

Contents

1	Introduction	1
1.1	Background	1
1.2	Applications of Software Similarity and Classification	2
1.3	Motivation	2
1.4	Problem Formulation	3
1.5	Problem Overview	4
1.6	Aims and Scope	5
1.7	Book Organization	5
	References	6
2	Taxonomy of Program Features	7
2.1	Syntactic Features	7
2.1.1	Raw Code	7
2.1.2	Abstract Syntax Trees	8
2.1.3	Variables	8
2.1.4	Pointers	9
2.1.5	Instructions	9
2.1.6	Basic Blocks	10
2.1.7	Procedures	10
2.1.8	Control Flow Graphs	11
2.1.9	Call Graphs	12
2.1.10	Object Inheritances and Dependencies	12
2.2	Semantic Features	12
2.2.1	API Calls	12
2.2.2	Data Flow	12
2.2.3	Procedure Dependence Graphs	13
2.2.4	System Dependence Graph	13
2.3	Taxonomy of Features in Program Binaries	13
2.3.1	Object File Formats	13
2.3.2	Headers	13

2.3.3	Object Code	14
2.3.4	Symbols	14
2.3.5	Debugging Information	14
2.3.6	Relocations	14
2.3.7	Dynamic Linking Information	14
2.4	Case Studies	14
2.4.1	Portable Executable	14
2.4.2	Executable and Linking Format	15
2.4.3	Java Class File	16
	References	16
3	Program Transformations and Obfuscations	17
3.1	Compiler Optimisation and Recompilation.	17
3.1.1	Instruction Reordering.	18
3.1.2	Loop Invariant Code Motion	18
3.1.3	Code Fusion	18
3.1.4	Function Inlining	18
3.1.5	Loop Unrolling	18
3.1.6	Branch/Loop Inversion	18
3.1.7	Strength Reduction	19
3.1.8	Algebraic Identities.	19
3.1.9	Register Reassignment	19
3.2	Program Obfuscation	19
3.3	Plagiarism, Software Theft, and Derivative Works	19
3.3.1	Semantic Changes	20
3.3.2	Code Insertion	20
3.3.3	Code Deletion	20
3.3.4	Code Substitution	20
3.3.5	Code Transposition	21
3.4	Malware Packing, Polymorphism, and Metamorphism	21
3.4.1	Dead Code Insertion	21
3.4.2	Instruction Substitution	22
3.4.3	Variable Renaming	22
3.4.4	Code Reordering	22
3.4.5	Branch Obfuscation	23
3.4.6	Branch Inversion and Flipping	23
3.4.7	Opaque Predicate Insertion	24
3.4.8	Malware Obfuscation Using Code Packing	24
3.4.9	Traditional Code Packing	25
3.4.10	Shifting Decode Frame	25
3.4.11	Instruction Virtualization and Malware Emulators	26
3.5	Features under Program Transformations.	27
	References	27

4	Formal Methods of Program Analysis	29
4.1	Static Feature Extraction	29
4.2	Formal Syntax and Lexical Analysis	30
4.3	Parsing	30
4.4	Intermediate Representations	31
4.4.1	Intermediate Code Generation	31
4.4.2	Abstract Machines	31
4.4.3	Basic Blocks	32
4.4.4	Control Flow Graph	32
4.4.5	Call Graph	32
4.5	Formal Semantics of Programming Languages	32
4.5.1	Operational Semantics	33
4.5.2	Denotational Semantics	33
4.5.3	Axiomatic Semantics	33
4.6	Theorem Proving	33
4.6.1	Hoare Logic	33
4.6.2	Predicate Transformer Semantics	34
4.6.3	Symbolic Execution	34
4.7	Model Checking	34
4.8	Data Flow Analysis	34
4.8.1	Partially Ordered Sets	35
4.8.2	Lattices	35
4.8.3	Monotone Functions and Fixed Points	35
4.8.4	Fixed Point Solutions to Monotone Functions	36
4.8.5	Dataflow Equations	36
4.8.6	Dataflow Analysis Examples	36
4.8.7	Reaching Definitions	37
4.8.8	Live Variables	37
4.8.9	Available Expressions	37
4.8.10	Very Busy Expressions	38
4.8.11	Classification of Dataflow Analyses	38
4.9	Abstract Interpretation	38
4.9.1	Widening and Narrowing	38
4.10	Intermediate Code Optimisation	38
4.11	Research Opportunities	39
	References	39
5	Static Analysis of Binaries	41
5.1	Disassembly	41
5.2	Intermediate Code Generation	42
5.3	Procedure Identification	43
5.4	Procedure Disassembly	44
5.5	Control Flow Analysis, Deobfuscation and Reconstruction	44
5.6	Pointer Analysis	44

5.7	Decompilation of Binaries	45
5.7.1	Condition Code Elimination.	45
5.7.2	Stack Variable Reconstruction	45
5.7.3	Preserved Register Detection	46
5.7.4	Procedure Parameter Reconstruction	46
5.7.5	Reconstruction of Structured Control Flow	47
5.7.6	Type Reconstruction	48
5.8	Obfuscation and Limits to Static Analysis	48
5.9	Research Opportunities	48
	References	48
6	Dynamic Analysis	51
6.1	Relationship to Static Analysis	51
6.2	Environments	52
6.3	Debugging	52
6.4	Hooking	52
6.5	Dynamic Binary Instrumentation	53
6.6	Virtualization	53
6.7	Application Level Emulation	53
6.8	Whole System Emulation	54
	References	55
7	Feature Extraction	57
7.1	Processing Program Features	57
7.2	Strings	58
7.3	Vectors	58
7.4	Sets.	58
7.5	Sets of Vectors.	58
7.6	Trees.	59
7.7	Graphs	59
7.8	Embeddings	59
7.9	Kernels	61
7.10	Research Opportunities	61
	References	61
8	Software Birthmark Similarity	63
8.1	Distance Metrics.	63
8.2	String Similarity.	64
8.2.1	Levenshtein Distance	64
8.2.2	Smith-Waterman Algorithm.	64
8.2.3	Longest Common Subsequence (LCS)	65
8.2.4	Normalized Compression Distance	65

8.3	Vector Similarity	66
8.3.1	Euclidean Distance	66
8.3.2	Manhattan Distance	66
8.3.3	Cosine Similarity	66
8.4	Set Similarity	67
8.4.1	Dice Coefficient	67
8.4.2	Jaccard Index	67
8.4.3	Jaccard Distance	67
8.4.4	Containment	68
8.4.5	Overlap Coefficient	68
8.4.6	Tversky Index	68
8.5	Set of Vectors Similarity	68
8.6	Tree Similarity	69
8.7	Graph Similarity	69
8.7.1	Graph Isomorphism	69
8.7.2	Graph Edit Distance	69
8.7.3	Maximum Common Subgraph	70
	References	70
9	Software Similarity Searching and Classification	71
9.1	Instance-Based Learning and Nearest Neighbour	71
9.1.1	k Nearest Neighbours Query	71
9.1.2	Range Query	72
9.1.3	Metric Trees	72
9.1.4	Locality Sensitive Hashing	72
9.1.5	Distributed Similarity Search	73
9.2	Statistical Machine Learning	73
9.2.1	Vector Space Models	73
9.2.2	Kernel Methods	74
9.3	Research Opportunities	74
	References	74
10	Applications	77
10.1	Malware Classification	77
10.1.1	Raw Code	77
10.1.2	Instructions	78
10.1.3	Basic Blocks	78
10.1.4	API Calls	79
10.1.5	Control Flow and Data Flow	79
10.1.6	Data Flow	79
10.1.7	Call Graph	79
10.1.8	Control Flow Graphs	80

10.2	Software Theft Detection (Static Approaches)	80
10.2.1	Instructions	80
10.2.2	Control Flow	80
10.2.3	API Calls.	81
10.2.4	Object Dependencies.	81
10.3	Software Theft Detection (Dynamic Approaches).	81
10.3.1	Instructions	81
10.3.2	Control Flow	81
10.3.3	API Calls.	81
10.3.4	Dependence Graphs	82
10.4	Plagiarism Detection.	82
10.4.1	Raw Code and Tokens	82
10.4.2	Parse Trees	82
10.4.3	Program Dependency Graph	83
10.5	Code Clone Detection.	83
10.5.1	Raw Code and Tokens	83
10.5.2	Abstract Syntax Tree	83
10.5.3	Program Dependency Graph	83
10.6	Critical Analysis.	83
	References	84
11	Future Trends and Conclusion	87
11.1	Future Trends.	87
11.2	Conclusion.	88



<http://www.springer.com/978-1-4471-2908-0>

Software Similarity and Classification

Cesare, S.; Xiang, Y.

2012, XIV, 88 p. 26 illus., Softcover

ISBN: 978-1-4471-2908-0