Summary of
"Analysis of Invariants for Efficient Bounded Verification"

Nikhil Satish Pai Vishal Mishra Nikhil Anand

16th September 2015

Important Keywords:

ii1. **Static analysis:** It's the process of analysing of a source code without actually executing the code.

ii2.**SAT-based code analysis:** It is mostly referred to the application of static code analysis tools to find possible vulnerabilities within a source code by using various techniques like Taint Analysis and Data Flow Analysis. SAT-based bounded verification of programs consists of the translation of the code and its annotations into a propositional formula.

ii3.**Alloy*:**Its a relational modeling language that captures the essence of software abstractions simply and succinctly, using a minimal toolkit of mathematical notions.
The Alloy Analyzer transforms models in which domains' sizes are bounded to a fix scope, into propositions that are later fed to SAT-solvers. Then, given an assertion to be verified in the model, the Alloy Analyzer attempts to produce a model of the specification that violates the assertion. If no such model is found within the provided scopes, we can gain more confidence that the analyzed property holds in the model.

ii4.**KodKod**:**unlike Alloy Analyzer, KodKod is suitable as a generic relational engine. Kodkod outperforms the Analyzer dramatically on problems involving partial instances. The underlying technology involves translation from relational to boolean logic, and the application of an off-the-shelf SAT solver on the resulting boolean formula.

ii5.**DynAlloy*****:**DynAlloy is an extension of the Alloy specification language that allows one to specify and analyze dynamic properties of models and allow us to cope with the lack of dynamics of Alloy. The analysis is supported by the DynAlloy Analyzer tool.

##References for Alloy,KodKod and DynAlloy

*Jackson, D., Software Abstractions. MIT Press, 2006.
**Torlak E., Jackson, D., Kodkod: A Relational Model Finder. in TACAS '07, LNCS 4425, pp. 632–647.
***Galeotti, J. P., Frias, M. F., DynAlloy as a Formal Method for the Analysis of Java Programs, in Proceedings of IFIP Working Conference on Software Engineering Techniques, Warsaw, 2006, Springer.

Brief notes:

iii1 **Motivation:**
In a Boolean satisfiability based bounded verification technique, the given source code along with any annotations is mapped to a propositional formula and this is given as an input to an SAT solver to check for any violations. Determining violations in code which has data structures with links like Linked list, trees etc using current methodologies is not extensive and does not cover all possible cases. The paper uses a tool TACO to perform automated testing of code which contains these data structures.

iii2 **Related Work :** Khurshid et al. * introduced the Alloy Annotation Language to propose a translation similar to this paper. Vaziri et al. ** propose a set of rules to be applied along the translation to a SAT-formula in order to profit from properties of functional relations. Saturn *** is also a SAT-based static analysis tool for C. It uses as its main techniques a slicing algorithm and function summaries.

iii3 **Baseline results :**
The data structures analyzed were singly linked list, Java collection abstract linked list, node caching linked list, tree set and standard implementations of binomial heap and AVL tree.
The analysis was done for two different output goals and the run times for each model was compared.
1. Analysis of Bug Free code:
Checks were done to verify that for all the data structures the corresponding invariants were preserved for e.g. that the singly linked list remains acyclic. The performance of TACO was compared with JForge and TACO$^-$(TACO without symmetry reduction algorithms).

2. Bug detection
The remove or extract operations of the respective data structures were compared with TACO$^-$, JForge, ESC/Java2, Java PathFinder, Jacob and Kaisan.

iii4 **Future work :** The paper provides for an efficient computation of bounds with an improved analysis time. However, the authors believe the article provides a naive way of computing bounds and there is a further scope for future work to make the bound computation efficient.

* Khurshid, S., Marinov, D., Jackson, D., An analyzable annotation language. In OOPSLA 2002, pp. 231-245
** Vaziri, M., Jackson, D., Checking Properties of Heap-Manipulating Procedures with a Constraint Solver, in TACAS 2003, pp. 505-520
*** Xie, Y., Aiken, A., Saturn: A scalable framework for error detection using Boolean satisfiability. in ACM TOPLAS, 29(3): (2007).

Area of Improvement:

iv1. The paper says that TACO consumes less memory than during translation to a propositional formula and during SAT-solving than TACO− and JForge. However, there are no experimental results to substantiate this.

iv2. The paper considers an object oriented language namely Java to develop the testing tool, further investigations can be done on testing of procedural languages like C using the same techniques.

Relation to Original paper : The AVLTree, TreeSet, NodeCachingLL and the SinglyLinkedList data structures used in the original paper were generated by referring to this paper.

Reference to the Paper:

Juan Pablo Galeotti, Nicolas Rosner, Carlos G. L. Pombo, and Marcelo F. Frias. 2010. Analysis of invariants for efficient bounded verification. In Proceedings of the ACM International Symposium on Software Testing and Analysis (ISSTA'10). 25–36.

Reference:

1.Chalin P., Kiniry J.R., Leavens G.T., Poll E. Beyond Assertions: Advanced Specification and Verification with JML and ESC/Java2. FMCO 2005: 342-363.

2.Bouillaguet Ch., Kuncak V., Wies T., Zee K., Rinard M.C., Using First-Order Theorem Provers in the Jahob Data Structure Verification System. VMCAI 2007, pp. 74–88

3.Deng, X., Robby, Hatcliff, J., Towards A Case-Optimal Symbolic Execution Algorithm for Analyzing Strong Properties of Object-Oriented Programs, in SEFM 2007, pp. 273-282.