# Contents

Road map ix
Acknowledgments x
1 Introduction 1

I Generic separation logic 9
2 Hoare logic 10
3 Separation logic 16
4 Soundness of Hoare logic 25
5 Mechanized Semantic Library 33
6 Separation algebras 35
7 Operators on separation algebras 44
8 First-order separation logic 49
9 A little case study 55
10 Covariant recursive predicates 63
11 Share accounting 69

II Higher order separation logic 75
12 Separation logic as a logic 76
13 From separation algebras to separation logic 84
14 Simplification by rewriting 89
15 Introduction to step-indexing 94
16 Predicate implication and subtyping 99
17 General recursive predicates 104
18 Case study: Separation logic with first-class functions 111
# Table of Contents

19 Data structures in indirection theory 123  
20 Applying higher-order separation logic 130  
21 Lifted separation logics 134  

III Separation logic for CompCert 141  
22 Verifiable C 142  
23 Expressions, values, and assertions 148  
24 The VST separation logic for C light 153  
25 Typechecking for Verifiable C 173  
26 Derived rules and proof automation for C light 184  
27 Proof of a program 195  
28 More C programs 208  
29 Dependently typed C programs 217  
30 Concurrent separation logic 222  

IV Operational semantics of CompCert 232  
31 CompCert 233  
32 The CompCert memory model 237  
33 How to specify a compiler 272  
34 C light operational semantics 288  

V Higher-order semantic models 294  
35 Indirection theory 295  
36 Case study: Lambda-calculus with references 316  
37 Higher-order Hoare logic 340  
38 Higher-order separation logic 347  
39 Semantic models of predicates-in-the-heap 351  

VI Semantic model and soundness of Verifiable C 362  
40 Separation algebra for CompCert 363  
41 Share models 374  
42 Juicy memories 385  
43 Modeling the Hoare judgment 392  
44 Semantic model of CSL 401
Road map

Readers interested in the theory of separation logic (with some example applications) should read Chapters 1–21. Readers interested in the use of separation logic to verify C programs should read Chapters 1–6 and 8–30. Those interested in the theory of step-indexing and indirection theory should read Chapters 35–39. Those interested in building models of program logics proved sound for certified compilers should read Chapters 40–47, though it would be helpful to read Chapters 1–39 as a warm-up.