Principles of Program Analysis
Flemming Nielson  Hanne Riis Nielson
Chris Hankin

Principles of Program Analysis

With 56 Figures and 51 Tables

Springer
Flemming Nielson
Hanne Riis Nielson
The Technical University of Denmark
Informatics and Mathematical Modelling
Richard Petersens Plads, Bldg. 321
2800 Kongens Lyngby, Denmark
E-mail: {nielson, riis}@imm.dtu.dk
WWW: http://www.imm.dtu.dk/~riis/PPA/ppa.html

Chris Hankin
Department of Computing
The Imperial College of Science, Technology, and Medicine
180 Queen’s Gate, London SW7 2BZ, UK
E-mail: clh@doc.ic.ac.uk

Cataloging-in-Publication data applied for
Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Nielson, Flemming:
Principles of program analysis: with 51 tables/F. Nielson; H.R.
Nielson; C. Hankin. – Berlin; Heidelberg; New York; Barcelona;
Hong Kong; London; Milan; Paris; Singapore; Tokyo: Springer,
1999
ISBN 3-540-65410-0

ISBN 3-540-65410-0 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned,
specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on
microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is
permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version,
and permission for use must always be obtained from Springer. Violations are liable for prosecution under the
German Copyright Law.

Springer is a part of Springer Science+Business Media
sprigeronline.com
© Springer-Verlag Berlin Heidelberg 1999, 2005
Printed in Germany

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in
the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and
therefore free for general use.

Typesetting: Camera-ready by the authors
Cover design: Erich Kirchner, Heidelberg
Production: LE-TeX Jelonek, Schmidt & Vöckler GbR, Leipzig
Printed on acid-free paper 45/3142/YL - 5 4 3 2 1 0
Preface

Aims of the book. Our motivation for writing this book is twofold. One is that for quite some time we have lacked a textbook for our own courses on program analysis; instead we have been forced to base the courses on conference papers supplemented with the occasional journal paper or chapter from a text book out of print. The other is the growing feeling that the various subcommunities in the field often study similar problems without being sufficiently aware of insights or developments generally known in other subcommunities. The idea then emerged that perhaps a text book could be written that both would be useful for advanced courses on program analysis and that would help increase the awareness in the field about the many similarities between the various approaches.

There is an important analogy to complexity theory here. Consider researchers or students looking through the literature for a clever algorithm or heuristics to solve a problem on graphs. They might come across papers on clever algorithms or heuristics for solving problems on boolean formulae and might dismiss them on the grounds that graphs and boolean formulae surely are quite different things. Yet complexity theory tells us that this is quite a mistaken point of view. The notions of log-space reduction between problems and of NP-complete problems lead to realising that problems may appear unrelated at first sight and nonetheless be so closely related that a good algorithm or heuristics for one will give rise to a good algorithm or heuristics for the other. We believe it to be a sad fact that students of programming languages in general, and program analysis in particular, are much too often allowed to get away with making similarly naive decisions and arguments. Program analysis is still far from being able to precisely relate ingredients of different approaches to one another but we hope that this book will help to bring this about. In fact, we hope to shock quite a number of readers by convincing them that there are important similarities between the approaches of their own work and other approaches deemed to be of no relevance.

This book concentrates on what we believe to be the four main approaches to program analysis: Data Flow Analysis; Constraint Based Analysis; Abstract
Interpretation; and Type and Effect Systems. For each approach we aim at identifying and describing the important general principles, rather than presenting a cook-book of techniques and language constructs, with a view to how the principles scale up for more complex programming languages and analyses.

As a consequence we have deliberately decided that this book should not treat a number of interesting approaches, some of which are dear to the heart of the authors, in order to be able to cover the four main approaches to some depth; the approaches not covered include denotationally based program analysis, projection analysis, and logical formulations based on Stone dualities. For reasons of space, we have also had to omit material that would have had a natural place in the book; this includes a deeper treatment of set constraints, fast techniques for implementing type based analyses, single static assignment form, a broader treatment of pointer analyses, and the interplay between analysis and transformation and how to efficiently recompute analysis information invalidated by the transformation.

How to read the book. The book is relatively self-contained although the reader will benefit from previous exposure to discrete mathematics and compilers. The main chapters are generally rigorous in the early parts where the basic techniques are covered, but less so in the later parts where more advanced techniques are covered.

Chapter 1 is intended to be read quickly. The purpose of the chapter is to give an overview of the main approaches to program analysis, to stress the approximate nature of program analysis, and to make it clear that seemingly different approaches may yet have profound similarities. We recommend reading through all of Chapter 1 even though the reader might want to specialise in only parts of the book.

Chapter 2 introduces Data Flow Analysis. Sections 2.1 to 2.4 cover the basic techniques for intraprocedural analysis, including the Monotone Frameworks as generalisations of the Bit Vector Frameworks, and a worklist algorithm for computing the information efficiently. Section 2.2 covers more theoretical properties (semantic correctness) and can be omitted on a first reading. The presentation makes use of notions from lattice theory and for this we refer to Appendix A.1, A.2 and parts of A.3. — Section 2.5 is a more advanced section that gives an overview of interprocedural analysis including a treatment of call string based methods and methods based on assumption sets; since Section 2.5 is used as a stepping stone to Chapter 3 we recommend to read at least up to Subsection 2.5.2. Section 2.6 is an advanced section that illustrates how the relatively simple techniques introduced so far can be combined to develop a very complex shape analysis, but the material is not essential for the remainder of the book.
Chapter 3 covers Constraint Based Analysis. The treatment makes a clear distinction between determining the safety of an analysis result and how to compute the best safe result; it also stresses the need to analyse open systems. Sections 3.1, 3.3 and 3.4 cover the basic techniques; these include coinduction which is likely to be new to most readers and we refer to the treatment in Appendix B (building upon Tarski’s theorem as covered in Appendix A.4). Section 3.2 covers more theoretical properties (semantic correctness and the existence of best solutions) and can be omitted on a first reading. — Sections 3.5 and 3.6 extend the development so as to link up with the treatment of Data Flow Analysis. Section 3.5 shows how to incorporate Monotone Frameworks (Section 2.3) and Section 3.6 shows how to add context in the manner of call strings and assumption sets (Section 2.5).

Chapter 4 covers Abstract Interpretation in a programming language independent fashion in order to stress that it can be integrated both with Data Flow Analysis and Constraint Based Analysis. Section 4.1 introduces some of the key considerations and is used to motivate some of the technical definitions in later sections. Section 4.2 deals with the use of widening and narrowing for approximating fixed points and Sections 4.3 deals with Galois connections; this order of presentation has been chosen to stress the fundamental nature played by widening but the sections are largely independent of one another. — Sections 4.4 and 4.5 study how to build Galois connections in a systematic manner and how to use them for inducing approximate analyses; the material is not essential for the remainder of the book.

Chapter 5 covers Type and Effect Systems which is an approach to program analysis that is often viewed as having a quite different flavour from the approaches covered so far. Section 5.1 presents the basic approach (by linking back to the Constraint Based Analyses studied in Chapter 3) and suffices for getting an impression of the approach. Section 5.2 studies more theoretical properties (semantic correctness) and Section 5.3 studies algorithmic issues (soundness and completeness of a variation of algorithm $\mathcal{W}$) and these sections can be omitted on a first reading. — Sections 5.4 and 5.5 gradually introduce more and more advanced Type and Effect Systems.

Chapter 6 presents algorithms for Data Flow Analysis and Constraint Based Analysis. The treatment concentrates on general techniques for solving systems of inequations. We emphasise the fact that, to a large extent, the same set of techniques can be used for a number of different approaches to program analysis. Section 6.1 presents a general worklist algorithm, where the operations on the worklist constitute an abstract data type, and its correctness and complexity is established. Section 6.2 organises the worklist so that iteration takes place in reverse postorder and the Round Robin Algorithm is obtained as a special case. Section 6.3 then further identifies strong components and iterates through each strong component in reverse postorder before considering the next.
Appendices A and C review the concepts from partially ordered sets, graphs and regular expressions that are used throughout the book. Appendix B is more tutorial in nature since coinduction is likely to be a new concept for most readers.

To help the reader when navigating through the book we provide a table of contents, a list of tables, and a list of figures at the front of the book and an index of the main concepts and an index of notation at the end of the book. The index of notation is divided into three parts: first the mathematical symbols (with an indication of where they take their parameters), then the notation beginning with a Greek letter, and finally the notation beginning with a letter in the Latin alphabet (regardless of the font used). The book concludes with a bibliography.

Our notation is mostly standard and is explained when introduced. However, it may be helpful to point out that we will be using "iff" as an abbreviation for "if and only if", that we will be using \cdots [\cdots \mapsto \cdots] to mean both syntactic substitution as well as update of an environment or store, and that we will be writing \cdots \mapsto_{\text{fin}} \cdots for the set of finitary functions: these are the partial functions with a finite domain. We also use \lambda-notation for functions when it improves the clarity of presentation: \lambda x. \cdots x \cdots stands for the unary function \( f \) defined by \( f(x) = \cdots x \cdots. \)

Most proofs and some technical lemmas and facts are in small print in order to aid the reader in navigating through the book.

**How to teach from the book.** The book contains more material than can be covered in a one semester course. The pace naturally depends on the background of the students; we have taught the course at different paces to students in their fourth year as well as to Ph.D.-students with a variety of backgrounds. Below we summarise our experiences on how many lectures are needed for covering the various parts of the book; it supplements the guidelines presented above for how to read the book.

Two or three lectures should suffice for covering all of Chapter 1 and possibly some of the simpler concepts from Appendix A.1 and A.2.

Sections 2.1, 2.3 and 2.4 are likely to be central to any course dealing with data flow analysis. Three to four lectures should suffice for covering Sections 2.1 to 2.4 but five lectures may be needed if the students lack the necessary background in operational semantics or lattice theory (Appendix A.1, A.2 and parts of A.3). — Sections 2.5 and 2.6 are more advanced. One or two lectures suffice for covering Section 2.5 but it is hard to pay justice to Section 2.6 in less than two lectures.

Four or five lectures should suffice for a complete treatment of both Chapter 3 and Appendix B; however, the concept of coinduction takes some time to get used to and should be explained more than once.
Four or five lectures should suffice for a complete treatment of Chapter 4 as well as Appendix A.4.

About four lectures should suffice for a complete treatment of Chapter 5.

Two lectures should suffice for Chapter 6 but three lectures may be needed if large parts of Appendix C need to be reviewed.

We have always covered the appendices as an integrated part of the other chapters; indeed, some of the foundations for partial orders are introduced gently in Chapter 1 and most of our students have had some prior exposure to partial orders, graphs and regular expressions.

The book contains numerous exercises and several mini projects which are small projects dealing with practical or theoretical aspects of the development. Many of the important links and analogies between the various chapters are studied in the exercises and mini projects. Some of the harder exercises are starred.

Acknowledgements. We should like to thank Reinhard Wilhelm for his long and lasting interest in this project and for his many encouraging and constructive remarks. We have also profited greatly from discussions with Alan Mycroft, Mooly Sagiv and Helmut Seidl about their perspective on selected parts of the manuscript. Many other colleagues have influenced the writing of the book and we should like to thank them all; in particular Alex Aiken, Torben Amtoft, Patrick Cousot, Laurie Hendren, Suresh Jagannathan, Florian Martin, Barbara Ryder, Bernhard Steffen. We are grateful to Schloss Dagstuhl for having hosted two key meetings: a one-week meeting among the authors in March of 1997 (during which we discovered the soothing nature of Vangelis’ 1492) and our advanced course in November of 1998. Warm thanks go to the many students attending the advanced course in Dagstuhl, and to the many students in Aarhus, London, Saarbrücken and Tel Aviv who have tested the book as it evolved ever so slowly. Finally, we should like to thank Alfred Hofmann at Springer for a very satisfactory contract and René Rydhof Hansen for his help in tuning the \LaTeX\ commands.

Aarhus and London
August, 1999

Flemming Nielson
Hanne Riis Nielson
Chris Hankin

Preface to the second printing. In this second printing we have corrected all errors and shortcomings pointed out to us. We should like to thank Torben Amtoft, John Tang Boyland, Jurriaan Hage and Mirko Luedde as well as our students for their observations.

Official web page. Further information about the book is available at the web page http://www.imm.dtu.dk/~riis/PPA/ppa.html. Here we will
provide information about availability of the book, a list of misprints (initially empty), pointers to web based tools that can be used in conjunction with the book, and transparencies and other supplementary material. Instructors are encouraged to send us teaching material for inclusion on the web page.

Lyngby and London
October, 2004

Flemming Nielson
Hanne Riis Nielson
Chris Hankin
Contents

1 Introduction  .................................................. 1
  1.1 The Nature of Program Analysis  ..................... 1
  1.2 Setting the Scene  .................................. 3
  1.3 Data Flow Analysis  ................................ 5
    1.3.1 The Equational Approach  ..................... 5
    1.3.2 The Constraint Based Approach  .............. 8
  1.4 Constraint Based Analysis  .......................... 10
  1.5 Abstract Interpretation  .............................. 13
  1.6 Type and Effect Systems  ............................ 17
    1.6.1 Annotated Type Systems  ....................... 18
    1.6.2 Effect Systems  ................................ 22
  1.7 Algorithms  ........................................... 25
  1.8 Transformations  ...................................... 27
  Concluding Remarks  ..................................... 29
  Mini Projects  ........................................... 29
  Exercises  ................................................. 31

2 Data Flow Analysis  ........................................ 35
  2.1 Intraprocedural Analysis  ............................ 35
    2.1.1 Available Expressions Analysis ................. 39
    2.1.2 Reaching Definitions Analysis .................. 43
    2.1.3 Very Busy Expressions Analysis ................. 46
    2.1.4 Live Variables Analysis  ......................... 49
    2.1.5 Derived Data Flow Information ................. 52
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>Theoretical Properties</td>
<td>54</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Structural Operational Semantics</td>
<td>54</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Correctness of Live Variables Analysis</td>
<td>60</td>
</tr>
<tr>
<td>2.3</td>
<td>Monotone Frameworks</td>
<td>65</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Basic Definitions</td>
<td>67</td>
</tr>
<tr>
<td>2.3.2</td>
<td>The Examples Revisited</td>
<td>70</td>
</tr>
<tr>
<td>2.3.3</td>
<td>A Non-distributive Example</td>
<td>72</td>
</tr>
<tr>
<td>2.4</td>
<td>Equation Solving</td>
<td>74</td>
</tr>
<tr>
<td>2.4.1</td>
<td>The MFP Solution</td>
<td>74</td>
</tr>
<tr>
<td>2.4.2</td>
<td>The MOP Solution</td>
<td>78</td>
</tr>
<tr>
<td>2.5</td>
<td>Interprocedural Analysis</td>
<td>82</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Structural Operational Semantics</td>
<td>85</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Intraprocedural versus Interprocedural Analysis</td>
<td>88</td>
</tr>
<tr>
<td>2.5.3</td>
<td>Making Context Explicit</td>
<td>90</td>
</tr>
<tr>
<td>2.5.4</td>
<td>Call Strings as Context</td>
<td>95</td>
</tr>
<tr>
<td>2.5.5</td>
<td>Assumption Sets as Context</td>
<td>99</td>
</tr>
<tr>
<td>2.5.6</td>
<td>Flow-Sensitivity versus Flow-Insensitivity</td>
<td>101</td>
</tr>
<tr>
<td>2.6</td>
<td>Shape Analysis</td>
<td>104</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Structural Operational Semantics</td>
<td>105</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Shape Graphs</td>
<td>109</td>
</tr>
<tr>
<td>2.6.3</td>
<td>The Analysis</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>Concluding Remarks</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>Mini Projects</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>Exercises</td>
<td>135</td>
</tr>
</tbody>
</table>

### 3 Constraint Based Analysis

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Abstract 0-CFA Analysis</td>
<td>141</td>
</tr>
<tr>
<td>3.1.1</td>
<td>The Analysis</td>
<td>143</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Well-definedness of the Analysis</td>
<td>150</td>
</tr>
<tr>
<td>3.2</td>
<td>Theoretical Properties</td>
<td>153</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Structural Operational Semantics</td>
<td>153</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Semantic Correctness</td>
<td>158</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Existence of Solutions</td>
<td>162</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Coinduction versus Induction</td>
<td>165</td>
</tr>
<tr>
<td>3.3</td>
<td>Syntax Directed 0-CFA Analysis</td>
<td>168</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Syntax Directed Specification</td>
<td>169</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Preservation of Solutions</td>
<td>171</td>
</tr>
<tr>
<td>3.4</td>
<td>Constraint Based 0-CFA Analysis</td>
<td>173</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Preservation of Solutions</td>
<td>175</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Solving the Constraints</td>
<td>176</td>
</tr>
<tr>
<td>3.5</td>
<td>Adding Data Flow Analysis</td>
<td>182</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Abstract Values as Powersets</td>
<td>182</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Abstract Values as Complete Lattices</td>
<td>185</td>
</tr>
<tr>
<td>3.6</td>
<td>Adding Context Information</td>
<td>189</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Uniform $k$-CFA Analysis</td>
<td>191</td>
</tr>
<tr>
<td>3.6.2</td>
<td>The Cartesian Product Algorithm</td>
<td>196</td>
</tr>
<tr>
<td></td>
<td>Concluding Remarks</td>
<td>198</td>
</tr>
<tr>
<td></td>
<td>Mini Projects</td>
<td>202</td>
</tr>
<tr>
<td></td>
<td>Exercises</td>
<td>205</td>
</tr>
<tr>
<td>4</td>
<td>Abstract Interpretation</td>
<td>211</td>
</tr>
<tr>
<td>4.1</td>
<td>A Mundane Approach to Correctness</td>
<td>211</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Correctness Relations</td>
<td>214</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Representation Functions</td>
<td>216</td>
</tr>
<tr>
<td>4.1.3</td>
<td>A Modest Generalisation</td>
<td>219</td>
</tr>
<tr>
<td>4.2</td>
<td>Approximation of Fixed Points</td>
<td>221</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Widening Operators</td>
<td>224</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Narrowing Operators</td>
<td>230</td>
</tr>
<tr>
<td>4.3</td>
<td>Galois Connections</td>
<td>233</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Properties of Galois Connections</td>
<td>239</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Galois Insertions</td>
<td>242</td>
</tr>
<tr>
<td>4.4</td>
<td>Systematic Design of Galois Connections</td>
<td>246</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Component-wise Combinations</td>
<td>249</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Other Combinations</td>
<td>253</td>
</tr>
<tr>
<td>4.5</td>
<td>Induced Operations</td>
<td>258</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Inducing along the Abstraction Function</td>
<td>258</td>
</tr>
</tbody>
</table>
4.5.2 Application to Data Flow Analysis .................. 262
4.5.3 Inducing along the Concretisation Function ........... 267

Concluding Remarks ........................................ 270
Mini Projects .................................................. 274
Exercises ....................................................... 276

5 Type and Effect Systems .................................. 283
  5.1 Control Flow Analysis .................................. 283
  5.1.1 The Underlying Type System ....................... 284
  5.1.2 The Analysis ........................................ 287
  5.2 Theoretical Properties ................................ 291
  5.2.1 Natural Semantics ................................ 292
  5.2.2 Semantic Correctness ................................ 294
  5.2.3 Existence of Solutions ............................. 297
  5.3 Inference Algorithms .................................. 300
  5.3.1 An Algorithm for the Underlying Type System .... 300
  5.3.2 An Algorithm for Control Flow Analysis ........... 306
  5.3.3 Syntactic Soundness and Completeness ............. 312
  5.3.4 Existence of Solutions ............................. 317
  5.4 Effects ................................................ 319
  5.4.1 Side Effect Analysis .............................. 319
  5.4.2 Exception Analysis ................................ 325
  5.4.3 Region Inference .................................. 330
  5.5 Behaviours ............................................. 339
  5.5.1 Communication Analysis ........................... 339
Concluding Remarks ........................................ 349
Mini Projects .................................................. 353
Exercises ....................................................... 359

6 Algorithms .................................................. 365
  6.1 Worklist Algorithms .................................. 365
  6.1.1 The Structure of Worklist Algorithms .............. 368
  6.1.2 Iterating in LIFO and FIFO ....................... 372
  6.2 Iterating in Reverse Postorder ...................... 374
6.2.1 The Round Robin Algorithm .................................. 378
6.3 Iterating Through Strong Components ...................... 381
Concluding Remarks .................................................. 384
Mini Projects ......................................................... 387
Exercises ............................................................... 389

A Partially Ordered Sets .......................................... 393
A.1 Basic Definitions ............................................... 393
A.2 Construction of Complete Lattices ......................... 397
A.3 Chains .................................................................. 398
A.4 Fixed Points ...................................................... 402
Concluding Remarks ................................................ 404

B Induction and Coinduction ........................................ 405
B.1 Proof by Induction ............................................... 405
B.2 Introducing Coinduction ....................................... 407
B.3 Proof by Coinduction ........................................... 411
Concluding Remarks ................................................ 415

C Graphs and Regular Expressions ............................... 417
C.1 Graphs and Forests ............................................ 417
C.2 Reverse Postorder ............................................. 421
C.3 Regular Expressions ............................................ 426
Concluding Remarks ................................................ 427

Index of Notation .................................................... 429

Index ................................................................. 433

Bibliography .......................................................... 439