Tihomir Gvero

Avenue du Temple 13b
1012 Lausanne
+41 76 567 95 22
tihomir.gvero@gmail.com

Strengths

- PhD in Computer Science
- 8+ years of practical experience in programming
- Strong analytical skills and abstract thinking

Professional Experience

2015-present **IT Developer and Consultant, Itecor, Switzerland**

- Implemented Health Information System for Médecins Sans Frontières which consists of a web portal based on Java technologies and a Swing desktop application
- Maintained the system and wrote Selenium tests for the web portal

2009-2015 **Doctoral Assistant, LARA, IC, EPFL, Switzerland**

- Implemented tools and algorithms for automatic software testing and software synthesis
- Authored eight research papers that appeared at top computer science conferences
- Participated in teaching activities, gave talks at academic conferences and seminars

2010 **Summer Intern, Microsoft Research Redmond, USA**

- Implemented techniques that enable Pex, an automatic white-box test input generation tool for .NET, to increase the coverage of tested code
- The techniques are incorporated in Visual Studio 2010-2015 (tools Code Digger and Smart Unit Tests)

2008 **Summer Intern, LARA, IC, EPFL, Switzerland**

- Implemented a technique that transforms a test input generator into a test output checker

2007 **Summer Intern, UIUC, USA**

- Implemented a technique that speeds up the Java PathFinder (JPF), a model checker developed in NASA

Education

2009-2015 PhD in Computer Science, Lab for Automated Reasoning and Analysis (LARA), IC
EPFL, Switzerland

2007-2009 M.Sc. in Computer Science, School of Electrical Engineering
University of Belgrade, Serbia, GPA 10.00/10.00

2003-2007 B.Sc. in Computer Science, School of Electrical Engineering
University of Belgrade, Serbia, GPA 9.44/10.00

Projects

Health Information System which consists of: 1) a web portal that allows doctors to define patient forms and aggregate data tables, and 2) a local Swing desktop application which allows doctors to download, display and manipulate forms and tables. Form and table descriptions are stored in databases, both at the portal and the local application. The local application communicates with the portal and receives updates on forms and tables. The portal and the application are based on the MVC architecture. Technologies: JSP, Spring MVC, Hibernate, SQL Server, Swing, H2, Selenium, Tomcat. Implemented in Java, JavaScript, SQL and HTML.

CPU Simulator is a Swing based visual simulator of a processor with emphasis on register-transfer level. It allows a user to simulate the execution of an assembly program. The simulation is performed per-clock, per-instruction or per-program. Technologies: Swing and Java threads. Implemented in Java.

AnyCode is a tool that uses a textual input to synthesize and suggest Java code snippets. It has a flexible input that supports synonyms and other related words. It ranks the snippets based on the statistics from the large Java GitHub corpus. Technique: uses unigram (declaration occurrence frequency) and PCFG, statistical language models, to guide code synthesis. Results: synthesizes expected snippets in 90% of the benchmarks, on average in 60ms. Technologies: CoreNLP, WordNet, Eclipse JDT and Java thread pools. Implemented in Java (as an Eclipse plugin).

InSynth is a tool that uses a type as input to synthesize and suggest Scala code snippets. It ranks the snippets based on the statistics from a corpus of Scala programs. Technique: synthesis guided by a unigram model. Results: synthesizes expected snippets in 96% of the benchmarks, on average in 150ms. Implemented in Scala (as an Eclipse plugin).

Automatic Invariant Inference (AI²) is a technique that aids Pex to generate complex test inputs, which are hard to construct due to access control constraints. Technique: automatically extracting constraints from C# code, solving the constraints using the Z3 SMT constraint solver, and using the results of Z3 to generate test inputs. Results: Pex with AI² covers 87% code under test, 19% more than Pex without AI², and 7% more than Randoop, a random test input generation tool. Technologies: Pex API and Z3. Implemented in C# (integrated in Visual Studio 2010-2015).

UDITA is a Java-like language that allows testers to combine declarative and imperative test input descriptions to create more expressive test generators. Technique: new backtracking search algorithm. Results: with UDITA we have discovered a number of bugs in the Eclipse and NetBeans refactoring engines, the Sun javac compiler, and JPF. Implemented in Java.

UNDO backtracking optimization that speeds up test input generation in JPF by order of magnitude. Implemented in Java.

Technical Skills

Algorithms and Techniques
AI Search, Software Synthesis, Automatic Software Testing, Machine Learning, Natural Language Processing

Programming Languages
Java, C#, Scala, C, C++, UML, SQL, XML, Python, JavaScript, HTML, CSS, Assembly

Frameworks and Database Systems
Swing, SWT, Spring, Hibernate, JSP, JUnit, Selenium, Java thread pools, CoreNLP, H2, SQL Server, Apache Derby

Development Tools
Eclipse, NetBeans, Visual Studio, Git, Jenkins, Maven, Selenium, Tomcat, Pex, Trello, Jira (Agile)

Awards and Honors

2010	ACM SIGSOFT Distinguished Paper Award
2007	Selected for Summer Internship at the Information Trust Institute, UIUC, USA
2004-2008	Serbian Ministry of Education Student Scholarship, Serbia

Languages

Serbian	Native language
English	C2 Level
French	B2 Level

Personal Information

31 years old, single, Serbian citizenship, B permit