Software Model Checking at Design and Implementation *

Fang Yu

Advisor: Tevfik Bultan
Department of Computer Science
University of California, Santa Barbara
November 9, 2006

Abstract

This work is aimed to investigate main techniques of the cutting-edge model/program/design checkers, such that scalable software model checking may be achieved in future research.

1 Introduction

Software model checking is one of the essential techniques to guarantee system correctness. Being a witness of a shift from the verification of abstract hand-built models of codes, towards the direct verification of implementation level code, in this survey we present the main techniques in model checking, as well as a broad survey of modern tools. We aim to achieve scalable software model checking in future research based on this study.

We propose three survey directions: a) model checking, b) program verification, and c) software modeling. We first go through main milestones of model checking in past twenty years, which include symbolic model checking [BCL90, BCMDH90, GP00], abstraction [CGL92], symmetry reduction [ES94, ES97], partial order reduction [CGMP99, KLMPY98], bounded model checking [AKMM03, BCCFZ99, CBRZ01, CGRPST02], induction [MRS03, SSS00], interpolation [McMillan03], predicate abstraction [HJMK04, HJMS02] and refinement [BCDR04]. Methods successfully addressed these issues did expand the ability of model checking and had been widely adopted in model checkers, such as SPIN [Holzmann97], NuSMV [CCGRPST02], and ALV [YBB05], as well as modern program checkers.

*This is the reading list for Fang’s Major Area Exam.
While model checkers verify general properties in specified languages, program checkers usually verify specific properties of general languages. At the second part, we broadly survey program checkers: JPF [BHPV00], Bandera [CD00], MOPS [CD02, CDW04], Bebop [BR01], Blast [BMMR01, HJM04, HJMQ03], CMC [MPCED02], CBMC [CKY03, CKL04], VeriSoft [Godefroid97], eXplode [YTEM04, YSE06], and WebSSARI [HYHTLK04]. These checkers differ in their algorithms, target language and verified properties. We will focus on the core algorithms and pros/cons of these checkers.

We then delve into software modeling to seek an efficient framework to facilitate scalable model checking. We first look at general modeling languages: JML [LBR06, BCCEKLRLP05], UML [OMG99]/OCL [OCL99, WK98], IOA [GL98], STATEMATE [Harel90], MSC [MSC96, HT03]. These modeling languages are used to formally specify system requirements before code generation/implementation. We believe that ensuring that designs are robust and free from conceptual flaws forms a solid foundation of software system development. Previous researches addressing the correctness of design languages include a) Alur’s work [AY99] for MSC, b) Bogar [DHHRRW06, RDH06, RRD04, RRDH04] for JML, c) Aloca for Alloy [DCJ06, JSS01, Jackson02, Jackson06] d) Model checking for UML state machines [LM99, SK01] and USE [GBR03] for UML/OCL.

For future research, we aim to take advantage on design language to achieve scalable code-level checking.

2 Model Checking

Reading List:


3 Program Verification

Reading List:

BMMR01 T. Ball, R. Majumdar, T. Millstein, S.K. Rajamani, Automatic Predicate Abstrac-

BR00 Thomas Ball, Sriram K. Rajamani, Bebop: A Symbolic Model Checker for Boolean
Programs, SPIN 2000 Workshop on Model Checking of Software, LNCS 1885, Au-
gust/September 2000, pp. 113-130.

CD00 J.C. Cobett, M.B. Dwyer, et al. Bandera: Extracting finite state models from Java
source code, Proc. 22nd Int. Conf. on Software Engineering (ICSE00), pp.439-448.

CD02 Hao Chen and David Wagner. MOPS: an Infrastructure for Examining Security

CDW04 Hao Chen, Drew Dean, and David Wagner, Model Checking One Million Lines of

CGPL06 Cristian Cadar, Vijay Ganesh, Peter M. Pawlowski, David L. Dill, Dawson R. En-
gler. EXE: Automatically Generating Inputs of Death, In Proc. of the 13th ACM
Conference on Computer

CKY03 E. Clarke, D. Kroening, K. Yorav, “Behavioral Consistency of C and Verilog Pro-
grams using Bounded Model Checking.” In Proc. of the 40th Design Automation

Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004),
LNCS 2988, pp. 168-176.

CSD02 Chandrasekhar Boyapati, Sarfraz Khursheid and Darko Marinov. Korat: Automated
Testing Based on Java Predicates. ACM/SIGSOFT International Symposium on

Godefroid97 P. Godefroid, VeriSoft: A Tool for the Automatic Analysis of Concurrent Reactive
Software. Proc. of the 9th Conference on Computer Aided Verification, LNCS 1254,

HJM04 Thomas A. Henzinger, Ranjit Jhala, and Rupak Majumdar. Race checking by

HJMS02 Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Gregoire Sutre. Lazy
Abstraction. Proceedings of the 29th Annual Symposium on Principles of Program-
4 Software Modeling and Analysis:

Reading List:


BBKRT04 Michael Balser, Simon Baumler, Alexander Knapp, Wolfgang Reif, Andreas Thums. Interactive Verification of UML State Machines. ICFEM 2004. pp. 434-448


5 Conclusion and Future Research