Aleksandar Milicevic

CONTACT INFORMATION

Microsoft Corporation
9255 Towne Centre Dr, Ste 400
San Diego, CA 92121 USA

e-mail: almili@microsoft.com
web: https://aleksandarmilicevic.github.io

CURRENT POSITION

Senior Software Engineer at Microsoft **August, 2015 - present**
in the *Tools for Software Engineers* Group

Currently working on a modern build system that enables fast, reliable, distributed, cacheable, and incremental builds for projects ranging from single-developer apps to large-scale enterprise systems.

Filesystem Virtualization for Linux. Lead role in designing, implementing, and optimizing a virtual filesystem for Linux that allows a client's filesystem to be mimicked on a remote server, overlaid on top of a server's local filesystem, with client files being uploaded selectively and on-demand, as soon as they become needed. (FUSE, OverlayFs, Posix, gRPC, C++, C#)

Process Sandboxing and Substitution for Linux. Lead role in designing, implementing, and optimizing a library for dynamically observing (and preventing undesired) file accesses of a process, as well as substituting its select child processes and executing them in a virtual filesystem on a remote server. (LD_PRELOAD, rtld-audit, C++, C#, Azure VMs)

Process Sandboxing for macOS. One of two main contributors to a process sandboxing library implemented as a kernel extension for macOS. Designed/implemented a concurrent lock-free trie data structure, used it to implement layers of caching for the code running in kernel mode, resulting in a 35% performance increase. (Kext, KAuth, TrustedBsd, C++, C#)

Debugger and Query Language for Analyzing Build Logs. Designed/implemented a first-order relational query language for conveniently analyzing build logs data and graphically representing the results in a VSCode debugger session. (VSCode Extension; C#, TypeScript)

Extend a Build Engine to Support Daemon Processes. Argued strongly for developing a generic support instead of hardcoding an ad-hoc implementation for a single use case we had at a time. This is atypical for build engines, which can typically only run a process after all its dependencies have completed. In some rare cases, mostly for performance reasons, it is useful to have a process (e.g., a service, or a daemon) that runs throughout the build. The argument prevailed, I designed the abstractions, implemented the original use case, and observed a 90% reduction in build tail time. By now several more use cases have benefited from this technology. (C#)

RESEARCH INTERESTS

Declarative programming, specification languages, executable specifications, programming languages, connecting high-level specifications with low-level code, formal methods, software verification, program synthesis, program analysis, software engineering.

EDUCATION

Massachusetts Institute of Technology, Cambridge, Massachusetts USA

Ph.D., Computer Science, May 2015
- Topic: "*Advancing Declarative Programming*" (advised by Prof. Daniel Jackson)

M.S., Computer Science, September 2010
- Topic: "*Executable Specifications for Java Programs*" (advised by Prof. Daniel Jackson)

School of Electrical Engineering, Belgrade, Serbia

B.Sc. in Computer Science, November 2007

- Topic: Parallel Test Generation and Execution with Korat (advised by Prof. Dragan Milicev)

Academic Experience

Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

Research Assistant **August, 2008 - May 2015**
Includes Ph.D. research, Masters-level coursework and research projects.

Teaching Assistant **Spring 2009, Fall 2009**
"6.005 Elements of Software Construction": gave recitations, graded problem sets and projects.

Research Internships

Microsoft Research, Redmond, WA, USA
Research intern **June, 20011 - August, 2011**
Worked with Rustan Leino on program synthesis from first-order declarative specifications.

Microsoft Research Cambridge, Cambridge, United Kingdom
Research intern **June, 2009 - August, 2009**
Worked with Hillel Kugler on analyzing and executing Live Sequence Charts using SMT.

University of Illinois at Urbana Champaign, Urbana, Illinois, USA
Visiting Scholar **August, 2006 - September, 2006**
Worked with Darko Marinov on bounded-exhaustive test input generation.

Industry Experience

Serbian Object Laboratories, Belgrade, Serbia
Software Engineer **March, 2006 - August, 2008**
Actively worked on the development of the EDMT Server (www.bmmsoft.com). Technologies used: WebWork, Java Servlets, WS, SOAP, JSP, HTML, CSS, JS, AJAX, with Sybase IQ database.

Google Inc., New York, New York, USA
Software Engineering Intern **July, 2007 - September, 2007**
Worked with Nemanja Petrovic on decoding barcodes from images taken with a cell phone.

Research Projects

- *Alloy** (`https://aleksandarmilicevic.github.io/hola`): a general-purpose, higher-order relational constraint solver (over bounded domains).
- *Sunny* (`https://github.com/aleksandarmilicevic/sunny.js`): a model-based, event-driven, policy-agnostic paradigm for devel-oping reactive web applications.
- *αRby* (`https://aleksandarmilicevic.github.io/arby`): an embedding of a declarative modeling/specification language (alloy) into an imperative object-oriented programming language (ruby).
- *Squander* (`http://people.csail.mit.edu/aleks/squander`): a unified environment for execution of declarative specification (written in first-order relational logic) and imperative Java code.
- *Jennisys* (`http://research.microsoft.com/en-us/projects/jennisys`): a programming language and a synthesis tool from declarative first-order specifications to imperative code.
- *The Alloy Analyzer* (`https://alloytools.org`): an automated model finder for a first-order relational specification language.
- *Korat* (`http://korat.sourceforge.net`): a tool for bounded-exhaustive generation of test inputs based on complex constraints the inputs must satisfy.

Publications

P. Nie, A. Çelik, M. Coley, **A. Milicevic**, J. Bell, M. Gligoric. Debugging the Performance of Maven's Test Isolation: Experience Report.*ISSTA 2020, Los Angeles, USA*

B. Buhse, T. Wei, Z. Zang, **A. Milicevic**, M. Gligoric. VeDebug: Regression Debugging Tool for Java *ICSE Demo 2019, Montreal, Canada.*

A. Çelik, M. Vasic, **A. Milicevic**, M. Gligoric. Regression Test Selection Across JVM Boundaries. *FSE 2017, Paderborn, Germany.*

M. Vasic, Z. Parvez, **A. Milicevic**, M. Gligoric. File-level vs. Module-level Regression Test Selection for .NET. *FSE 2017 (Industry Track), Paderborn, Germany.*

E. Kang, **A. Milicevic**, D. Jackson. Multi-Representational Security Analysis, *FSE 2016, Seattle, WA, USA.* [Distinguished Paper Award]

A. Çelik, A. Knaust, **A. Milicevic**, M. Gligoric. Build System with Lazy Retrieval for Java Projects, *FSE 2016, Seattle, WA, USA.*

A. Milicevic. Advancing Declarative Programming, *Massachusetts Institute of Technology, Ph.D. Thesis, May 2015.*

A. Milicevic, J. P. Near, E. Kang, and D. Jackson. Alloy*: A Higher-Order Relational Constraint Solver, *ICSE 2015, Florence, Italy.* [Distinguished Paper Award]

A. Milicevic, J. P. Near, E. Kang, and D. Jackson. Alloy*: A Higher-Order Relational Constraint Solver, *MIT CSAIL Technical Report, September 2014.*

A. Milicevic, and D. Jackson. Preventing Arithmetic Overflows in Alloy (extended journal version), *Science of Computer Programming, May 2014.*

A. Milicevic, I. Efrati, and D. Jackson. αRby—An Embedding of Alloy in Ruby, *ABZ 2014, Toulouse, France.*

A. Milicevic, M. Gligoric, D. Marinov, and D. Jackson. Model-Based, Event-Driven Programming Paradigm for Interactive Web Applications, *Onward! 2013, Indianapolis, Indiana, USA*

K. R. M. Leino, and **A. Milicevic**. Program Extrapolation with Jennisys, *Splash 2012, Tucson, Arizona, USA.*

A. Milicevic, and D. Jackson. Preventing Arithmetic Overflows in Alloy, *ABZ 2012, Pisa, Italy.*

A. Milicevic, D. Rayside, K. Yessenov, and D. Jackson. Unifying Execution of Imperative and Declarative Code, *ICSE 2011, Waikiki, Honolulu, Hawaii.*

J. P. Near, **A. Milicevic**, E. Kang, D. Jackson. A Lightweight Approach to Construction and Evaluation of a Dependability Case, *ICSE 2011, Waikiki, Honolulu, Hawaii.*

A. Milicevic, and H. Kugler. Model Checking with SMT and Theory of Lists, *3rd NASA Formal Method Symposium (NFM 2011), Pasadena, California, USA.*

A. Milicevic. Executable Specifications for Java Programs, *Massachusetts Institute of Technology, Master's Thesis, September 2010.*

D. Rayside, **A. Milicevic**, K. Yessenov, G. Dennis, and D. Jackson. Agile Specifications, *OOPSLA Onward! 2009 (short paper), Orlando, Florida, USA.*

D. Rayside, Z. Benjamin, J. Near, R. Sing, **A. Milicevic**, and D. Jackson. Equality and Hashing for (almost) Free: Generating Implementations from Abstraction Functions, *ICSE 2009, Vancouver, Canada.*

S. Misailovic, **A. Milicevic**, N. Petrovic, S. Khurshid, and D. Marinov. Parallel Test Generation and Execution with Korat, *ESEC/FSE 2007, Dubrovnik, Croatia.*

A. Milicevic, S. Misailovic, D. Marinov, and S. Khurshid. Korat: A Tool for Generating Struc-

turally Complex Test Inputs, *ICSE Demo 2007, Minneapolis, Minnesota, USA*.

S. Misailovic, **A. Milicevic**, S. Khurshid, and D. Marinov. Generating Test Inputs for Fault-Tree Analyzers using Imperative Predicates, *STEP 2007, Memphis, Tennessee, USA*

Class Projects

- *Software model checking using the SMT Theory of Lists* **December 2010**
  (Foundations of Program Analysis) Resulted in a publication in NFM'11.
- *Puzzler* **May 2009**
  (Natural Language Processing) Solver for natural-language logic puzzles (e.g., the famous Einstein puzzle) via a translation to formal relational logic and a use of an automated constraint solver for it. Done in collaboration with colleagues Joseph P. Near and Eunsuk Kang.
- *Visual CPU simulator* **July 2006**
  (Computer Architecture) Register Transfer Logic view, per-clock, per-instruction and per-program simulation advance, real-time register and memory modification, compiler from an assembly language. Done in collaboration with Ana Hadzievska, Dusan Matic, Milos Petrovic, Milos Siroka.
- *Multithreading library for the 16-bit C++ compiler* **July 2005**
  (Operating Systems) Java-like threading model for the 16-bit C++ compiler. Features: context switching, explicit synchronous preemption, asynchronous preemption (caused by an interrupt), time sharing, round-robin scheduling. Concepts: semaphores, events, mutexes, monitors.