Contents

Preface v

1 An Introduction to Data Flow Analysis 1
1.1 A Motivating Example 1
1.1.1 Optimizing for Heap Memory 1
1.1.2 Computing Liveness 4
1.1.3 Computing Aliases 9
1.1.4 Performing Optimization 10
1.1.5 General Observations 10
1.2 Program Analysis: The Larger Perspective 12
1.3 Characteristics of Data Flow Analysis 16
1.4 Summary and Concluding Remarks 18
1.5 Bibliographic Notes 19

I Intraprocedural Data Flow Analysis 21

2 Classical Bit Vector Data Flow Analysis 23
2.1 Basic Concepts and Notations 23
2.2 Discovering Local Data Flow Information 24
2.3 Discovering Global Properties of Variables 26
2.3.1 Live Variables Analysis 26
2.3.2 Dead Variables Analysis 29
2.3.3 Reaching Definitions Analysis 29
2.3.4 Reaching Definitions for Copy Propagation 32
2.4 Discovering Global Properties of Expressions 33
2.4.1 Available Expressions Analysis 33
2.4.2 Partially Available Expressions Analysis 36
2.4.3 Anticipable Expressions Analysis 37
2.4.4 Classical Partial Redundancy Elimination 39
2.4.5 Lazy Code Motion 49
2.5 Combined *May-Must* Analyses 53
2.6 Summary and Concluding Remarks 56
2.7 Bibliographic Notes 57

3 Theoretical Abstractions in Data Flow Analysis 59
3.1 Graph Properties Relevant to Data Flow Analysis 59
3.2 Data Flow Framework 63
3.2.1 Modeling Data Flow Values Using Lattices 64
3.2.2 Modeling Flow Functions 71
3.2.3 Data Flow Frameworks 72
3.3 Data Flow Assignments 74
3.3.1 Meet Over Paths Assignment 75
3.3.2 Fixed Point Assignment 76
3.3.3 Existence of Fixed Point Assignment 77
3.4 Computing Data Flow Assignments 79
3.4.1 Computing *MFP* Assignment 79
3.4.2 Comparing *MFP* and *MOP* Assignments 81
3.4.3 Undecidability of *MOP* Assignment Computation 83
3.5 Complexity of Data Flow Analysis for Rapid Frameworks 85
3.5.1 Properties of Data Flow Frameworks 86
3.5.2 Complexity for General CFGs 90
3.5.3 Complexity in Special Cases 97
3.6 Summary and Concluding Remarks 99
3.7 Bibliographic Notes 100

4 General Data Flow Frameworks 101
4.1 Non-Separable Flow Functions 101
4.2 Discovering Properties of Variables 103
4.2.1 Faint Variables Analysis 103
4.2.2 Possibly Uninitialized Variables Analysis 106
4.2.3 Constant Propagation 108
4.2.4 Variants of Constant Propagation 115
4.3 Discovering Properties of Pointers 119
4.3.1 Points-To Analysis of Stack and Static Data 119
4.3.2 Alias Analysis of Stack and Static Data 129
4.3.3 Formulating Data Flow Equations for Alias Analysis . . . 132
4.4 Liveness Analysis of Heap Data 135
4.4.1 Access Expressions and Access Paths 137
4.4.2 Liveness of Access Paths 138
4.4.3 Representing Sets of Access Paths by Access Graphs . . . 141
4.4.4 Data Flow Analysis for Explicit Liveness 146
4.4.5 The Motivating Example Revisited 151
4.5 Modeling Entity Dependence 152
4.5.1 Primitive Entity Functions 153
4.5.2 Composite Entity Functions 155
4.6 Summary and Concluding Remarks 156
4.7 Bibliographic Notes 156

5 Complexity of Iterative Data Flow Analysis 159
5.1 Generic Flow Functions and Data Flow Equations 159
5.2 Generic Round-Robin Iterative Algorithm 162
5.3 Complexity of Round-Robin Iterative Algorithm 164
5.3.1 Identifying the Core Work Using Work List 165
5.3.2 Information Flow Paths in Bit Vector Frameworks 171
5.3.3 Defining Complexity Using Information Flow Paths . . . 173
5.3.4 Information Flow Paths in Fast Frameworks 175
5.3.5 Information Flow Paths in Non-separable Frameworks . . 179
5.4 Summary and Concluding Remarks 184
5.5 Bibliographic Notes 184

6 Single Static Assignment Form as Intermediate Representation 185
6.1 Introduction 185
6.1.1 An Overview of SSA 186
6.1.2 Benefits of SSA Representation 188
6.2 Construction of SSA Form Programs 189
6.2.1 Dominance Frontier 191
6.2.2 Placement of ϕ-instructions 194
6.2.3 Renaming of Variables 196
6.2.4 Correctness of the Algorithm 198
6.3 Destruction of SSA 207
6.3.1 An Algorithm for SSA Destruction 209
6.3.2 SSA Destruction and Register Allocation 216
6.4 Summary and Concluding Remarks 227
6.5 Bibliographic Notes 228

II Interprocedural Data Flow Analysis 231

7 Introduction to Interprocedural Data Flow Analysis 233
7.1 A Motivating Example 233
7.2 Program Representations for Interprocedural Analysis 234
7.3 Modeling Interprocedural Data Flow Analysis 236
7.3.1 Summary Flow Functions 236
7.3.2 Inherited and Synthesized Data Flow Information 237
7.3.3 Approaches to Interprocedural Data Flow Analysis 238
7.4 Compromising Precision for Scalability 239
7.4.1 Flow and Context Insensitivity 240
7.4.2 Side Effects Analysis 244
7.5 Language Features Influencing Interprocedural Analysis 244
7.6 Common Variants of Interprocedural Data Flow Analysis 246
7.6.1 Intraprocedural Analysis with Conservative Interprocedural Approximation 246
7.6.2 Intraprocedural Analysis with Side Effects Computation . 248
7.6.3 Whole Program Analysis 253

7.7 An Aside on Interprocedural Optimizations 254
7.8 Summary and Concluding Remarks 256
7.9 Bibliographic Notes 256

8 Functional Approach to Interprocedural Data Flow Analysis 259
8.1 Side Effects Analysis of Procedure Calls 259
8.1.1 Computing Flow Sensitive Side Effects 261
8.1.2 Computing Flow Insensitive Side Effects 263
8.2 Handling the Effects of Parameters 266
8.2.1 Defining Aliasing of Parameters 267
8.2.2 Formulating Alias Analysis of Parameters 268
8.2.3 Augmenting Data Flow Analyses Using Parameter Aliases 271
8.2.4 Efficient Parameter Alias Analysis 273
8.3 Whole Program Analysis 274
8.3.1 Lattice of Flow Functions 274
8.3.2 Reducing Function Compositions and Confluences 275
8.3.3 Constructing Summary Flow Functions 278
8.3.4 Computing Data Flow Information 282
8.3.5 Enumerating Summary Flow Functions 285
8.4 Summary and Concluding Remarks 290
8.5 Bibliographic Notes 291

9 Value-Based Approach to Interprocedural Data Flow Analysis 293
9.1 Program Model for Value-Based Approaches to Interprocedural Data Flow Analysis 293
9.2 Interprocedural Analysis Using Restricted Contexts 296
9.3 Interprocedural Analysis Using Unrestricted Contexts 301
9.3.1 Using Call Strings to Represent Unrestricted Contexts . . 302
9.3.2 Issues in Termination of Call String Construction 305
9.4 Bounding Unrestricted Contexts Using Data Flow Values 311
9.4.1 Call String Invariants 311
9.4.2 Value-Based Termination of Call String Construction . . . 317
9.5 The Motivating Example Revisited 324
9.6 Summary and Concluding Remarks 326
9.7 Bibliographic Notes 328

III Implementing Data Flow Analysis 331

10 Implementing Data Flow Analysis in GCC 333
10.1 Specifying a Data Flow Analysis 333
10.1.1 Registering a Pass With the Pass Manager in GCC 334
10.1.2 Specifying Available Expressions Analysis 336
10.1.3 Specifying Other Bit Vector Data Flow Analyses 338
10.2 An Example of Data Flow Analysis 340
10.2.1 Executing the Data Flow Analyzer 341

10.2.2 Examining the Gimple Version of CFG 342
10.2.3 Examining the Result of Data Flow Analysis 346
10.3 Implementing the Generic Data Flow Analyzer *gdfa* 352
10.3.1 Specification Primitives 352
10.3.2 Interface with GCC 354
10.3.3 The Preparatory Pass 358
10.3.4 Local Data Flow Analysis 358
10.3.5 Global Data Flow Analysis 360
10.4 Extending the Generic Data Flow Analyzer *gdfa* 363

A An Introduction to GCC **365**
A.1 About GCC 365
A.2 Building GCC 366
A.3 Further Readings in GCC 368

References **371**

Index **378**