

1. Record Nr.	UNINA9910143887003321
Autore	Fahringer Thomas
Titolo	Advanced Symbolic Analysis for Compilers : New Techniques and Algorithms for Symbolic Program Analysis and Optimization // by Thomas Fahringer, Bernhard Scholz
Pubbl/distr/stampa	Berlin, Heidelberg : , : Springer Berlin Heidelberg : , : Imprint : Springer, , 2003
ISBN	3-540-36614-8
Edizione	[1st ed. 2003.]
Descrizione fisica	1 online resource (XII, 136 p.)
Collana	Lecture Notes in Computer Science, , 0302-9743 ; ; 2628
Disciplina	005.4/53
Soggetti	Software engineering Programming languages (Electronic computers) Operating systems (Computers) Computer logic Software Engineering/Programming and Operating Systems Programming Languages, Compilers, Interpreters Software Engineering Operating Systems Logics and Meanings of Programs
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Bibliographic Level Mode of Issuance: Monograph
Nota di bibliografia	Includes bibliographical references and index.
Nota di contenuto	Symbolic Analysis of Programs -- Generating Program Contexts -- Symbolic Analysis Algorithms and Transformations -- Symbolic Analysis for Parallelizing Compilers -- Related Work -- Conclusion.
Sommario/riassunto	The objective of program analysis is to automatically determine the properties of a program. Tools of software development, such as compilers, performance estimators, debuggers, reverse-engineering tools, program verification/testing/proving systems, program comprehension systems, and program specialization tools are largely dependent on program analysis. Advanced program analysis can: help to find program errors; detect and tune performance-critical code regions; ensure assumed constraints on data are not violated; tailor a generic program to suit a specific application; reverse-engineer software modules, etc. A prominent program analysis

technique is symbolic analysis, which has attracted substantial attention for many years as it is not dependent on executing a program to examine the semantics of a program, and it can yield very elegant formulations of many analyses. Moreover, the complexity of symbolic analysis can be largely independent of the input data size of a program and of the size of the machine on which the program is being executed. In this book we present novel symbolic control and data flow representation techniques as well as symbolic techniques and algorithms to analyze and optimize programs. Program contexts which define a new symbolic description of program semantics for control and data flow analysis are at the center of our approach. We have solved a number of problems encountered in program analysis by using program contexts. Our solution methods are efficient, versatile, unified, and more general (they cope with regular and irregular codes) than most existing methods.
