Research Statement

Feng Chen

Developing effective techniques and methodologies for building reliable and secure complex software systems is the main objective of my research. I am active in various domains related to this goal, including software engineering, programming languages, formal methods and algorithm design. During my Ph.D., I mainly focused on approaches based on program analysis, both dynamic and static, and automated program generation. I co-developed *monitoring oriented programming* (MOP) [TR 2008-1, TACAS 2009, ASE 2008, RV 2008, OOPSLA 2007, TACAS 2005, RV 2005, ICFEM 2004, RV 2003][1], a generic and efficient runtime verification framework, and *predictive runtime analysis* [TR 2008-3, ICSE 2008, CAV 2007, SAS 2006], a technique that detected many unknown concurrency errors in popular open source programs. Together with various collaborators, I have developed several static analysis tools, including, *JavaFAN* [CAV 2004], a static analysis framework for multithreaded Java programs and *CUnit* [RULE 2008, ASE 2003, RTA 2003], a domain policy checker for C. In addition to error detection, I have also been interested in property mining, or inference, leading to the development of two tools: *SpecMeister* [ICFEM 2006], which statically infers concise and comprehensive interface constraints for .Net programs using symbolic execution, and *jMiner* [TR 2008-2], which mines automata-based specifications from observed executions of Java programs. All these successful experiences strengthen my belief that program analysis can be very practical, provided appropriate automatic tool support. In my future research, I intend to investigate novel approaches that leverage the strengths of both static analysis and dynamic analysis instead of making use of either alone, as well as to explore other techniques for software development, such as testing and debugging. I next summarize the major contributions of my research and then briefly elaborate my plan for future research.

Monitoring Oriented Programming. Monitoring executions of programs against requirements has proven to be effective in improving reliability and security of software systems. Numerous monitoring-based techniques have been proposed, most of which share fundamental ideas and techniques even though they aim at different application domains. Based on this important observation, together with Grigore Roşu, I have developed Monitoring Oriented Programming (MOP), a generic framework for applying monitoring in software development. MOP provides support for commonly-shared, foundational mechanisms needed for monitoring-based techniques, including program instrumentation and handling of parameters (i.e., free variables bound at runtime) [TR-2008-1, TACAS 2009, OOPSLA 2007]. It also supports an extensible formalism framework that allows one to define new formalisms to specify desired properties [ICFEM 2004]. This way, MOP can be easily instantiated to a specific application domain. In addition to its genericity, MOP facilitates development of efficient monitoring tools thanks to its separation-of-concerns nature. We have developed an MOP environment for Java, JavaMOP [TACAS 2005, RV 2005], based on which several specialized monitoring tools have been implemented to support different specification formalisms for different applications [ASE 2008, RV 2008, RV 2003], including Linear Temporal Logic, Extended Regular Expressions and Context-Free Grammars. Evaluation showed that MOP not only significantly reduces the effort to develop new monitoring systems but also allows generating efficient monitoring code: MOP-based tools generate more efficient monitoring code than other approaches hardwired with particular formalisms. MOP can also be used for more than property checking. For example, based on the parameter handling mechanism in MOP, I am developing a specification mining tool, jMiner, which efficiently and effectively mines parametric specifications

[1]The referred papers can be found in my CV or my webpage at http://fsl.cs.uiuc.edu/ fengchen.

from observed program executions [TR-2008-2]. I plan to continue in this direction in the future.

Predictive Runtime Analysis. Concurrent programs are highly challenging to test and analyze. Together with Grigore Roşu and Traian Florin Şerbănuţă, I have developed a technique called predictive runtime analysis [TR-2008-3, ICSE 2008, CAV 2007, SAS 2006], which provides an effective solution to detect potential concurrency errors by combining dynamic analysis and static analysis. It is based on *sliced causality* [CAV 2007], a causal partial relation among observed runtime events, computed by drastically but soundly slicing Lamport's happen-before causality using semantic information about the program obtained with static analysis. Evaluation of jPredictor, a predictive runtime analysis tool for Java programs, was very encouraging: we found all the previously known bugs as well as many unknown errors in nontrivial open source systems, e.g., Apache Webserver and Java Util libraries. Sliced causality can also be applied to improve concurrent testing and to reduce the complexity of verifying concurrent programs, which will be part of my future work.

Static Analysis. Together with Azadeh Farzan, Jose Meseguer and Grigore Rosu, I have defined a complete semantic specification of the Java language using Maude, a rewriting logic system, based on which a static analysis tool for multi-threaded Java programs, named JavaFAN, was implemented [CAV 2004]. Our evaluation showed that JavaFAN performs comparably with other Java model checkers, e.g., Java PathExplorer. A similar approach was adopted to implement a domain-policy checker for C programs (a joint work with Mark Hills and Grigore Rosu): we first defined a symbolic semantic specification of C using Maude and then extended it to obtain the policy checker [RULE 2008, ASE 2003, RTA 2003]. I also developed an approach to statically infer comprehensive and concise interface specifications for .Net programs using symbolic execution [ICFEM 2006], together with Wolfram Schulte and Nikolai Tillman.

Future Work. I believe that more effective program analysis approaches can be achieved by combining static analysis, which provides strong guaranties about correctness but usually does not scale well or produces false alarms, and runtime analysis, which is scalable and precise but hard to cover all possible program behaviors. For example, static analysis can be used to detect potentially unsafe points in a program and then runtime monitoring can be used to guard these critical points. This way, the safety of the program is guaranteed without causing unnecessary monitoring overhead. I have started some preliminary attempts in this direction, including a static analysis technique for reducing monitoring overhead (with Eric Bodden and Grigore Rosu) and verification of a complex distributed system by combining static verification and runtime checking and recovery (with Patrick Meredith, Kyoung-Dae Kim, Shobha Vasudevan and Grigore Rosu). I am also interested in other techniques for developing reliable complex software, such as testing and specification mining. For example, together with Traian Serbanuta, Steven Lauterburg, Darko Marinov and Grigore Rosu, I am designing and implementing a concurrent testing tool that allows for writing and executing concurrent tests for specific thread schedulings. It will be used as a foundation for more advanced concurrent testing approaches, e.g., regression testing and concurrent test generation.

In summary, I have been working and will continue my endeavor on developing automated approaches to provide effective solutions for building reliable and secure complex software systems. My research has produced substantial outcomes in both theoretical foundations and practical applications. I have gained abundant academic experience during my Ph.D. study, getting involved in preparing proposals for various funding sources (e.g., NSF, NASA, and Microsoft), leading research projects, and mentoring junior graduate students. All the knowledge, experience, and achievements that I have accumulated constitute a solid foundation for my future academic career and give me confidence in accomplishing the objectives of my career.