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

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)

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. Naseerunddin, 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