[T1] I. Konnov. *Model Checking of Fault-tolerant Distributed Algorithms*. Tutorial at the Spring School Logic and Verification, Vienna, April. 2016. URL: http://forsyte.at/events/love2016/.

[T2] H. Veith and I. Konnov. *Model Checking of Fault-tolerant Distributed Algorithms*. Tutorial at the Summer School on Verification Technology, Systems & Applications, Luxembourg, Luxembourg, October. 2014. URL: http://resources.mpi-inf.mpg.de/departments/rg1/conferences/vtsa14/.

Invited speaker at conferences and workshops

[T3] I. Konnov. *Model Checking of Threshold-based Fault-Tolerant Distributed Algorithms*. Invited talk at the 7th Workshop on Program Semantics, Specification & Verification, St. Petersburg, Russia, June. 2016. URL: http://pssv-conf.ru/en/2016/program.

[T4] I. Konnov. *Parametrized Model Checking of Fault-tolerant Distributed Algorithms by Abstraction*. Tutorial at the International Conference Tools and Methods of Program Analysis, Kostroma, Russia, November. 2014. URL: http://tmpaconf.org/pasteventsmaterialsen/keynote-speakersen#2014.

Invited Seminar Talks

[T5] I. Konnov. *Model Checking of Fault-tolerant Distributed Algorithms: Safety and Liveness*. Invited talk at the Seminar of Rigorous System Design Laboratory, Lausanne, Switzerland, September. 2016.

[T6] I. Konnov. *Model Checking of Threshold-based Fault-tolerant Distributed Algorithms*. Invited talk at the Seminar on Foundations of Mathematics and Theoretical Computer Science, Ljubljana University, Ljubljana, Slovenia, May. 2016.

[T7] I. Konnov. *Model Checking of Threshold-Guarded Distributed Algorithms: Beyond Reachability*. Invited talk at the Vericlub Seminar (Bertrand Meyer), Toulouse, France, November. 2016.

[T8] I. Konnov. *Model Checking of Threshold-based Fault-tolerant Distributed Algorithms*. Invited talk at Amazon, Herndon, VA, USA, June. 2015.

[T9] I. Konnov. *Model checking of threshold-based fault-tolerant distributed algorithms*. Talk at the Dagstuhl Seminar on Distributed Cloud Computing, Dagstuhl, Germany, February. 2015.

[T10] I. Konnov. *SMT and POR beat Counter Abstraction*. Invited talk at the RiSE Seminar at Institute of Science and Technology Austria, Klosterneuburg, Austria, April. 2015.

[T11] I. Konnov. *On Completeness of Bounded Model Checking for Threshold-based Distributed Algorithms: Reachability*. Talk at the Seminar on Theoretical Problems in Programming, Moscow State University, Moscow, Russia, February. 2014.

[T12] I. Konnov. *Counter Attack on Byzantine Generals*. Talk at the Dagstuhl Seminar on Formal Verification of Distributed Algorithms, Dagstuhl, Germany, April. 2013.

[T13] I. Konnov. *Counter Attack on Byzantine Generals*. Talk at the Seminar on Theoretical Problems in Programming, Moscow State University, Moscow, Russia, February. 2013.

[T14] I. Konnov. *Who is Afraid of Model Checking Distributed Algorithms*. Talk at the PUMA/RiSE Seminar, Goldegg, Austria, September. 2012.

[T15] I. Konnov. *An invariant-based approach to the verification of asynchronous parameterized networks*. Talk at the Concurrency Seminar, Computing Laboratory, Oxford University, Oxford, UK, February. 2011.

[T16] I. Konnov. *Two Techniques of Parameterized Model Checking and Symmetry Reduction*. Talk at the RiSE Seminar, TU Vienna, Vienna, Austria, April. 2011.

[T17] I. V. Konnov. *CheAPS: Parameterized Model Checking Tool*. Joint Workshop of Microsoft Research and Institute for System Programming Russian Academy of Sciences, Moscow, June 2009. 2009.

Formulaire 2 — DESCRIPTION SYNTHÉTIQUE DE L'ACTIVITÉ ANTÉRIEURE
Form 2 — SUMMARY OF YOUR PAST ACTIVITY

Nom / *Last name*: KONNOV
Prénom/*First name*: Igor

Failures of distributed systems sometimes have drastic effects. Classical examples are networked embedded systems in flight control and automotive industry. Recent examples are cloud systems, which contain thousands of servers built of fault-prone commodity hardware. Reasoning about distributed systems of such scale is inherently hard, and human intuition is often defeated by the complexity of the task. A rigorous approach to verification of distributed systems has to address the following *research questions*: **(A)** How to verify a distributed system of real scale, e.g., with thousand components; **(B)** How to verify a distributed system designed for a faulty environment; and **(C)** How to verify a distributed system design written in an expressive specification language such as TLA$^+$, rather than in a toy modeling language?

Questions (A)–(C) are notoriously hard and are amplified by ever-growing complexity of software. Model checking is a pragmatic and successful approach to automatic verification. State-of-the-art model checkers are used in industry to find bugs (or prove absence thereof) in hardware, network protocols, and device drivers. So far, model checking worked best for sequential programs, e.g., C code without threads, and synchronous systems, e.g., hardware. To make model checking amenable for distributed systems — in both, the embedded and the cloud computing worlds — we have to address the following challenges: (1) *concurrency and non-determinism*, (2) *parameterization in the number of components*, and (3) *faults*. Since the inception of model checking in the 1980es, challenge (1) was considered to be the most urgent in the field, as concurrent executions and non-deterministic choices result in combinatorial explosion, and thus complicate application of model checkers in practice. In my research, I address Questions (A)–(C) and focus on Challenges (2)–(3).

Parameterized model checking (PMC). As raised in Question (A), one often has to verify systems where the number of replicated components is not fixed a priori, but is a parameter. This problem was the main topic of my PhD thesis [14], in which I investigated PMC for network protocols for standard topologies such as: rings, stars, and trees. I extended the framework of network invariants, which uses the simulation relation to reduce PMC to model checking of finitely many networks derived from a network grammar. To make this framework amenable to asynchronous systems, I introduced new simulation relations that are insensitive to stuttering [31, 3, 28, 42, 33, 36]; the results are summarized in [21, 3]. I implemented this approach in my tool CheAPS[1] [3, 27].

During my postdoc at TU Wien, I worked on a survey on parameterized model checking in the team of researchers from TU Graz and TU Wien. We published the survey as the book [12] and summarized the results in the ACM SIGACT News [2]. Using these results, in our collaboration with researchers from EPFL, we extended boolean interaction logic to first-order logic and formalized several network architectures [6]. The details on this research can be found in **Form 3:Fiche 3**.

Model checking of threshold-guarded fault-tolerant distributed algorithms. In the last five years, my research focused on Question (B), namely, parameterized verification of fault-tolerant distributed algorithms. These contributions are *my best research*. As the standard model checkers cannot verify fault-tolerant distributed algorithms from the distributed computing literature, we had to address two challenges: (i) develop a theory for parameterized verification of fault-tolerant distributed algorithms, and (ii) develop a tool for practical verification of the algorithms found in the literature.

Regarding theory, we were the first to introduce techniques for model checking of threshold-guarded algorithms such as reliable broadcast, non-blocking atomic commit, and one-step consensus. Prominently, our verification results have two features crucial for the algorithm designers: we verified both safety and liveness, and our results hold for all numbers of processes and faults. Our results were presented at POPL'17 [4], Information & Computation [1], CAV'15 [8], CONCUR'14 [9], FMCAD'13 [10], PODC'13 [19], VMCAI'17 [5], and SPIN'13 [11].

Concerning practical verification, I started development of Byzantine Model Checker[2] in 2012. ByMC implements our techniques: non-parameterized verification with Spin [11]; data and counter abstraction techniques that use Spin and NuSMV as backend model checkers [1, 9, 10]; our SMT-based bounded model checking for safety and liveness [4, 8]. ByMC and our experiments on verification of threshold-guarded algorithms successfully underwent artifact evaluations at CAV'15 [8] and POPL'17 [4]. The details can be found in **Form 3:Fiche 1**.

Model checking of TLA$^+$ specifications. In recognition of the breakthroughs on parameterized model checking of fault-tolerant distributed algorithms, I was awarded a grant by Vienna Science and Technology Fund. In 2016, I started the project "APALACHE: Abstraction-based Parameterized TLA Checker"[3] to develop a new symbolic model checker for TLA$^+$, which addresses Questions (A)–(C). As TLA$^+$ is designed to be concise and expressive, it challenges automatic verification methods. We focus on TLA$^+$ specifications of fault-tolerant distributed systems, and thus use specifics of these systems. As the project started recently, I discuss it only briefly in **Form 3:Fiche 2**.

[1] CheAPS: Checker of Asynchronous Parameterized Systems [http://lvk.cs.msu.su/~konnov/cheaps/index_en.html].
[2] ByMC: Byzantine Model Checker [http://forsyte.at/software/bymc/]. Source code [https://github.com/konnov/bymc].
[3] APALACHE: Abstraction-based Parameterized TLA$^+$ Checker [http://forsyte.at/research/apalache/].

Formulaire 3 — CONTRIBUTIONS MAJEURES
Form 3 — MAJOR CONTRIBUTIONS

Nom / *Last name*: KONNOV
Prénom/*First name*: Igor

Fiche 1 : Model checking of threshold-guarded fault-tolerant distributed algorithms

1. Description de la contribution / *Description of the contribution*

Together with experts in distributed computing, we started a joint effort on verification of fault-tolerant distributed algorithms (FTDAs) and we were *the first* to automatically verify safety and liveness of 10 sophisticated FTDAs such as: asynchronous reliable broadcast and Byzantine agreement, non-blocking atomic commit, condition-based consensus, and one-step consensus. We focused on *threshold-guarded* FTDAs that count messages and compare them to threshold expressions over parameters such as the number of processes n and an upper bound on the number of faults t, for example,

```
if received <ECHO> from at least n-t distinct processes then accept().
```

Distributed algorithms often come in pseudo-code. Hence, we introduced a modelling framework for threshold-guarded algorithms [11] with the following features: the algorithms are written in a machine-readable format (an extension of Promela — the input language of the Spin model checker); message-passing is efficiently expressed via operations on integer counters; and behavior of the threshold-guarded algorithms is preserved [5]. In [10], we introduced new data and counter abstractions that simulate integer operations with abstract operations on intervals with symbolic boundaries, e.g., $[t+1, n-t)$. By checking the abstractions with Spin, we verified two reliable broadcast algorithms for all parameter values.

This approach did not scale to more complex algorithms, and I experimented with bounded model checking. I found that our benchmarks were violating specifications in few steps, and thus I formulated a hypothesis that counter abstractions of threshold-guarded algorithms have small bounded diameters. We proved that this hypothesis indeed holds for all threshold-guarded algorithms [1, 9]. We transferred this result on accelerated counter systems of threshold automata and developed a sound and complete SMT-based bounded model checking technique for reachability properties [8]. Recently, we proved similar result for liveness properties [4], which belong to a fragment of unary temporal logic LTL(F, G) over counter tests.

2. Contribution personnelle de la candidate ou du candidat / *Personal contribution of the applicant*

This research was conducted together with Josef Widder, Helmut Veith, Marijana Lazić, Annu Gmeiner, and Ulrich Schmid. All co-authors developed the main ideas together and contributed to text writing. I contributed to the detailed proofs of [4, 10], and I did most of the detailed proofs of [8]. For [4, 8–10], I implemented new techniques in the tool ByMC and conducted the experiments. For [4], I formulated high-level proof ideas that were elaborated by Marijana Lazić, whom I supervised in this research. I advised Annu Gmeiner in conducting the experiments of [11].

3. Originalité et difficulté / *Originality and difficulty*

The vast majority of distributed algorithms are published with paper & pencil proofs. A small number of them were proven to be correct with proof assistants such as TLAPS, PVS, Isabelle, and Coq. Automatic verification of distributed algorithms is typically limited to model checking of systems of 3–7 components using tools such as TLC, Spin, and NuSMV. However, it is imperative for experts in distributed computing to prove results for an unbounded number of components. By formalizing fault-tolerant distributed algorithms, we found that the standard model checkers are designed for application domains that differ from our benchmarks, e.g., network protocols and concurrent algorithms (Spin), hardware protocols (NuSMV), sequential C (CBMC), etc. Hence, we developed techniques and a tool that efficiently use our domain knowledge.

Parameterized model checking (PMC) is undecidable for many computational models (cf. Apt, Kozen (1986), our survey [2, 12]). PMC is most studied on concurrent protocols such as mutual exclusion and cache coherence. Consequently, many PMC tools were evaluated only on these protocols. Unlike mutual exclusion algorithms, fault-tolerant distributed algorithms are inherently parameterized in many dimensions: the number of processes, the upper bound on faults, and the number of actual faults. That is probably why there are only a few results on parameterized model checking of FTDAs. To express fault-tolerant algorithms, Fisman et al. (2008) extended regular model checking with MSO, but did not offer practical solutions to the verification problem. Alberti et al. (2012) applied MCMT to three fault-tolerant algorithms, and noted that MCMT does not apply to the algorithms that require "*substantial arithmetic reasoning*", e.g., our threshold-guarded algorithms. Likewise, Conchon et al. (2013) more efficiently verified coordinator-based reliable broadcast by Chandra-Toueg with Cubicle.

While our first approach [10] was a generalization of counter abstraction by Pnueli, Xu, and Zuck (2002), we developed an original approach to parameterized verification in our follow-up papers [4, 8, 9]. This approach builds on mathematical arguments that apply acceleration and mover analysis to counter systems of threshold automata [8, 9]. Extending these results to liveness [4] required us to address two main challenges: (1) finding an expressive fragment of LTL that kept the verification problem decidable, and (2) finding reduction arguments that preserve the formulas of that fragment.

4. Validation et impact / *Validation and impact*

I have implemented the techniques [4, 1, 8–10] in my tool ByMC that underwent artifact evaluations at CAV'15 [8] and POPL'17 [4], both successful. Our results were a showcase in the hearing at the Austrian Science Fund for the extension of national research network project RiSE. In recognition of these results, I received a highly competitive grant from the Vienna Science and Technology Fund.

Our experiments on fixed-size systems [11] and parameterized systems [10] show that model checking of a system of 5–7 processes may be as hard as proving correctness for all sizes. In the work presented at CAV'15 [8], we compared our latest SMT-based approach implemented in ByMC with three techniques: acceleration of counter automata in FAST, bounded model checking of counter abstractions in NuSMV, and BDD-based model checking of counter abstractions in NuSMV. All four techniques instantly check simple algorithms (e.g., 10 control locations), but only our SMT-based technique scales to complex algorithms (e.g., 100 control locations). Our results presented at POPL'17 [4] show that ByMC can check liveness, whereas other tools either support only safety, or do not scale to our benchmarks. Thereby, *only* ByMC can automatically verify safety and liveness of complex threshold-guarded fault-tolerant distributed algorithms, for all parameter values.
Following up our work [8–10], Alberti et al.[4] implemented counter abstraction and acceleration in MCMT and verified some of our benchmarks. They concluded that although their array-based encoding is closer to the original pseudo-code, counter abstraction and acceleration are more efficient than model checking of array-based systems in SMT.
Our collaborator Francesco Spegni (Univ. Politecnica Delle Marche, Italy) extended ByMC to check clock synchronization algorithms with UPPAAL. These algorithms require time constraints on message delivery. We are preparing a submission.

5. Diffusion / *Dissemination*

Our results are published at POPL'17 [4], Information & Computation [1], VMCAI'17 [5], CAV'15 [8], CONCUR'14 [9], FMCAD'13 [10], PODC'13 [19], and Spin'13 [11]. My co-authors and I read lectures at 3 summer schools (Logic & Verification'16 in Vienna, VTSA'14 in Luxembourg, SFM'14 [13] in Bertinoro), gave 4 invited talks at conferences and workshops (FMCAD'16 in Silicon Valley, PSSV'16 in St. Petersburg, PSI'15 [7] in Kazan, TMPA'14 in Kostroma), and gave over 25 seminar talks. The tool and benchmarks, as well as virtual machines are available from: [forsyte.at/software/bymc].

Fiche 2 : Abstraction-based parameterized TLA$^+$ checker (work in progress)

1. Description de la contribution / *Description of the contribution*

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. We are aiming at developing two tools: (1) a finite-state model checker — similar to TLC but more efficient — should be able to check only systems with fixed parameters, and (2) a parameterized model checker that should be able to check TLA$^+$ designs with an unbounded number of components.

2. Contribution personnelle de la candidate / du candidat / *Personal contribution of the candidate*

In this project, I am advising Thanh Hai Tran and Jure Kukovec (PhD students) on development of the new tools for TLA$^+$. While we are discussing architectural decisions and analysis algorithms together, the students take lead in tool development. Together with Josef Widder (co-PI) and Marijana Lazić (PhD student), we are developing new verification techniques.

3. Originalité et difficulté / *Originality and difficulty*

Our goal is to verify fault-tolerant distributed systems expressed in TLA$^+$ using advanced parameterized and abstraction-based techniques. We are facing three challenges. (1) Specifications in TLA$^+$ are considerably more expressive than standard software: TLA$^+$ is untyped, it allows quantification over sets, comparison of set cardinalities, and comparison and updates of the states of concurrent components. (2) TLA$^+$ specifications should be checked for an unknown number of components, in order to verify systems such as fault-tolerant distributed algorithms. (3) As TLA$^+$ is used to specify complex systems, not only toy examples, we have to make our tool scalable to such systems. We observed that although TLA$^+$ is extremely expressive, its users are typically following coding idioms. Our insight is to use these idioms to find efficient abstractions, e.g., harvest predicates for predicate abstraction techniques.

4. Validation et impact / *Validation and impact*

In his master thesis, Thanh Hai Tran developed a prototype that implements predicate abstraction and IC3 for a subset of TLA$^+$. We found that, in contrast to structured sequential programs that are composed of many simple statements, a naive application of predicate abstraction does not scale beyond trivial examples, when one treats a TLA$^+$ specification as a monolithic transition relation. However, IC3 combined with predicate abstraction shows a good speed up in comparison to the explicit-state model checker TLC, but needs hints from the user to select good predicates.

5. Diffusion / *Dissemination*

We have collected challenges of model checking TLA$^+$ specifications in a contribution that we presented at EC^2 [17]. In July 2016, Thanh Hai Tran defended his master thesis on "User-guided Predicate Abstraction of TLA$^+$ Specifications".

[4] Alberti, Ghilardi, Orsini, Pagani. Counter Abstractions in Model Checking of Distributed Broadcast Algorithms: some case studies. CILC, 2016

Fiche 3 : Parameterized model checking of protocols

1. Description de la contribution / *Description of the contribution*

In this research line, I investigated model checking of communication protocols, e.g., network and hardware protocols. Given a transition system P, which represents a protocol, and a number $n > 1$, a system P^n consists of n copies of P. The parameterized model checking problem (PMC) answers the following question for a protocol P and a specification φ: does φ hold on P^n for *all* $n > 1$? Solving PMC for a class of protocols is often easier than checking a system P^{1000}.
In my PhD thesis [14], I investigated PMC for protocols that run in networks of standard topologies: rings, stars, and trees. Such networks are naturally defined with network grammars. I extended the framework of network invariants, which reduces PMC to model checking of finitely many networks derived from a network grammar; cf. [21, 3] for a summary. The reduction uses a simulation-checking algorithm that tests whether one transition system matches each step of another transition system with its own step. To make the framework applicable to asynchronous systems, I introduced new simulations: block [33, 36], quasi-block [42], and semi-block [31, 3, 28]. These simulations are insensitive to stuttering, and thus relate asynchronous systems in a finer way than the classical (strong) simulation. I implemented this approach in my tool CheAPS [lvk.cs.msu.ru/~konnov/cheaps/index_en.html] and evaluated it on a few examples [3, 27]. The tool works as follows: (i) it generates networks recognized by a grammar and tries to find a network that simulates the larger networks (i.e., this network is an invariant); and (ii) once invariants are found, the systems composed of the invariants are checked with Spin.
As a follow up of this work, together with colleagues from TU Graz and TU Wien, I have written a survey book on parameterized model checking [2, 12]. We reviewed a number of techniques for various architectures of parameterized systems: token rings, client-server systems, systems with conjunctive and disjunctive guards, and ad hoc networks. We presented the results in a uniform framework and refined proofs for some of the results.
To apply a parameterized verification technique, one has to invest a lot of manual effort, as the system architecture and process behavior must be captured in the language specific to the technique. To solve PMC in a more automated way, we extended the Behavior-Interaction-Priority (BIP) framework to parameterized designs by introducing *first-order interaction logic* [6]. Given a parameterized BIP design, our technique automatically finds the architecture, to which the design belongs, and thus gives us a way of systematically integrating many parameterized model checking techniques in a single tool.

2. Contribution personnelle de la candidate / du candidat / *Personal contribution of the candidate*

As for the work related to my PhD thesis [31, 3, 27, 28, 42, 33, 36], I had the primary role in generating the ideas, writing the proofs, developing the tool, and conducting the experiments. The survey [2, 12] is a joint work with Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Sasha Rubin, Helmut Veith, and Josef Widder. Given my expertise in the topic, I suggested the initial book structure, pointed to the related work and proposed the way to unify the heterogeneous computation models from different papers within a single framework. In addition, I led Chapter 6 (Guarded protocols), that is, I collected the results and did the detailed proofs. The paper [6] is joint work with Tomer Kotek, Wang Qiang, Simon Bliudze, and Joseph Sifakis. First-order interaction logic was initially suggested by Joseph Sifakis. Wang Qiang, Tomer Kotek, and I developed the framework and prepared the initial report during Wang Qiang's internship at our group.

3. Originalité et difficulté / *Originality and difficulty*

As PMC is undecidable, the research efforts in the field focus on finding classes of parameterized protocols that have (semi-) decidable properties. These classes are given in terms of "computational models", that is, mathematical definitions of concurrency, communication, etc., which vary in subtle details. The key features of a particular computational model often become apparent only after analyzing the respective decidability proof. Hence, the survey [12] required us to *reconstruct* the proofs and find a *uniform model* to fit the results. The difficult part of my thesis work was to extend the framework of network invariants, so it worked for *asynchronous* parameterized systems. Finally, the strength of [6] is its originality: we were *the first* to find a method for automatic identification of architectures and techniques for their parameterized verification.

4. Validation et impact / *Validation and impact*

I evaluated the tool CheAPS on several textbook algorithms and a manual abstraction of the RSVP protocol. Qiang Wang evaluated our technique [6] on parameterized BIP designs of four types: token passing rings, cliques of identical components communicating with rendezvous or broadcast, and client-server systems. Our survey [12] has been cited 17 times since Sep. 2015, and we were invited to presented it in the ACM SIGACT News [2].

5. Diffusion / *Dissemination*

The survey on parameterized model checking was published as a book [12] in 2015. We prepared a book overview for ACM SIGACT News [2]. The paper [6] was presented at CONCUR'16. The results of my PhD thesis were published in three journals: Symbolic Computation [3], Automatic Control & Computer Sciences [31], Programming & Computer Software [33]. I presented the results and the tool at international workshops [20, 21, 28] and Russian conferences [34, 27, 35, 36].

Formulaire 4 — PROGRAMME DE RECHERCHE
Form 4 — RESEARCH PROGRAM

Nom / *Last name*: KONNOV
Prénom/*First name*: Igor

Project(s)-team(s) assignation wishes: **VERIDIS**

Title of research program: **Model checking of distributed algorithms for large-scale systems**

My research goal is to develop a collection of efficient verification techniques and tools that scale to distributed algorithms that can be found in the distributed computing literature. I believe that my research program serves as a good foundation for a proposal for an ERC Consolidator grant, which I am going to apply for in a few years.

Motivation and state of the art. Distributed algorithms have many applications: embedded systems (e.g., in cars and planes), telecommunication, hardware designs, and, recently, cloud computing. Because of their inherent complexity and, in many cases, their critical role, verification of such systems has been a research topic for a long time. Until recently, research efforts focused on verification of multiprocessor and embedded systems as exemplified by distributed algorithms that solve the problems like resource allocation, mutual exclusion, and cache coherence. As multiprocessor and embedded systems were predominantly synchronous, verification of synchronous distributed algorithms and systems is a well-understood field that has industrial-scale frameworks and tools, e.g., Esterel.
Large-scale distributed systems — e.g., cloud systems — differ from embedded systems in two crucial aspects: they are inherently *asynchronous*, and at all times a small fraction of individual components in such systems *is faulty*, e.g., crashed, disconnected from the network, etc. A paradigmatic approach to such systems is to replicate computers in such a way that they behave as if they were a single machine — a replicated state machine. In a nutshell, to mitigate at most f Byzantine faults, one builds a system $S(n, f)$ of $n > 3f$ replicas, out of which at least the $n - f$ correct replicas have to agree on the outcome. (Formally, this idea is defined as a consensus problem.) To this end, researchers in distributed computing introduced many distributed algorithms, to name a few: Paxos by Lamport (1998), Practical Byzantine Fault Tolerance by Castro and Liskov (1999), and Raft by Ongaro (2014). Correctness of such algorithms is usually substantiated with paper-and-pencil proofs, which contain high-level correctness arguments but do not guarantee absence of bugs in a detailed version and an actual implementation. Indeed, sophisticated bugs were found in the earlier versions of Raft. In sum:

Correctness of distributed algorithms is vital for bug-free and fault-tolerant operation of large-scale distributed systems.

Figure 4.1 shows pseudo code of a leader-based consensus, which exemplifies features of many algorithms. This algorithm has typical features: (a) n processes run the code asynchronously; (b) at most $f < n/3$ processes may crash; (c) each process i starts with some initial value v_i; (d) the processes send their values to all other processes and count how many messages they received; (e) they have to decide on one value from the set $\{v_1, \ldots, v_n\}$; (f) the processes call the primitives `R_Broadcast` and `R_Delivery` of an underlying reliable broadcast algorithm; and (g) the processes refer to the values sent by a special *leader* process, which is elected by a separate algorithm.

```
1  Function Consensus(v_i)
2  Task T1:
3  r_i ← 0; est_i ← v_i;
4  while true do
5   r_i ← r_i + 1;
6   ---- Phase 1 of round r_i ----
7   broadcast PHASE1(r_i, est_i);
8   wait until (∃ℓ. leader = ℓ and
9    PHASE1(r_i, v) received from ℓ);
10  ---- Phase 2_3 of round r_i ----
11  broadcast PHASE2_3(r_i, est_i);
12  wait until (PHASE2_3(r_i, est) received from n − f proc.);
13  if (at least (n − 2f) PHASE2_3(r_i, est) carry same est = v)
14    then est_i ← v endif;
15  if (the (n − f) PHASE2_3(r_i, est) carry same est = v)
16    then R_Broadcast DECISION(est_i); stop T1 endif;
17 od
18 Task T2: upon R_Delivery of DECISION(v): return v
```

Figure 4.1: Leader-based consensus by Mostefaoui and Raynal (2001)

Despite critical role of distributed algorithms only very few distributed algorithms were formally, let alone automatically, verified. The key problems are as follows: (i) general-purpose model checkers for distributed algorithms fall back to state enumeration, e.g., Spin and TLC; (ii) software model checking tools work best on sequential programs, which have limited degree of non-determinism and no concurrency; (iii) many distributed algorithms, including fault-tolerant algorithms, are parameterized in the number of processes and faults and thus require *parameterized model checking*, e.g., proving $\forall n, f : n > 3f.\ S(n, f) \models \varphi$; and (iv) local process states are often unbounded or parameterized, e.g., in n and f. There is no model checking tool that can deal with the problems (i)–(iv) out of the box. Thus, practitioners recur to the error-prone process of manual abstraction and simplification. To change this situation,

My goal is to create a theoretically coherent framework and tools for automated verification of distributed algorithms and their implementations. Distributed algorithms need verification tools as mature as embedded systems have today.

Approach. In the research literature, distributed algorithms are given in pseudo-code and English, and thus algorithm formalization is a necessary prerequisite to verification efforts. Until recently, for this purpose, we have been using a parametric extension of PROMELA — the language of the Spin model checker. However, there are several drawbacks in using a modeling language specific to a model checker. First, an expert in distributed algorithms has to invest considerable efforts into encoding the algorithm in the special language, and this process is error-prone. Indeed, as the language is optimized for the problem domain of the model checker, the expert has to make many compromises in the encoding. Second, different model checking input languages are hard to translate into each other. For these reasons, we have changed the input language to TLA$^+$ recently. TLA$^+$ has found users at Amazon and Microsoft, and the tools for TLA$^+$ are developed at VERIDIS and MSR INRIA. In our running project APALACHE, we are developing a new model checker for TLA$^+$ (cf. Fiche 2).
When it comes to verification, abundance of undecidability results and techniques for parameterized model checking (cf. our book [12]), as well as our own results on verification of threshold-guarded algorithms (cf. Fiche 1) suggest that there is *no generic technique* for automatic verification of distributed algorithms. On the other hand, as our results show, domain-specific techniques can efficiently verify special classes of distributed algorithms. Thus,

Our key hypothesis is that distributed algorithms can be automatically verified by cooperation of highly specialized techniques, each aiming at a specific algorithmic feature.

Indeed, similar to network stacks, distributed algorithms are not monolithic, but are composed of algorithmic layers: n processes run concurrently many sub-algorithms. For instance, the algorithm shown in Figure 4.1 has three layers: reliable broadcast (lines 11–14), leader oracle (line 8), and asynchronous rounds (variable r_i). Another example is the celebrated Paxos. The core layer of Paxos is called "Synod" algorithm, whose purpose is to achieve consensus among non-faulty processes, that is, they should decide on a single value (assuming "normal operation of the system"). The Synod algorithm is interacting with a leader oracle, implemented by another algorithmic layer. Paxos is invoking many instances of Synod to implement a replicated state machine, by using asynchronous rounds. In other words, *to make the verification task amenable for model checking, we reuse the compositional structure naturally present in distributed algorithms.*

Relation to my previous work. Our results (cf. Fiche 1) broke new ground in automatic verification of fault-tolerant distributed algorithms. However, our techniques cover only one algorithmic layer, that is, reliable broadcast and similar algorithms. Multiple layers need fundamentally new techniques, e.g., leader oracles require us to reason about IDs, linear orders, or probabilities, whereas failure detectors need arguments about real time or partial synchrony. In contrast to threshold-guarded algorithms studied by us, processes in Paxos-like algorithms produce computations and local state spaces that are not a priori bounded by the parameters: by sending an unbounded number of messages (which carry ballot numbers) and collecting transactions in lists of unbounded lengths.
So far, methods for verification of distributed algorithms were disconnected from practical techniques of finding bugs in distributed systems. In the long run, I aim at bridging this gap and connecting research in three areas: computer-aided verification, distributed computing, and distributed systems engineering.

Short-term. I am planning three tasks in the first 12–18 months: (1) Further development of the new TLA$^+$ model checker, (2) Integration of APALACHE results with TLAPS, and (3) Model checking of the Synod algorithm.
Task 1: Further development of the new model checker for TLA$^+$. As we are currently working on a new model checker for TLA$^+$, it would be natural to continue development of this tool at INRIA. In particular, I will continue work on semi-decision procedures for TLA$^+$ and predicate abstraction for TLA$^+$.
Task 2: Integration of APALACHE *results with* TLAPS. A position at the VERIDIS group will allow me to work on a tighter integration of our model checker developed in APALACHE with the TLA$^+$ Proof System, which is developed at VERIDIS and MSR INRIA. As in APALACHE, we are developing tools for parameterized model checking of TLA$^+$ specifications, we aim at integrating the parameterized model checking procedures that can be used by TLA$^+$ Proof System to automatically prove facts about behaviour of distributed systems specified in TLA$^+$.
Task 3: Model checking of the Synod algorithm. Stephan Merz (INRIA Nancy), Josef Widder (TU Wien), and I have started to work on parameterized verification of the Synod algorithm specified in TLA$^+$ by Leslie Lamport[5]. In this algorithm, processes send ballot numbers that may grow arbitrarily large. Hence, ballot numbers and alike pose a challenge for model checkers: even a single process in the system produces infinitely-many states. Moreover, to ensure uniqueness of ballot numbers, consensus algorithms use lexicographic ordering on process IDs and round numbers. Process IDs break symmetry among processes and render counter abstractions such as [10] useless. We will develop abstraction techniques tailored to Synod-like algorithms, e.g., that use ballots, epoch numbers, etc. Automatic verification techniques for the algorithms similar to Paxos are needed by industry, as Paxos modifications are used in cloud systems, e.g., by Google.

[5] L. Lamport. Paxos made simple. ACM Sigact News 32.4 (2001): 18-25.

Long-term. For the more distant future, I propose three research lines: (A) Model checking techniques for algorithmic layers of consensus algorithms; (B) Composition techniques for algorithmic layers; and (C) Model checking of practical implementations of consensus algorithms.

Line A: Model checking for individual algorithmic layers. As discussed earlier, our main insight is to develop model checking techniques that deal with algorithmic layers separately. We have identified four algorithmic layers that are omnipresent in the research literature on consensus algorithms (cf. the book by M. Raynal[6]): (1) reliable broadcast; (2) Synod-like algorithms and asynchronous rounds; (3) failure detectors; and (4) leader oracles and rotating coordinators. As in the research literature, we assume that processes communicate by message-passing. Consensus algorithms vary by their assumptions on synchrony of the processes and delays on message delivery. Layer (1) — reliable broadcast — is already covered by our recent work [4, 1, 8–10]. Therefore, we will develop model checking techniques that deal with layers (2)–(4).

First, we will develop domain-specific abstractions for the classes of algorithms found at each layer. The purpose of abstractions is to encode the algorithmic primitives in well-studied mathematical models that have model checking algorithms or semi-decision procedures, e.g., as we did in [10]. Second, we will seek for reductions of the execution space similar to [4, 8, 9], which allows us to use SMT-based bounded model checking as an efficient and complete method. Introduced by Lipton, reduction became an important proof technique for concurrent systems and an efficient method against state explosion in model checking (called partial order reduction). Apart from that, a reduction of the execution space to a small set of representative executions (as in [4, 8]) allows us to test real implementations, as discussed in Line C.

Line B: Composing verification techniques for algorithmic layers. The techniques developed in Line A will allow us to verify that the algorithmic layers of a consensus algorithm satisfy their specifications. To draw conclusions about correctness of a consensus algorithm as a whole, we need methods for: (1) decomposing the verification problem into the problems for algorithmic layers and (2) automatically composing the verification results.

To address (1), we will investigate methods similar to our recent work [6] that introduced a method for automatic identification of parameterized model checking techniques that are applicable to a given design specified in the BIP framework. This was done for standard communication primitives such as pairwise rendezvous, synchronous broadcasts, and token passing.

To address (2), we will propose methods for compositional reasoning about algorithmic layers. The idea of compositional reasoning is not new, and is offered by many frameworks but for different kind of systems, e.g., Compositional Model Checking (Clarke, Long, and McMillan, 1989), Modular Model Checking (Kupferman and Vardi, 2000), and CMP (McMillan, 1998). The basic idea of these frameworks is to verify an isolated logical module of the system under assumptions about the rest of the system. This local reasoning does not fit fault-tolerant algorithms, where one has to show safety and liveness of *all non-faulty* machines. We conjecture that consensus algorithms need compositional reasoning in terms of algorithmic layers distributed across different machines, as opposite to logical modules on a single machine. The main challenge is to formalize and automate such reasoning, which is implicit in the distributed computing literature.

Line C: Model checking of algorithm implementations. While the results of Lines A-B will allow us to verify algorithms, their implementations may still have bugs. We propose to make bug finding systematic by exercising implementations with the tools for distributed algorithms. Typically, system model checkers take a snapshot of the distributed system, and then they exhaustively (or randomly) explore states reachable up to a fixed depth from the snapshot, both by normal system execution and faulty behavior. As distributed systems have prohibitively many states for state enumeration, all tools use heuristics to find deep bugs, which are reached only by long executions. They use: bounded random walks (MaceMC); bounded state exploration from a live snapshot (CrystallBall); heuristics for partial order and symmetry reductions (MoDist and SAMC).

We propose two ideas to systematically explore system executions: (1) steer message-passing in a distributed system with an algorithm-level model checker, and (2) symbolically execute local process code between message-passing points.

(1) We will use the algorithm-level techniques developed in Lines A-B to generate an execution of a consensus algorithm that violates its specification. By feeding the sequence of message sends and receives from this algorithm-level execution into the message queue of the implementation, we will guide the implementation-level model checker into a global state that is close to an error state. Then, we can exercise code of a single process to reveal the bug.

(2) If an error state is reachable via a message-passing schedule as was produced by (1), then it is sufficient to exercise a local execution of a single process without sending and receiving further messages. Hence, we can use symbolic execution tools (e.g., KLEE) to systematically explore the code of a sequential program (of a single process). We expect this approach to reveal deeper bugs and show higher degree of automation as compared to random walks and search guided by heuristics.

Integration at Veridis Team. The members of Veridis Team are developing techniques for automated verification of concurrent and distributed systems. We share background in formal specification and verification techniques as well as the vision of validating our research results on practically-relevant case studies. In particular, Dominique Méry, Stephan Merz, and I are familiar with concepts of distributed algorithms, and we are all well versed in the TLA$^+$ notation and tools. I will contribute my expertise in many model checking techniques — including parameterized — and expertise in analysis of fault-tolerant distributed algorithms. The work within Veridis on development of automatic theorem provers — notably, of SMT solvers — is beneficial to my research program, since these tools are instrumental to (and may need to be adapted for) parameterized model checking. Parts of the TLA$^+$ toolset are developed in Veridis, and thus it is a perfect environment for continuing my research and development of new model checking techniques for distributed algorithms and TLA$^+$.

[6] M. Raynal. Communication and Agreement Abstractions for Fault-Tolerant Asynchronous Dist. Systems. Synt. Lect. on Dist. Comp. Theory (2010)

Formulaire 5 — LISTE COMPLÈTE DES CONTRIBUTIONS[7]
Form 5 — COMPLETE LIST OF CONTRIBUTIONS[7]

Nom / *Last name*: KONNOV
Prénom/*First name*: Igor

1. Publications caractéristiques/*Representative publications*

Give here (at most 3) publications, representative of your research and know-how, of which you are especially proud. They must be available on your web page.

1. I. Konnov, M. Lazic, H. Veith, and J. Widder. "A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms". In: Proc. of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, Jan 18-20, 2017. pp. 719–734. URL: http://dl.acm.org/citation.cfm?id=3009860 (Publicly available under Creative Commons)
2. I. Konnov, H. Veith, and J. Widder. "SMT and POR beat Counter Abstraction: Parameterized Model Checking of Threshold-Based Distributed Algorithms". In: CAV (Part I). Vol. 9206. LNCS. 2015, pp. 85–102. DOI: 10.1007/978-3-319-21690-4_6. URL: http://forsyte.at/download/konnov-cav15.pdf
3. R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, and J. Widder. "Decidability in Parameterized Verification". In: ACM SIGACT News 47.2 (2016), pp. 53–64. DOI: 10.1145/2951860.2951873. URL: http://forsyte.at/wp-content/uploads/p53-bloem.pdf. This is an overview of the book [12], which I am proud of but not allowed to upload on my web page.

2. Publications

As is customary in my community, the list of authors is ordered alphabetically. In rare cases, when the authors wish to emphasize the essential contributions of some of the co-authors to a paper, they put first the names of the co-authors who contributed most.
As recommended, publications in high-quality journals (ranked Q1 or Q2 by Scimago Journal & Country Rank in 2015) and conferences (ranked $A^\star$ or A by Australian CORE in 2014) are highlighted in **bold font**.

2.1 Revues internationales/*International journals*

[1] **I. V. Konnov, H. Veith, and J. Widder. "On the completeness of bounded model checking for threshold-based distributed algorithms: Reachability". In: *Inf. Comput.* 252 (2017), pp. 95–109. DOI: 10.1016/j.ic.2016.03.006.**

[2] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, and J. Widder. "Decidability in Parameterized Verification". In: *ACM SIGACT News* 47.2 (2016), pp. 53–64. DOI: 10.1145/2951860.2951873.

[3] **I. V. Konnov and V. A. Zakharov. "An invariant-based approach to the verification of asynchronous parameterized networks". In: *Journal of Symbolic Computation* 45.11 (2010), pp. 1144–1162. DOI: 10.1016/j.jsc.2008.11.006.**

2.2 Conférence internationales avec comité de lecture/*Reviewed international conferences*

[4] **I. V. Konnov, M. Lazic, H. Veith, and J. Widder. "A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*. 2017, pp. 719–734. URL: http://dl.acm.org/citation.cfm?id=3009860.**

[5] I. V. Konnov, J. Widder, F. Spegni, and L. Spalazzi. "Accuracy of Message Counting Abstraction in Fault-Tolerant Distributed Algorithms". In: *Verification, Model Checking, and Abstract Interpretation - 18th International Conference, VMCAI 2017, Paris, France, January 15-17, 2017, Proceedings*. 2017, pp. 347–366. DOI: 10.1007/978-3-319-52234-0_19. URL: http://forsyte.tuwien.ac.at/static/download/17kwss-accuracy-vmcai.pdf.

[6] **I. Konnov, T. Kotek, Q. Wang, H. Veith, S. Bliudze, and J. Sifakis. "Parameterized Systems in BIP: Design and Model Checking". In: *CONCUR 2016*. Vol. 59. LIPIcs. 2016, 30:1–30:16. DOI: 10.4230/LIPIcs.CONCUR.2016.30.**

[7] I. Konnov, H. Veith, and J. Widder. “What You Always Wanted to Know About Model Checking of Fault-Tolerant Distributed Algorithms”. In: *Perspectives of System Informatics: PSI 2015, in Memory of Helmut Veith, Revised Selected Papers*. Springer, 2016, pp. 6–21. DOI: 10.1007/978-3-319-41579-6_2.

[8] **I. Konnov, H. Veith, and J. Widder. “SMT and POR beat Counter Abstraction: Parameterized Model Checking of Threshold-Based Distributed Algorithms”. In: *CAV (Part I)*. Vol. 9206. LNCS. 2015, pp. 85–102. DOI: 10.1007/978-3-319-21690-4_6.**

[9] **I. Konnov, H. Veith, and J. Widder. “On the Completeness of Bounded Model Checking for Threshold-Based Distributed Algorithms: Reachability”. In: *CONCUR 2014*. Vol. 8704. LNCS. 2014, pp. 125–140. DOI: 10.1007/978-3-662-44584-6_10.**

[10] 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. DOI: 10.1109/FMCAD.2013.6679411.

[11] 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. DOI: 10.1007/978-3-642-39176-7_14.

2.3 Livres et chapitres de livre/*Books and book chapters*

[12] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, and J. Widder. *Decidability of Parameterized Verification*. Vol. 6. 1. Morgan & Claypool, 2015, pp. 1–170. DOI: 10.2200/S00658ED1V01Y201508DCT013.

[13] 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. DOI: 10.1007/978-3-319-07317-0_4.

[14] I. V. Konnov. “Parameterized Model Checking of Distributed Systems”. In Russian. PhD thesis in Computer Science (Kandidat physiko-matematitcheskih nauk 05.13.11). PhD thesis. Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics, Nov. 2008, pp. 1–198. URL: http://lvk.cs.msu.su/~konnov/publications/konnov_phd_thesis.pdf.

2.4 Autres publications internationales (posters, short papers)/*Other international publications (posters, short papers)*

[15] I. Konnov. *Verifying Safety and Liveness of Threshold-guarded Fault-Tolerant Distributed Algorithms*. Talk at the Helmut Veith Memorial Workshop, Obergurgl, Austria, February. 2017. URL: http://cbr.uibk.ac.at/events/hvw/schedule.php.

[16] I. Konnov. *SMT and POR beat Counter Abstraction: Parameterized Model Checking of Threshold-based Distributed Algorithms*. Workshop contribution at Alpine Verification Meeting, Attersee, Austria, May. 2015.

[17] I. Konnov, H. Veith, and J. Widder. *Challenges in Model Checking of Fault-tolerant Designs in TLA+*. Contribution to the 8th International Workshop on Exploiting Concurrency Efficiently and Correctly, San Francisco, CA, USA, July. 2015. URL: http://multicore.doc.ic.ac.uk/events/ec2/KonnovVeithWidder.pdf.

[18] A. B. Glonina, I. Konnov, V. V. Podymov, D. Y. Volkanov, V. A. Zakharov, and D. A. Zorin. *An experience on using simulation environment DYANA augmented with UPPAAL for verification of embedded systems defined by UML statecharts*. Contribution to the CAV workshop VES13, St. Petersburg, Russia, July. 2013. URL: http://forsyte.at/wp-content/uploads/ves13-gkpvzz.pdf.

[19] **A. John, I. Konnov, U. Schmid, H. Veith, and J. Widder. “Brief announcement: parameterized model checking of fault-tolerant distributed algorithms by abstraction”. In: *PODC*. 2013, pp. 119–121. DOI: 10.1145/2484239.2484285.**

[20] I. Konnov. “CheAPS: a Checker of Asynchronous Parameterized Systems”. In: *WING 2010*. Ed. by A. Voronkov, L. Kovacs, and N. Bjorner. Vol. 1. EPiC Series. EasyChair, 2012, pp. 128–129. URL: http://www.easychair.org/publications/?page=355792421.

[21] I. Konnov. *Parameterized Model Checking by Network Invariants: the Asynchronous Case*. Contribution to: LICS Workshop AISS, Dubrovnik, Croatia, June 2012. 2012. URL: http://forsyte.at/wp-content/uploads/12konnov-aiss.pdf.

[22] I. Konnov, H. Veith, and J. Widder. *Who is afraid of Model Checking Distributed Algorithms?* Contribution to the 5th International Workshop on Exploiting Concurrency Efficiently and Correctly, Berkeley, CA, USA, July 2012. 7 citations excl. self-citations. 2012. URL: http://forsyte.at/wp-content/uploads/2012/07/ec2-konnov.pdf.

[23] V. V. Antonenko and I. V. Konnov. “On the Choice of a Simulation Run-Time Infrastructure based on High-Level Architecture”. In Russian. In: *17th International Conference on Computational Mechanics and Contemporary Application Software Systems 2011 (VMSPPS’2011), Alushta, Ukraine*. 2011, pp. 36–38. ISBN: 978-5-7035-2269-1.

[24] I. V. Konnov and O. Letichevsky. “Model Checking GARP Protocol using Spin and VRS”. In: *International Workshop on Automata, Algorithms, and Information Technologies*. 2010. DOI: 10.1007/s10559-010-9244-8.

[25] I. V. Konnov and V. A. Zakharov. “Using Adaptive Symmetry Reduction for LTL Model Checking”. In: *Proc. International Workshop on Program Semantics, Specification and Verification (PSSV 2010) affiliated with CSR 2010*. 2010, pp. 5–11. URL: http://csr2010.ksu.ru/PSSV.html.

[26] G. A. Klimov, D. D. Kozlov, and I. V. Konnov. “Static analysis for security of web applications developed in Python”. In Russian. In: *Proc. 5th All-Russia Scientific and Technical Conf. Microsoft technologies in theory and practice of programming*. 2008.

[27] I. V. Konnov. “The system for verfication of parameterized models of asynchronous distributed systems (CHEAPS)”. In Russian. In: *Proc. 5th All-Russia Scientific and Technical Conf. Microsoft technologies in theory and practice of programming*. 2008.

[28] V. Zakharov and I. Konnov. “An Invariant-based Approach to the Verification of Asynchronous Parameterized Networks”. In: *International Workshop on Invariant Generation (WING'07)*. 2007, pp. 41–55. URL: http://www.risc.uni-linz.ac.at/publications/download/risc_3128/proceedings.pdf.

2.5 Revues nationales/*National journals*

[29] D. Y. Volkanov, V. A. Zakharov, D. A. Zorin, V. V. Podymov, and I. V. Konnov. “A combined toolset for the verification of real-time distributed systems”. In: *Programming and Computer Software* 41.6 (2015), pp. 325–335. DOI: 10.1134/S0361768815060080.

[30] I. Konnov, V. Podymov, D. Volkanov, V. Zakharov, and D. 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.

[31] 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.

[32] I. V. Konnov and V. A. Zakharov. “Using Adaptive Symmetry Reduction for LTL Model Checking”. In Russian. In: *Modelling and Analysis of Information Systems* 17.4 (2010), pp. 78–87. URL: http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=mais&paperid=38&option_lang=eng.

[33] I. V. Konnov and V. A. Zakharov. “An Approach to the Verification of Symmetric Parameterized Distributed Systems”. In: *Programming and Computer Software* 31.5 (2005), pp. 225–236. DOI: 10.1007/s11086-005-0034-4.

2.6 Conférence nationales avec comité de lecture/*Reviewed national conferences*

[34] I. V. Konnov. “Application of CHEAPS System to Parameterized Model Checking of Distributed Systems”. In Russian. In: *Proc. 3rd All-Russia Conf. on Methods and Techniques of Information Processing*. Moscow, 2009, pp. 116–122. ISBN: 978-5-89407-373-3.

[35] V. A. Zakharov and I. V. Konnov. “On the Verification of Asynchronous Parameterized Distributed Programs”. In Russian. In: *Proc. 2nd All-Russia Conf. on Methods and Techniques of Information Processing*. MAKS Press, Moscow, 2005, pp. 267–372. ISBN: 5-89407-230-1.

[36] I. V. Konnov and V. A. Zakharov. “On the Verification of Parameterized Symmetric Distributed Programs”. In Russian. In: *Proc. 1st All-Russia Conf. on Methods and Techniques of Information Processing*. MAKS Press, Moscow, 2003, pp. 395–400. ISBN: 5-89407-163-1.

2.7 Rapports de recherche et articles soumis/*Research reports and publications under review*

[37] I. Konnov, M. Lazic, H. Veith, and J. Widder. *SMT and POR beat Counter Abstraction: Parameterized Model Checking of Threshold-Based Distributed Algorithms*. Submitted to the special issue in memoriam Helmut Veith of Formal Methods in System Design, Springer. 2017.

[38] I. Konnov, M. Lazic, H. Veith, and J. Widder. *A Short Counterexample Property for Safety and Liveness Verification of Fault-Tolerant Distributed Algorithms*. Extended version of the POPL'17 paper including the proofs. 2016. URL: http://arxiv.org/abs/1608.05327.

[39] A. John, I. Konnov, U. Schmid, H. Veith, and J. Widder. *Counter Attack on Byzantine Generals: Parameterized Model Checking of Fault-tolerant Distributed Algorithms*. Oct. 2012. URL: http://arxiv.org/abs/1210.3846.

[40] A. John, I. Konnov, U. Schmid, H. Veith, and J. Widder. *Starting a Dialog between Model Checking and Fault-tolerant Distributed Algorithms*. Oct. 2012. URL: http://arxiv.org/abs/1210.3839.

[41] P. Bulychev, I. V. Konnov, and V. A. Zakharov. “Computing (bi)simulation relations preserving CTL^*_{-X} for ordinary and fair Kripke structures”. In: *Mathemathical Methods and Algorithms, Institute of Systems Programming of the Russian Academy of Sciences*. Vol. 12. 2006, pp. 59–76. URL: http://discopal.ispras.ru/pdfs/issue-2006-12/cs-isp-sbornik.pdf.

[42] I. Konnov and V. Zakharov. “On the verification of asynchronous parameterized networks of communicating processes by model checking”. In: *Mathemathical Methods and Algorithms, Institute of Systems Programming of the Russian Academy of Sciences*. Vol. 12. 2006, pp. 37–58. URL: http://discopal.ispras.ru/pdfs/issue-2006-12/cs-isp-sbornik.pdf.

3. Développements technologiques : logiciel ou autre réalisation / *Technology development : software or other realization*

I developed two parameterized model checking tools: Byzantine Model Checker (ByMC) and Checker of Asynchronous Parameterized Systems (CheAPS). While I am adding new features to ByMC and maintaining it, I discontinued support of CheAPS.

3.1 Byzantine Model Checker (ByMC)

The ByMC status according to the evaluation criteria is as follows: A2-up3, SO4, SM2-up3, EM2, SDL4, OC4-down3: DA4, CD4-down3, MS2, that is:

- **Audience**: 2-up3. Currently, the tool audience is researchers from my group and collaborators. I believe that ByMC has attracted interest of other research groups, due to our recent publications at CAV’15 [8] and POPL’17 [4]. Hence, the tool should go to level 3 soon.
- **Software Originality**: SO4. This software is implementing several original techniques. Implementation of these techniques required to address many technical challenges, e.g., introduce sophisticated SMT encodings of our problems.
- **Software Maturity**: SM2-up3. As the tool grew up out of a quick prototype for testing new ideas, the tool contains old undocumented modules, mostly deprecated. The modules implementing new techniques are well-documented and tested with unit tests (ounit), as well as regression tests. When the old code is refactored, ByMC will go to level 3.
- **Evolution and Maintenance**: EM2. I keep the software alive and periodically add new techniques.
- **Software Distribution and Licensing**: SDL4. The source code is distributed under Simplified BSD License and is available from: [https://github.com/konnov/bymc]. Virtual machines containing the releases are available from: [http://forsyte.at/software/bymc]. The tool benchmarks are available from: [https://github.com/konnov/fault-tolerant-benchmarks].
- **Own Contribution**: OC4-down3. Being the main contributor since 2012, I am going to hand over the tool development and maintenance to an able student, as my interests shift towards the new TLA$^+$ model checker developed in Apalache.

Additional. ByMC is implemented using three languages: the tool core is implemented in `OCaml`; preprocessing and postprocessing are done in `Python`; and basic commands are implemented in `Bash`. ByMC implements series of techniques for parameterized model checking of threshold-guarded fault-tolerant distributed algorithms. It receives a description of a distributed algorithm on its input in one of two formats: (a) code in parametric Promela [13, 11], or (b) a threshold automaton [9]. ByMC implements the following tool chains:

1. ByMC computes a threshold automaton (by applying data abstraction) and checks the counter system with SMT-based bounded model checking [4, 8]. This search is complete at the level of threshold-automata (there still might be spurious counterexamples due to data abstraction).
2. ByMC applies parametric data and counter abstractions [10] and checks the abstract model with either Spin or NuSMV. Spurious counterexamples due to counter abstraction are refined with a CEGAR loop [10].
3. ByMC computes a threshold automaton (by applying data abstraction) and checks the counter system with FAST [9].
4. Given parameter values, e.g., $n = 4$, $f = 1$, ByMC translates its input into standard Promela and checks it with Spin.

3.2 Checker of Asynchronous Parameterized Systems (CheAPS)

The CheAPS status according to the evaluation criteria is as follows: A1, SO3, SM2, EM1, SDL4, OC4: DA4, CD4, that is:

- **Audience**: A1. The tool was developed as a prototype for testing new ideas.
- **Software Originality**: SO3. This software is reusing new known ideas and introducing a few new ideas.
- **Software Maturity**: SM2. The tool has terse but usable documentation, there is some software engineering, basic usage should work.
- **Evolution and Maintenance**: EM1. This software is no longer developed.
- **Software Distribution and Licensing**: SDL4. The source code is distributed under Simplified BSD License and is available from: [http://lvk.cs.msu.su/~konnov/cheaps/index_en.html].
- **Own Contribution**: OC4. I was the main contributor.

Additional. CheAPS was implemented in two languages: the tool core that required high-performance is written in C++, whereas the less performance-sensitive modules such as the parser, model preprocessor, and visualization of counterexamples are written in Python.

4. Impact socio-économique et transfert / *Socio-economic impact and transfer*

Please describe your activities that have had a socio-economic impact. Note that this is not limited to industrial transfer, although it is clearly included. For each such activity, please fill in the questionnaire below (one form per activity).

Diplomas of the PhD and master degrees

Please find attached below the copies of the diplomas that certify PhD and Specialist (approx. MSc) degrees awarded by Lomonosov Moscow State University (Russia) in 2003 and 2009 respectively.
My PhD degree is officially called “candidate of physico-mathematical sciences” (k.f.-m.n.), which is a first post-graduate scientific degree in Russia[8] [9]. The rules of a PhD defense are established by the Higher Attestation Commission (VAK) of the Russian Ministry of Education and Science. By these rules, I defended my thesis on “Parameterized Model Checking of Distributed Systems” in front of the Dissertation Counsil at Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University. The council was composed of 19 Professors and included my supervisors Prof. Zakharov and Prof. Smeliansky and two official opponents Prof. Sokolov (Yaroslavl State University) and Dr. Kulyamin (Institute of Systems Programming of the Russian Academy of Sciences). The protocol of the defense, as well as the scientific abstract[10] and the manuscript[11] were sent to the Higher Attestation Commission. After a positive review by the Higher Attestation Commission, I was awarded a PhD degree on **13th of February 2009**.
By the VAK rules, I had to complete my PhD thesis and all supporting documents in Russian. The Higher Attestation Commission maintains a list of approved journals, most of them were Russian journals at the time of my studies. Publications in these journals were counted towards PhD defense. The journal publications in English[12] and[13] cover large parts of my PhD thesis.
In June 2003, I was awarded a Specialist degree after completing a five years graduate program in “Applied mathematics and informatics”, which is considered equivalent to master studies in computer science. By the end of my graduate studies, I passed state-approved final examinations and defended my thesis[14].

[8] Candidate of Sciences: https://en.wikipedia.org/wiki/Candidate_of_Sciences

[9] Study on the organisation of doctoral programmes in EU neighbouring countries, p. 3: http://eacea.ec.europa.eu/tempus/participating_countries/study/doctoral_programme/russia.pdf

[10] The scientific abstract of my PhD thesis (in Russian): https://cs.msu.ru/sites/cmc/files/theses/konnov_autoref.pdf

[11] My PhD thesis (in Russian): http://lvk.cs.msu.ru/~konnov/publications/konnov_phd_thesis.pdf

[12] I. V. Konnov and V. A. Zakharov. An invariant-based approach to the verification of asynchronous parameterized networks. In: Journal of Symbolic Computation 45.11 (2010), pp. 1144–1162. Elsevier, 2010. DOI: 10.1016/j.jsc.2008.11.006. Online draft: http://lvk.cs.msu.su/~konnov/publications/kz09_jsc_invariant_draft.pdf

[13] 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. Allerton Press. DOI: 10.3103/S0146411610070035.

[14] Specialist thesis (in Russian): http://lvk.cs.msu.su/~konnov/2003konnov-masters.pdf