

Yulei Sui

Practicing the Art of Static Analysis

December 27, 2022

Springer Nature

Contents

Part I Aim & Scope

1	Introduction to Static Analysis	1
1.1	Program Analysis and Static Analysis	1
1.2	Applications of Static Analysis	2
1.2.1	Compiler Optimizations	2
1.2.2	Vulnerability Detection	4
1.2.3	Code Embedding	6
1.2.4	Many Other Program Analyses	6
2	Introduction to Intermediate Representation	7
2.1	LLVM	7
2.1.1	LLVM Compiler	7
2.1.2	LLVM Intermediate Representation (IR)	8
2.1.3	A Compilation Example	9
2.2	SVF	10
2.2.1	SVF Architecture	13
2.2.2	SVF IR and Code Graphs	15
3	Control and Data Dependence	19
3.1	Control Dependence	20
3.1.1	ICFG	21
3.1.2	Control-Flow Reachability	21
3.1.3	Context-Sensitive Control-Flow Reachability	27
3.2	Data Dependence	28
3.2.1	Pointer Analysis	29
3.2.2	Andersen’s Inclusion-based Analysis	32
3.2.3	Information Flow Tracking	41

4	Static Value-Flows	47
4.1	Value-Flows	47
4.2	Memory SSA Form	47
4.2.1	Intraprocedural and Interprocedural Memory SSA Forms ...	47
4.2.2	Interprocedural Side-Effect Analysis	48
4.2.3	Memory Region Generation	49
4.3	Sparse Value-Flows	51
4.4	Value-Flows for Multithreaded Programs	53
4.4.1	Static Thread Model	53
4.4.2	Computing Thread-Oblivious Def-Use Chains	55
4.4.3	Computing Thread-Aware Def-Use Chains	56
4.4.3.1	Interleaving Analysis	56
4.4.3.2	Value-Flow Analysis	58
4.4.3.3	Lock Analysis	58
4.4.4	Sparse Analysis	59
5	Pointer Alias Analysis	61
5.1	Field- and Array-Sensitive Pointer Analysis	62
5.2	Flow-Sensitive Pointer Analysis	66
5.3	Context-Sensitive Pointer Analysis	70
5.4	Points-to Data Structures	75
5.4.1	Compacting Points-To Sets	75
5.4.1.1	Representing Points-To Sets as Bit-Vectors	78
5.4.1.2	Contiguous and Sparse Bit-Vectors	78
5.4.1.3	Core Bit-Vector	81
5.4.1.4	Compacting Points-To Sets	82
5.4.1.5	Integer Programming Formulation	83
5.4.1.6	Hierarchical Clustering	85
5.4.1.7	Linkage Criteria	86
5.4.1.8	Clustering Objects	88
5.4.1.9	More Efficient Region-Based Clustering	89
5.4.1.10	Word-Aligned Identifier Mapping	91
5.4.2	Hash Consed Points-To Sets	93
5.4.2.1	Introduction	93
5.4.2.2	Motivating Examples	96
6	Context-Free Language Reachability	99
6.1	Edge-Labeled Graphs and Context-Free Grammars	99
6.2	All-Pair Analysis and Demand-Driven Analysis	99
6.3	Graph Compaction and Transitive Redundancy	99
7	Software Verification and Symbolic Execution	101
7.1	Code Verification	101
7.2	Predicate Logic	108
7.3	Software Verification Using Theorem Prover	115

7.4	Assertion-based Verification Using Symbolic Execution	124
8	Abstract Interpretation	131
8.1	Concrete and Abstract Domains	131
8.1.1	Non-Relational Abstract Domains	136
8.1.2	Relational Abstract Domains	138
8.2	Abstract State and Abstract Execution	138
8.2.1	Abstract Execution over Multiple Domains	143
8.2.2	Abstract Execution over Memory Address and Interval Domains	149
8.3	Sparse Abstract Execution	162
8.4	Weak Topological Ordering	163
8.5	Data-Flow Analysis	164
8.5.1	Maximal Fix Point (MFP)	164
8.5.2	Meet Over Feasible Paths	164
8.5.3	Meet Over All Paths (MoP)	164
8.6	Typestate Analysis	164
9	Machine Learning for Static Analysis	165
9.1	Learning-Based Static Analysis	165
9.2	Vulnerability Detection using Deep Graph Neural Networks	165
10	Static Analysis for Machine Learning	167
10.1	Deep Neural Networks	168
10.2	DNN Robustness Issues	169
10.3	Verification for Deep Neural Networks	170
10.4	SOTA Tools	181
	Appendix	183
A.1	Online Materials	183
A.1.1	Open Courses	183
	Glossary	185
	Solutions	187
	Index	189
	References	190