

# **Data Flow Analysis**

**Theory and Practice**

# Data Flow Analysis

## Theory and Practice

Uday P. Khedker  
Amitabha Sanyal  
Bageshri Karkare



CRC Press

Taylor & Francis Group

Boca Raton London New York

---

CRC Press is an imprint of the  
Taylor & Francis Group, an **informa** business

CRC Press  
Taylor & Francis Group  
6000 Broken Sound Parkway NW, Suite 300  
Boca Raton, FL 33487-2742

© 2009 by Taylor & Francis Group, LLC  
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works  
Printed in the United States of America on acid-free paper  
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-13: 978-0-8493-2880-0 (Hardcover)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access [www.copyright.com](http://www.copyright.com) (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

**Trademark Notice:** Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

---

**Library of Congress Cataloging-in-Publication Data**

---

Khedker, Uday.  
Data flow analysis : theory and practice / Uday Khedker, Amitabha Sanyal, Bageshri Karkare.  
p. cm.  
Includes bibliographical references and index.  
ISBN 978-0-8493-2880-0 (hardcover : alk. paper)  
1. Compilers (Computer programs) 2. Data flow computing. 3. Software engineering. 4. Computer software--Verification. I. Sanyal, Amitabha. II. Karkare, Bageshri. III. Title.

QA76.76.C65K54 2009  
004'.35--dc22

2009002056

---

Visit the Taylor & Francis Web site at  
<http://www.taylorandfrancis.com>  
and the CRC Press Web site at  
<http://www.crcpress.com>

---

# Preface

Data flow analysis is a classical static analysis technique that has been used to discover useful properties of programs being analyzed. It has found many useful applications ranging from compiler optimizations to software engineering to software verification. Modern compilers use this technique to produce code that maximize performance. In software engineering, it is used to re-engineer or reverse engineer programs. Finally, data flow analysis based techniques are used in software verification to prove the soundness of programs with respect to properties of interest.

This book provides a detailed treatment of data flow analysis. Although we explain it in the context of compiler optimizations, the concepts are general enough to be used for other applications. This is possible because we use a general model of data flow equations to represent the specification of data flow analysis. These data flow equations are defined in terms of constant and dependent *Gen* and *Kill* components. For classical bit vector frameworks, the constant *Gen* and *Kill* suffice; dependent parts are required for frameworks like constant propagation, points-to analysis etc. Such a modeling explicates the inter-dependence of data flow values and leads to an orthogonal generality that models flow functions in terms of a rather small set of constituent functions called *entity functions*. On the one hand, modeling flow functions in terms of entity functions allows us to define *information flow paths* that explain empirical observations for a large class of data flow frameworks and facilitate tight complexity bounds on solution procedures for data flow equations. On the other hand, this modeling also allows reasoning about the feasibility of constructing summary flow functions.

The book is organized in three parts: The first part deals with the specification of data flow frameworks and the solution process at the intraprocedural level. This part presents the lattice theoretic modeling of data flow frameworks apart from the generalizations of constant and dependent parts in flow functions and entity functions as constituents of flow functions. It shows how these generalizations lead to tight complexity bounds. This part also presents a large number of data flow frameworks. The diversity of these analyses is an evidence of the wide applicability of the generalizations presented. The final chapter of the first part presents SSA representation of programs. This is interesting because it builds an additional layer of abstraction over the control flow graph representation of programs and directly relates the definition points and the use points of data. This increases the efficiency with which a class of optimizations can be performed.

The second part of the book presents interprocedural data flow analysis. As a matter of choice, we avoid methods that are specific to a particular application or

a particular data flow framework and instead, focus on generic approaches. The first approach is a *functional* approach that constructs context independent summary flow functions of procedures. These flow functions are used at the call points to incorporate the effects of procedure calls. The second approach is a *value-based* approach that computes distinct values for distinct calling contexts; this is achieved by augmenting the data flow values with context information.

The third part of the book describes the implementation of a generic data flow analyzer for bit vector frameworks in GCC and shows how it can be instantiated to a given framework.

This book is an outcome of our notes for the course *CS618: Program Analysis* which is a graduate course at the Department of Computer Science and Engineering, IIT Bombay. The slides used in the course and the source of the generic data flow analyzer *gdfa* are available at the web page of the book:

<http://www.cse.iitb.ac.in/~uday/dfaBook-web>

As errors are discovered, we will upload an errata on the above web page. Any additional material that we find relevant to a course based on this book will also be made available on the same web page.

Many people have gone through the earlier versions of this manuscript. The registrants of CS618 were our captive audience for testing our examples—some examples tested their patience in the examinations of CS618. The following students of CS618 pointed out errors to us: Abhishek Shrivastav, Amitraj Singh Chouhan, Dhritiman Das, Harshada Gune, Md. Naseeruddin, Nilesh Padariya, Prashima Sharma, and Pushpraj Agrawal. Among others, Jaishri Waghmare, Prashant Singh Rawat, Sameera Deshpande, Santosh Sonawane, and Seema Ravandale read some chapters and gave valuable comments. Seema extended *gdfa* to include support for reaching definitions analysis. Sameera's help in preparing the first draft of the index is gratefully acknowledged.

Finally, this book would not have been possible without the patience and constant encouragement of our families. They have gracefully tolerated our mental, if not physical, absence, relieving us from a sense of guilt. We express a deep sense of gratitude for their support.

Uday P. Khedker, Amitabha Sanyal, and Bageshri Karkare

To my mother Rajani and the memory of my father Prabhakar Khedker

Uday Khedker

To my parents Arunojjwal and Prakriti Sanyal

Amitabha Sanyal

---

# Contents

<b>Preface</b>	<b>v</b>
<b>1 An Introduction to Data Flow Analysis</b>	<b>1</b>
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
<b>I Intraprocedural Data Flow Analysis</b>	<b>21</b>
<b>2 Classical Bit Vector Data Flow Analysis</b>	<b>23</b>
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 <i>May-Must</i> Analyses . . . . .	53
2.6 Summary and Concluding Remarks . . . . .	56
2.7 Bibliographic Notes . . . . .	57

<b>3</b>	<b>Theoretical Abstractions in Data Flow Analysis</b>	<b>59</b>
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 <i>MFP</i> Assignment . . . . .	79
3.4.2	Comparing <i>MFP</i> and <i>MOP</i> Assignments . . . . .	81
3.4.3	Undecidability of <i>MOP</i> 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
<b>4</b>	<b>General Data Flow Frameworks</b>	<b>101</b>
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



<b>5</b>	<b>Complexity of Iterative Data Flow Analysis</b>	<b>159</b>
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
<b>6</b>	<b>Single Static Assignment Form as Intermediate Representation</b>	<b>185</b>
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 $\phi$ -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
<b>II</b>	<b>Interprocedural Data Flow Analysis</b>	<b>231</b>
<b>7</b>	<b>Introduction to Interprocedural Data Flow Analysis</b>	<b>233</b>
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
<b>8</b>	<b>Functional Approach to Interprocedural Data Flow Analysis</b>	<b>259</b>
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
<b>9</b>	<b>Value-Based Approach to Interprocedural Data Flow Analysis</b>	<b>293</b>
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
<b>III</b>	<b>Implementing Data Flow Analysis</b>	<b>331</b>
<b>10</b>	<b>Implementing Data Flow Analysis in GCC</b>	<b>333</b>
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 <i>gdfa</i> . . . . .	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 <i>gdfa</i> . . . . .	363
<b>A</b>	<b>An Introduction to GCC</b> . . . . .	<b>365</b>
A.1	About GCC . . . . .	365
A.2	Building GCC . . . . .	366
A.3	Further Readings in GCC . . . . .	368
	<b>References</b> . . . . .	<b>371</b>