# Preface

Many complex systems, ranging from social, industrial, economics, financial, educational, to military, require that we obtain high-quality solutions to combinatorial problems. Linear programming and its extensions developed in operations research once formed the primary paradigm for solving combinatorial problems. During the last three decades, a plethora of paradigms have been developed for combinatorial problems, including constraint programming (CP), propositional satisfiability testing (SAT), satisfiability modulo theories (SMT), answer set programming (ASP), tabled logic programming, and heuristic search planners.

Picat is a new logic-based multi-paradigm programming language that integrates logic programming, functional programming, dynamic programming with tabling, and scripting. Picat provides facilities for solving combinatorial search problems, including solver modules that are based on CP, SAT, and MIP (mixed integer programming), and a module for planning that is implemented by the use of tabling. Chapter 1 gives an overview of the Picat language and system.

This book presents Picat as a modeling and solving language for two important classes of combinatorial problems: constraint satisfaction and planning. The constraint satisfaction problem (CSP) is a basic class of combinatorial problems. A CSP consists of a set of variables, each of which is defined over a domain, a set of constraints among the variables, and, optionally, an objective function. A solution to a CSP is a valuation of the variables that satisfies all the constraints and optimizes the objective function, if it exists. Chapters 2 and 3 are devoted to constraint modeling. Chapter 2 introduces the basic built-in constraints in the common API of the three solver modules (cp, sat, and mip), and gives several simple example models that use these constraints. Chapter 3 describes the more sophisticated constraints in the API and gives "real-world" example models for scheduling, resource allocation, and design.

Planning is an important task for building many systems, such as autonomous robots, industrial design software, system security, and military operational systems. Planning is also closely related to model checking. Given an initial state, a set of goal states, and a set of possible actions, the classic planning problem is to find a

plan that transforms the initial state to the goal state. Chapters 5 and 6 are devoted to planning. Chapter 5 introduces depth-unbounded search predicates in the planner module, and gives models for several planning puzzles. Chapter 6 introduces depth-bounded search predicates in the planner module, and shows how domain knowledge and heuristics can be incorporated into planning models to improve the performance. Because planning is solved as a dynamic programming problem with tabling in Picat, a separate chapter (Chap. 4) is included to introduce the tabling feature of Picat.

A combinatorial problem can normally be modeled in different ways and solved by different solvers. The book is wrapped up with a chapter (Chap. 7) that gives several models for solving the Traveling Salesman Problem with different solvers, including CP, SAT, MIP, and tabled planning.

This book is useful for students, including undergraduate-level students, researchers, and practitioners, to learn the modeling techniques for Picat. No prerequisite knowledge about Picat is required, although familiarity with logic or functional programming is a plus. Chapter 1 gives an overview of the data types, built-ins, and language constructs that are needed for the later chapters. For readers who are familiar with Prolog, Haskell, or Python, this chapter also compares Picat with these three languages.

This book does not cover the implementation of Picat. The bibliographical note at the end of each chapter aims to meet the reader's curiosity about how the presented tools are built. Each chapter ends with a set of exercises, which is intended for the reader to practice the presented modeling techniques. The code examples used in the book are available at: http://picat-lang.org/picatbook2015.html.

Neng-Fa Zhou would like to thank his other co-authors for their collaboration on some of the important ideas that underpin the Picat implementation: Roman Barták, Agostino Dovier, Christian Theil Have, Taisuke Sato, and Yi-Dong Shen; his recent students at CUNY who worked on application projects using Picat: Mike Bionchik and Lei Chen; his colleagues at CUNY who have been involved in some way in the Picat project: David Arnow, James Cox, Danny Kopec, and Subash Shankar. The authors would like to thank Ronan Nugent, the Springer editor, for his helpful comments and advice, and the following people who gave us permission to use their examples: Brian Dean, Christopher Jefferson, Tony Hürlimann, and Peter James Stuckey.

Brooklyn, NY, USA — Neng-Fa Zhou
Malmö, Sweden — Håkan Kjellerstrand
New York, NY, USA — Jonathan Fruhman
August 2015

http://www.springer.com/978-3-319-25881-2

Constraint Solving and Planning with Picat

Zhou, N.-F.; Kjellerstrand, H.; Fruhman, J.

2015, XI, 148 p. 40 illus., 31 illus. in color., Softcover
ISBN: 978-3-319-25881-2