| REPORT DOCUMENTATION PAGE | Form Approved OMB No. 0704-0188 |
|---|---|



| 1. REPORT DATE (DD-MM-YYYY) | 2. REPORT TYPE | 3. DATES COVERED (From - To) |
|---|---|---|
| 23-12-2010 | Final Report | 15-Mar-2009 - 30-Sep-2010 |

**4. TITLE AND SUBTITLE**
Invariant Rules for Software Producibility and Assurance

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**
N00014-09-1-0651

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**
Liu, Yanhong (Annie)
Stoller, Scott D.

**5d. PROJECT NUMBER**

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Research Foundation of the State University of New York
Stony Brook, NY 11794-3362

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Office of Naval Research
One Liberty Center
875 North Randolph Street, Suite 1425
Arlington, VA 22203-1995

**10. SPONSOR/MONITOR'S ACRONYM(S)**
ONR

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION/AVAILABILITY STATEMENT**



**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**
This project develops a unified framework for rigorously capturing and applying software design and development knowledge to significantly improve software producibility and assurance under complex and challenging requirements facing Navy software for cyberspace. The framework is based on invariants, which underly all requirements about dependencies, concurrency, distribution, fault-tolerance, security, and general safety and correctness as well as cost and efficiency conditions.
Invariant rules are used to declaratively specify how complex invariants are maintained under all possible updates to system states. The design and development knowledge captured by invariant rules underlies not only invariant maintenance for design and optimization, but also invariant verification for validation and assurance, as well as general transformations for instrumentation, refactoring, etc. We especially investigate the use of invariant rules for specifying critical aspects of complex systems, such as in web frameworks and mashups, that may involve concurrency, distribution, trust and security.

**15. SUBJECT TERMS**
invariants, program transformation, program optimization, program analysis, runtime checking, rule languages, complexity guarantees, computer security, distributed systems

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | 19b. TELEPHONE NUMBER (Include area code) |

Final Project Report to Office of Naval Research

Invariant Rules for Software Producibility and Assurance

PI: Yanhong (Annie) Liu
Co-PI: Scott D. Stoller
PI Institution: State University of New York at Stony Brook
PI E-mail: liu@cs.sunysb.edu
Grant Number: N000140910651
Period: 3/15/09-9/30/10
Amount: $282,719

---

Project Attributes.

* Number of refereed papers/book chapters published: 10
* Number of unrefereed reports and other articles: 4
* Number of project presentations: 16
* Number of post-doc supported: 1, at 20-40% of full time
* Number of graduate students supported: 3, at 10-100% of full time

---

Abstract.

This project develops a unified framework for rigorously capturing and applying software design and development knowledge to significantly improve software producibility and assurance under complex and challenging requirements facing Navy software for cyberspace. The framework is based on the notion of invariants that are essential in all software systems, because invariants underly all requirements about dependencies, concurrency, distribution, fault-tolerance, security, and general safety and correctness as well as cost and efficiency conditions.

Invariant rules are used to declaratively specify how complex invariants are maintained under all possible updates to system states. The design and development knowledge captured by invariant rules underlies not only invariant maintenance for design and optimization, but also invariant verification for validation and assurance, as well as general transformations for instrumentation, refactoring, etc. We especially investigate the use of invariant rules for specifying critical aspects of complex systems, such as in web frameworks and mashups, that may involve concurrency, distribution, trust and security.

---

Summary of Technical Progress.

Contents

1. We designed an invariant rule language and framework and developed new analysis and transformations for using invariants to significantly increase software producibility and assurance. Invariant rules declaratively specify how complex invariants in software systems are incrementally maintained under all possible updates to system states. These invariants underly all requirements about dependencies, concurrency, distribution, security, and general safety and correctness as well as cost and efficiency conditions. Invariant-driven transformations include not only invariant maintenance for design and optimization, but also invariant verification for validation and assurance, as well as general transformations for instrumentation, refactoring, etc.

1.1. We created a powerful language and framework that allow coordinated transformations driven by invariants to be specified declaratively, as invariant rules, and applied automatically. We also developed prototypes for transforming Python and C programs, and successfully used the prototypes in a variety of applications: generating efficient implementations from clear and modular specifications by incrementalization; instrumenting programs for runtime verification, profiling, and debugging; and code refactoring.

> Y. A. Liu, M. Gorbovitski, and S. D. Stoller. A language and framework for invariant-driven transformations. In Proceedings of the 8th International Conference on Generative Programming and Component Engineering, pages 55-64, Denver, Colorado, October 2009. ACM Press.
> http://www.cs.sunysb.edu/~liu/papers/Inv-GPCE09.pdf

1.2. We developed an alias analysis for a full dynamic object-oriented language, and used it for applying invariant rules for incrementalization and specialization. The analysis is flow-sensitive; uses precise type analysis and trace sensitivity, a powerful form of context sensitivity; and uses a compressed representation to drastically reduce space usage. Despite the challenge of the problem, experiments show that our analysis has good precision and efficiency and represents the best trade-off among 18 variations of the analysis.

> M. Gorbovitski, Y. A. Liu, S. D. Stoller, T. Rothamel, and K.T. Tekle. Alias analysis for optimization of dynamic languages. In Proceedings of the 6th Symposium on Dynamic Languages, pages 27-42, Reno, Nevada, October 2010. ACM Press.
> http://www.cs.sunysb.edu/~liu/papers/Alias-DLS10.pdf

1.3. We developed invariant rules for instrumenting and incrementalizing real applications, and a method for composing the rules to improve both the transformed programs and the application of the rules. The example instrumentation is for ranking peers in BitTorrent. The example incrementalizations are for optimizing the instrumentation of BitTorrent, for efficiently computing the quality of network hosts' connections using NetFlow, and for generating efficient implementations from formal specifications for Constrained RBAC.

> M. Gorbovitski, Y. A. Liu, S. D. Stoller, and T. Rothamel. Composing transformations for instrumentation and incrementalization of real applications. Technical Report DAR 10-48, Computer Science Department, SUNY Stony Brook, May 2010.
> http://www.cs.sunysb.edu/~liu/papers/Compose-TR10.pdf

1.4. We developed a method for statically transforming complex queries over objects and sets into efficient incremental implementations. The queries may involve objects that are arbitrarily aliased and sets that are arbitrarily nested. The method handles any update to the objects and sets that are queried over. The generated implementations use sophisticated auxiliary data structures and incrementally maintain them for efficient indexing. Previous work either was not fully automatic or

could not handle as general forms of queries, produce as efficient implementations, and give as strong static guarantees.

J. Brandvein and Y. A. Liu. Compiling object-set queries for demanded incremental computation. Technical Report in Preparation, Computer Science Department, SUNY Stony Brook, December 2010.

2. We developed automatic methods for complexity analysis and efficient implementations of logic query languages and applied them to a variety of applications.

2.1. We extended and refined our method for generating optimal implementations from Datalog rules and added experimental evaluations that confirmed the analyzed time and space complexities. The running time is optimal in that only useful combinations of facts that lead to all hypotheses of a rule being simultaneously true are considered, and each such combination is considered exactly once in constant time. We also added new applications to several analysis problems, some with improved algorithm complexities and all with greatly improved algorithm understanding and greatly simplified complexity analysis.

Y. A. Liu and S.D. Stoller. From Datalog rules to efficient programs with time and space guarantees. ACM Transactions on Programming Languages and Systems, 31(6):1-38, August 2009. http://www.cs.sunysb.edu/~liu/papers/Rules-TOPLAS09.pdf

2.2. We developed precise time and space complexity analysis for efficiently answering Datalog queries, and precise relationships between top-down evaluation with tabling and bottom-up evaluation driven by demand. We also developed a method for transforming the rules for efficient bottom-up evaluation; the method is simpler than magic set transformation and produces simpler rules that yield exponentially smaller space in the number of arguments of predicates. Experiments on benchmarks from OpenRuleBench support our results.

K.T. Tekle and Y. A. Liu. Precise complexity analysis for efficient Datalog queries. In Proceedings of the 12th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming, pages 35-44, Hagenberg, Austria, July 2010. http://www.cs.sunysb.edu/~liu/papers/RuleQuery-PPDP10.pdf

2.3. We developed a novel combination of transformations for generating efficient implementations for a powerful graph query language. Our method combines transformation to Datalog, recursion conversion, demand transformation, and specialization, and finally generates efficient programs with precise complexity guarantees. It improves an O(VE) time complexity factor using previous methods to O(E), where V and E are the numbers of vertices and edges, respectively. Our experiments confirm the analyzed complexities.

K. T. Tekle, M. Gorbovitski, and Y. A. Liu. Graph queries through Datalog optimizations. In Proceedings of the 12th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming}, pages 25-34, Hagenberg, Austria, July 2010. http://www.cs.sunysb.edu/~liu/papers/ImplGraphQL-PPDP10.pdf

2.4. We developed automatic analysis of role activation and deactivation rules in trust management policies. The analysis checks important consistency and completeness criteria. When violations are detected, the analysis automatically detects possible causes of the violation and determines possible remedies and simplifications to the policies. We applied the analysis to a large trust management policy for a proposed national Electronic Health Record (EHR) service. It found a number of violations and determined possible remedies and simplifications.

Y. A. Liu, K. T. Tekle, and S. D. Stoller. Role activation

analysis for trust management policies. Technical Report in Preparation, Computer Science Department, SUNY Stony Brook, 2011.

3. We developed powerful compilation and verification methods that heavily exploit invariants for optimizing distributed algorithms and for verifying concurrent algorithms.

3.1. We created a simple and powerful language for programming distributed algorithms, a compilation method to generate their implementations, and powerful optimizations to incrementalize expensive synchronization conditions and remove unnecessary messages. The language can express distributed algorithms cleanly, almost like pseudo code descriptions, free of implementation details, but with a precise semantics for execution. We programmed a variety of distributed algorithms, and successfully generated their implementations and optimized implementations.

Y. A. Liu, B. Lin, and S. D. Stoller. Programming and optimizing distributed algorithms. Technical Report in Preparation, Computer Science Department, SUNY Stony Brook, December 2010.

3.2. We developed a new method to automatically check linearizability based on refinement relations from abstract specifications to concrete implementations. The method exploits model checking of finite state systems specified as concurrent processes with shared variables, and uses partial order reduction to reduce search space. Our tool automatically checked a variety of concurrent algorithms, including the first algorithms for the mailbox problem and scalable NonZero indicators, and was able to find all known and injected bugs.

Y. Liu, W. Chen, Y. A. Liu, and J. Sun. Model checking linearizability via refinement. In Proceedings of the 16th International Symposium on Formal Methods, pages 321-337, Eindhoven, The Netherlands, November 2009. Springer. http://www.cs.sunysb.edu/~liu/papers/MCLinear-FM09.pdf

3.3. We carried out a formal verification of a new concurrent data structure, Scalable NonZero Indicators. The algorithm supports incrementing, decrementing, and querying the shared counter efficiently without blocking. It is highly non-trivial and its correctness is challenging to prove. We proved that it satisfies linearizability, by showing a trace refinement relation from the concrete implementation to its abstract specification. These models are specified in CSP and verified automatically using the model checking toolkit PAT.

S. J. Zhang, Y. Liu, J. Sun, J. S. Dong, W. Chen, and Y. A. Liu. Formal verification of scalable nonzero indicators. In Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering, pages 406-411, Boston, Massachusetts, July 2009.
http://www.cs.sunysb.edu/~liu/papers/SNIZ-SEKE09.pdf

4. We developed new methods and frameworks for analyzing security policies and helping increase software assurance.

S. D. Stoller, P. Yang, M. Gofman, and C. R. Ramakrishnan. Symbolic Reachability Analysis for Parameterized Administrative Role Based Access Control. Computers & Security, 2010. In Press.
http://www.cs.sunysb.edu/~stoller/papers/symbolic-reachability-journal.pdf

S. D. Stoller. Trust Management for Web Services. In Proceedings of the 6th International Conference on Network and Service Management, pages 262-265. IEEE Press, 2010.
http://www.cs.sunysb.edu/~stoller/papers/cnsm-2010.pdf

J. Seyster, K. Dixit, X. Huang, R. Grosu, K. Havelund, S. A. Smolka, S. D. Stoller, and E. Zadok. Aspect-Oriented

Instrumentation with GCC. In Proceedings of the 1st International Conference on Runtime Verification, volume 6418 of Lecture Notes in Computer Science, pages 405-420. Springer-Verlag, 2010. http://www.cs.sunysb.edu/~stoller/papers/rv2010.pdf

Finally, a book has been basically completed and is being revised. It covers extensive research results that exploit invariants for developing efficient programs with high productivity and assurance.

Y. A. Liu. Systematic Program Design: From Clarity to Efficiency. With offers for publication from MIT Press, Cambridge University Press, Springer, and Higher Education Press. Publication expected 2011.

---

Participation in Other Research Projects.

The following NSF project focused on optimizations across object abstraction that transform clear modular design into sophisticated efficient design based on invariants:

Y. A. Liu, PI (S.D. Stoller, co-PI). Clarity and Efficiency in Design, 2006-2011.

The following industry gift grant focuses on generating incremental implementations from set-based specifications:

Y. A. Liu, PI. Generating Incremental Implementations From Set-Based Specifications. Hengsoft, 2009-2010.

The following NSF project focuses on complexity analysis and powerful optimizations for improving logic rule engines:

Y. A. Liu, co-PI (M. Kifer, PI, and D. Warren, co-PI). Performance Analysis and Optimization for Logic Rule Engines, 2010-2013.

The following AFOSR project focuses on making embedded software more robust through runtime monitoring and recovery based on control theory:

S. D. Stoller, co-PI (S. A. Smolka, PI, and Radu Grosu, Klaus Havelund, and Erez Zadok, co-PIs). Survivable Software. 2009-2011.

The following NSF project focuses on generation and enforcement of low-level access control policies that ensure higher-level system integrity goals:

S. D. Stoller, co-PI (R. Sekar, PI, and C.R. Ramakrishnan, co-PI). Proactive Techniques for Preserving System Integrity: A Basis for Robust Defense Against Malware, 2008-2012.

The following ONR project focuses on generation, analysis, and enforcement of security policies in distributed systems:

S. D. Stoller, PI (R. Sekar and C.R. Ramakrishnan, co-PIs). A Framework for Analyzing and Ensuring Trust in Service-Oriented Architectures, 2007-2012.

---

Transitions and DOD Interactions.

A startup software company, Hengsoft, is working on building the incrementalization method we have developed into their product. During this grant period, they supported our research with a gift grant to Stony Brook University, focusing on a prototype for generating incremental implementations from set-based specifications. Our method improves over the best finite differencing method of Paige by automatically deriving the incremental maintenance code in the rules.

A hot startup company in logic databases, LogicBlox, eagerly hired our most recent Ph.D. graduate, Tuncay Tekle, and tasked hims to build our method for optimizing Datalog rules and queries into their product. Our method cleanly solves the challenging open problems of providing precise time and space complexity analysis for rule-based queries, and achieving drastic optimizations in both time and space.

---

Software and Hardware Prototypes.

1. Prototype Name: InvTS
   + URL: https://secure.mickg.net/darlab/cgi-bin/InvTS/index.cgi
   + Availability: Used in teaching and by interested graduate students. The system is available through a web interface.
   + Description: A system for automatically applying invariant rules for incrementalization and program transformations in general. Through the web interface, a user can run it for Core RBAC and Constrained RBAC implementations or provide other programs or rules for transformations.

2. Prototype Name: RuleQuery
   + URL: will be in XSB, which is at http://xsb.sourceforge.net
   + Availability: Internal use by interested faculty and students. Parts of it will be included in the XSB distribution.
   + Description: A system for precisely analyzing time and space complexities of Datalog queries and for optimizing Datalog queries by source-to-source transformations.

3. Prototype Name: SetInc
   + URL:
   + Availability: Internal and external use by graduate students and Hengsoft developers. The system is available by request.
   + Description: A system for automatically compiling set queries and updates into efficient incremental computations.

---

Honors, Prizes, Awards, or Promotions Received.

. Liu was awarded State University of New York Chancellor's Award for Excellence in Scholarship and Creative Activities, April 2010.

. Stoller was promoted to full professor, effective September 2010.

. Stoller received Outstanding Community Service Award, IEEE Technical Committee on Security and Privacy, 2009.

---

URLs.

List of selected publications available online:

http://www.cs.sunysb.edu/~liu/#publications

Overview of our research projects:

http://www.cs.sunysb.edu/~liu/#projects
http://www.cs.sunysb.edu/~liu/#overview

Advanced graduate courses:

Advanced Programming Languages
http://www.cs.sunysb.edu/~liu/cse626/

Protocol Design and Analysis
http://www.cs.sunysb.edu/~liu/cse592/

---

Post-doc and Ph.D. Students Supported.

1. Name: Jonathan Brandvein, Ph.D. Student
   + US Citizen/Permanent Resident: US Citizen
   + Thesis:
   + Graduated: Expected 2013
   + Job:

2. Name: Tuncay Tekle, Ph.D. Student
   + US Citizen/Permanent Resident:
   + Thesis: Efficient Datalog Queries with Time and Space Guarantees
   + Graduated: December 2010
   + Job: LogicBlox

3. Name: Tom Rothamel, Post-doc
   + US Citizen/Permanent Resident: US Citizen
   + Thesis:
   + Graduated: September 2010
   + Job: Lemahtor, LLC