CV

Alexey Bakhirkin

Email	abakhirkin@gmail.com (personal)
	alexey.bakhirkin@univ-grenoble-alpes.fr (work)
Web	http://www-verimag.imag.fr/~bakhirki/ (work, note no 'n' in the end)
Occupation	Postdoctoral Reseacher, Verimag/Université Grenoble Alpes
Education	PhD (Computer Science, University of Leicester, UK)
Now living in	Grenoble, France
Citizenship	Russian
Born	1987 in Moscow, Russia
Languages	English – fluent, French – beginner, Russian – native.

Research Interests Formal Methods, Program Analysis, Runtime Verification, Temporal Logics

About me I'm currently a postdoctoral researcher at Verimag (Grenoble, France). My current research directions are program analysis (funded by the European STATOR project), runtime verification and temporal logics. My task is to develop new algorithms for program analysis and runtime verification, provide prototype implementations (that highlight the advantages of the algorithms without being compete software products), and describe this in research papers.

I have a PhD in Computer Science from the University of Leicester, UK. Before that, I studied computing at Bauman University in Moscow, Russia.

For a few years, I was a software developer in Moscow. I can program in different languages and environments. I have good knowledge of C++, C#, Java, OCaml.

Brief Timeline

This is a very brief timeline. What I'm doing now, I explain just below. What I was doing in the past, I explain further down.

Oct 2016 – present	Postdoctoral Researcher at Verimag/Université Grenoble Alpes, France.
Jun 2012 – Sep 2016	PhD Student in Computer Science, University of Leicester, UK.
Aug – Oct 2014	Intern at Microsoft Research, Cambridge, UK.
2007 – 2012	Software Developer in Moscow, Russia
2004 – 2010	Student in Computing (Engineering degree, similar to MSc) at Bauman Moscow State University, Russia

Current Work

Runtime Verification and Temporal Logics I work with Signal Temporal Logic (STL) and related formalisms. They allow to specify the desired (or undesired) shape of an output signal from a system and admit efficient algorithms that check a signal against a specification.

I'm currently working on extending STL in a way that allows to prove more interesting properties, but keeps monitoring efficient. The corresponding publication under review, and the ideas are implemented in a prototype tool written in C++, see here.

My main published contribution is an algorithm (with a prototype in OCaml) that can find parameter values for an STL formula, such that it is satisfied by observed signals (so called parameter identification problem) [2].

I also worked on Timed Regular Expressions (TRE, another similar formalism) and developed a competitively efficient procedure for timed pattern matching with a prototype in C++) [5, 3].

Program Analysis I am working on an abstract interpreter for constrained Horn clauses (see here) that aims at overcoming the challenges of using abstract interpretation in this setting: combining forward (from the initial conditions to the goal) and backward (from the goals to the initial condition) analysis [6], analysing non-linear systems of clauses, analysing systems with both numeric and Boolean variables [4], etc.

Horn clauses promise to be a convenient intermediate language to encode problems from program analysis, and potentially from other areas of formal methods. A program (e.g., in C) can be translated to a system of Horn clauses, and then an analysis tool (like the one I'm working on) can find potential safety violations (e.g., out-of-bounds array accesses).

As a part of this work, I am developing a prototype of a relational domain that combines linear and Boolean constraints.

This work is hosted by David Monniaux and funded by the European STATOR project.

Programming Skills

I can program in different languages and environments. I have good knowledge of modern C++, C#, Java, OCaml; I worked on projects that use them both in Windows and Linux; both academic and commercial.

I have experience working in a team and using appropriate team development practices and tools: version control (SVN, Git), automated testing, continuous integration, bug tracking, etc. I have experience with a number of build systems and scripting languages (Python, Bash).

I have experience with Oracle and MS SQL databases.

Teaching

In 2012 – 2016, I was a teaching assistant for several modules, including: C++ programming, model checking (MSc level), Java programming, Haskell programming, operating systems (BSc level). In 2016, I supervised a group of BSc students on a project involving researching a topic and preparing a report and presentations.

Publications

[1] Alexey Bakhirkin, Thomas Ferrère, Thomas A. Henzinger, and Dejan Nickovic. “The first-order logic of signals: keynote”. In: *International Conference on Embedded Software (EMSOFT)*. Download link: pdf. 2018.

[2] Alexey Bakhirkin, Thomas Ferrère, and Oded Maler. “Efficient Parametric Identification for STL”. In: *International Conference on Hybrid Systems: Computation and Control (HSCC)*. Nominated for the best repeatability evaluation award. Also presented in FAC 2018 workshop. Download links: pdf, hal, slides. 2018.

[3] Alexey Bakhirkin, Thomas Ferrère, Dejan Nickovic, Oded Maler, and Eugene Asarin. “Online Timed Pattern Matching Using Automata”. In: *Formal Modeling and Analysis of Timed Systems (FORMATS)*. Download links: pdf, hal, slides. 2018.

[4] Alexey Bakhirkin and David Monniaux. “Extending Constraint-Only Representation of Polyhedra with Boolean Constraints”. In: *Static Analysis Symposium (SAS)*. Download links: pdf hal, slides, video. 2018.

[5] Alexey Bakhirkin, Thomas Ferrére, Oded Maler, and Dogan Ulus. “On the Quantitative Semantics of Regular Expressions over Real-Valued Signals”. In: *Formal Modelling and Analysis of Timed Systems (FORMATS)*. Also presented in MT-CPS 2018 workshop. Download links: pdf, hal, slides. 2017.

[6] Alexey Bakhirkin and David Monniaux. “Combining Forward and Backward Abstract Interpretation of Horn Clauses”. In: *Static Analysis Symposium (SAS)*. Also presented in HCVS 2017 workshop. Download links: pdf, hal, slides. 2017.

[7] Alexey Bakhirkin. “Recurrent Sets for Non-Termination and Safety of Programs”. Download links: official , printer-friendly. PhD thesis. University of Leicester, Department of Informatics, 2016.

[8] Alexey Bakhirkin and Nir Piterman. “Finding Recurrent Sets with Backward Analysis and Trace Partitioning”. In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Download links: pdf, lra, slides. 2016.

[9] Alexey Bakhirkin, Josh Berdine, and Nir Piterman. “A Forward Analysis for Recurrent Sets”. In: *Static Analysis Symposium (SAS)*. Download links: tr pdf, lra, slides. 2015.

[10] Alexey Bakhirkin, Josh Berdine, and Nir Piterman. “Backward Analysis via over-Approximate Abstraction and under-Approximate Subtraction”. In: *Static Analysis Symposium (SAS)*. Download links: tr pdf, lra, slides. 2014.

[11] Alexey Bakhirkin. “A comparison of blocking and non-blocking synchronization in object-based software transactional memory”. In: *Parallel Computations and Control Problems (PACO)*. Download link: pdf. 2010.

Tools

A path-focusing abstract interpreter for Horn clauses – here.

An evaluator of STL (an not only) formulas – here.

Misc. Talks

Does My Program Ever Finish – a brief introduction to (non-)termination. At BCS event at Leicester, 2015. Slides – here.

Software Transactional Memory – at Moscow ALT.NET group meeting, 2011. In Russian. Video – here.

Other

In Leicester, I was organizing the departmental PhD seminars – a regular event where we invited PhD students from UK universities to talk about their research.

In August 2013, I attended Marktoberdorf Summer School on Software Systems Safety.

I'm have the "Microsoft Certified Professional" and "Microsoft Certified Technology Specialist: .NET Framework 4, Web Applications" certifications.

Past Timeline

June 2012 – Sep 2016 PhD Student in Computer Science, University of Leicester, UK. Supervisor: Nir Piterman. I developed two novel techniques (based on abstract interpretation) that allow to find non-terminating behaviours in programs [8, 9]. For programs that are supposed to terminate (e.g., event dispatch routines), existence of a non-terminating behaviour is a bug and non-termination analyzers can be seen as bug-finding tools.

In another work [10], I studied how a non-termination proof can be used in a safety proof.

My thesis [7] was based on the above three papers [8, 9, 10].

Also, I had experience with shape analysis with 3-valued logic and the tool TVLA.

Aug – Oct 2014 Intern at Microsoft Research, Cambridge, UK. Host: Josh Berdine. The goal of the internship was to better understand scheduling in abstract interpretation (the problem is similar to choosing in which order to resolve equation when solving a system with an iterative method). Apart from learning the state of the art, I formalized some knowledge and experience that was accumulated in MSR on this topic.

2007 – 2012 Working as Software Developer in Moscow, Russia I worked for several companies (will provide details if needed). I developed new and maintained existing information systems. I mostly worked with the .NET platform and Oracle database, and I had limited experience with other technologies (web, GIS, etc).

Sep 2004 – Jul 2010 Student in Computing (Engineering degree, similar to MSc) at Bauman Moscow State University, Russia. My final project was on software transactional memory – I implemented two STM algorithms in C# and performed benchmarking. The main results of this work were published in a local conference [11].