

Emergence, Complexity and Computation ECC

Georgios Ch. Sirakoulis  
Andrew Adamatzky *Editors*

# Robots and Lattice Automata

Georgios Ch. Sirakoulis · Andrew Adamatzky  
Editors

# Robots and Lattice Automata

*Editors*

Georgios Ch. Sirakoulis  
Department of Electrical and Computer  
Engineering  
Democritus University of Thrace  
Xanthi  
Greece

Andrew Adamatzky  
Unconventional Computing Centre  
University of the West of England  
Bristol  
UK

ISSN 2194-7287

ISBN 978-3-319-10923-7

DOI 10.1007/978-3-319-10924-4

ISSN 2194-7295 (electronic)

ISBN 978-3-319-10924-4 (eBook)

Library of Congress Control Number: 2014951740

Springer Cham Heidelberg New York Dordrecht London

© Springer International Publishing Switzerland 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, 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.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

# Preface

Robots and Automata are notionally related. In this context, Automata (originated from the latinization of the Greek word “αυτόματον”) as self-operating autonomous machines, invented from ancient years can be easily considered as the first steps of these robotic-like efforts. On other words, an Automaton is a self-operating machine, while a robot is a hardware agent with role(s) to operate usually without an immediate human operator. Automata are useful tools for formal descriptions of robots. Automata themselves are formally represented by final state machines: the abstract machines which take finite number of states and change their state while triggered by certain conditions. Authors of the book bring together concepts, architectures and implementations of Lattice Automata and Robots. Lattice Automata are minimal universal instantiation of space and time. A Lattice Automaton is either a regular array of finite state machines or collectives of mobile finite state machines inhabiting a discrete space. In both cases the finite states machines, or Automata, update their states by the same rules depending on states of their immediate neighbours. Automata and Robots often share the same notional meaning: Automata are mathematical models of robots and also they are integral parts of robotic control systems.

The book opens with inspiring text by Rosenberg—Chap. 1—on computational potential of groups of identical finite-state machines. The chapter lays somewhat foundational theoretical background for the rest of the book.

Modular robots are kinematic machines of many units capable for changing its topology by dynamically updating connections between the units. To develop efficient algorithms of reconfiguration, we represent the robotic units by configurations of Lattice Automata and study Automaton transition rules corresponding to reconfiguration. The topic is studied in full details in three chapters: Chap. 2 by Stoy introduces the reader to the theoretical and general aspects of modular reconfigurable robots in Lattice Automata; Chap. 3 by Eckenstein and Yim reproduces all the up-to-date related works and corresponding modular reconfigurable robotic systems; while in Chap. 4, Tomita and co-authors provide full details for some of these modular systems, namely Fractum and M-Tran in every possible aspect and discuss the general problems of Lattice-based robotic systems.

Motion control and path planning are amongst key problems of robotics, they put high demands on detailed knowledge of environment and consume substantial computational resources. Five chapters explicitly deal with these problems. Thus, Arena and co-authors, in Chap. 5, use Automaton networks to control locomotion of the fly-inspired robot. Efficient ways of routing, an abstract version of path planning, are designed and analysed by Hoffman and Désérable in Chap. 6. Marchese proposes to use particular families of Cellular Automata to provide an optimal representation of space and maps in precise parallel motion planning, in Chap. 7. Charalampous and co-authors in Chap. 8 adapt classical designs of Cellular Automaton based shortest path finders to undertake autonomous collision-free navigation. Moreover, Ioannidis and co-authors proposed the employment of Cellular Automata advanced with Ant Colony Optimization techniques resulting to Cellular Robotic Ants synergy coordination for tackling the path planning problem for robotic teams in Chap. 9.

Further applications of Lattice Automata in Robotics are presented in the following chapters. A novel method of map representation is proposed in Chap. 10 by Kapoutsis and his co-authors. There, a configuration of elevation heights is converted to cells' states; thus, an entire map is represented by a Cellular Automaton configuration. Cellular Automata have been a classical tool in image processing community since mid-1970s, yet, there is still vast lands of unexplored features and algorithms. In his Chap. 11, Nalpantidis demonstrates practical, real-life implementation of Cellular Automaton algorithms onboard of a mobile robot.

The last two chapters deal with cooperative actions in large-scale robotic collectives. In both chapters, robots are oscillating mechanisms arranged on a two-dimensional array: their aim is to adjust their oscillations or states to produce a specified vibration pattern. Silva and co-authors, in Chap. 12 provide modelling and analysis of the space-time behaviour of such collectives and transitions between different modes of behaviour. Application of the vibrating automaton array to physical manipulator of objects in real life is studied by Georgilas and co-authors in Chap. 13. They show how Automaton model of an sub-excitable medium can be used to purposefully transport objects.

All chapters are written in an accessible manner and lavishly illustrated. The book will help computer and robotic scientists and engineers to understand mechanisms of decentralised functioning of robotic collectives and to design future and emergent reconfigurable, parallel and distributed robotic systems.

Georgios Ch. Sirakoulis  
Andrew Adamatzky

# Contents

<b>1</b>	<b>Algorithmic Insights into Finite-State Robots</b>	<b>1</b>
	Arnold L. Rosenberg	
1.1	Introduction	1
1.2	Technical Background	4
1.2.1	FSM “Robots” and Their Domains	4
1.2.2	Algorithmic Standards and Simplifications	8
1.3	$\mathcal{M}_n$ ’s “Walls” as an Algorithmic Tool	9
1.3.1	Algorithms Based on “Bouncing Off” $\mathcal{M}_n$ ’s “Walls”	9
1.3.2	Algorithms Based on “Hugging” $\mathcal{M}_n$ ’s “Walls”	15
1.4	The Power of Cooperation	21
1.4.1	The Need for Cooperation	21
1.4.2	Tasks Enabled by Cooperative Behavior	23
1.5	Conclusion	28
1.5.1	Retrospective	28
1.5.2	Prospective	29
	References	30
<b>2</b>	<b>Lattice Automata for Control of Self-Reconfigurable Robots</b>	<b>33</b>
	Kasper Stoy	
2.1	Self-Reconfigurable Robots	33
2.1.1	Origin, Features, and Applications	35
2.1.2	Mechatronic Implementation	36
2.2	Assumptions of Lattice Automata	37
2.3	Lattice Automata-Based Control	40
2.4	Hybrid Control	41
2.5	Conclusion	43
	References	44

<b>3</b>	<b>Modular Reconfigurable Robotic Systems: Lattice Automata . . . .</b>	<b>47</b>
	Nick Eckenstein and Mark Yim	
3.1	Introduction . . . . .	47
3.1.1	Motivation . . . . .	48
3.1.2	Key Terminology . . . . .	48
3.1.3	Environments . . . . .	50
3.2	Challenges and Practical Issues for MRR . . . . .	50
3.2.1	General Limitations . . . . .	50
3.2.2	Key Metrics . . . . .	51
3.2.3	Modular Robot Morphology—Shape and Connectedness . . . . .	52
3.3	Example Lattice System Hardware . . . . .	52
3.3.1	Key Designs . . . . .	52
3.3.2	Lattice Locomotion . . . . .	61
3.3.3	Connection Types . . . . .	63
3.4	Software Systems for MRR . . . . .	68
3.4.1	Reconfiguration Planning . . . . .	68
3.5	Assembly Robotics . . . . .	69
3.5.1	Self-Assembly and Self-Repair . . . . .	69
3.6	Conclusions and the Future of MRR . . . . .	72
	References . . . . .	72
<b>4</b>	<b>Lattice-Based Modular Self-Reconfigurable Systems . . . . .</b>	<b>77</b>
	Kohji Tomita, Haruhisa Kurokawa, Eiichi Yoshida, Akiya Kamimura, Satoshi Murata and Shigeru Kokaji	
4.1	Introduction . . . . .	77
4.2	Fractum . . . . .	79
4.2.1	Basic Design . . . . .	79
4.2.2	Algorithm I . . . . .	80
4.2.3	Algorithm II . . . . .	81
4.2.4	Meta Unit . . . . .	82
4.3	3D Units . . . . .	84
4.3.1	Design . . . . .	84
4.3.2	Reconfiguration . . . . .	84
4.4	M-TRAN . . . . .	86
4.4.1	Design . . . . .	86
4.4.2	Reconfiguration . . . . .	87
4.4.3	Robotic Motion . . . . .	88
4.5	General Problems of Lattice-Based Mechanical Systems . . . . .	90
4.5.1	Improvement in M-TRAN Hardware . . . . .	91
4.5.2	General, Physical Problem in Modular Reconfigurable Systems . . . . .	91
4.5.3	Achievement by M-TRAN . . . . .	94
4.6	Conclusions . . . . .	95
	References . . . . .	96

<b>5</b>	<b>Speed Control on a Hexapodal Robot Driven by a CNN-CPG</b>	
	<b>Structure</b> . . . . .	97
	E. Arena, P. Arena and L. Patané	
5.1	Introduction . . . . .	97
5.2	The Neural Network for Locomotion Control . . . . .	99
5.2.1	Leg Motor Neuron Network . . . . .	102
5.3	Reward-Based Learning for Speed Control . . . . .	103
5.4	Dynamic Simulator . . . . .	106
5.5	Simulation Results . . . . .	108
5.6	Conclusions . . . . .	113
	References . . . . .	115
<b>6</b>	<b>Routing by Cellular Automata Agents in the Triangular</b>	
	<b>Lattice</b> . . . . .	117
	Rolf Hoffmann and Dominique Désérable	
6.1	Introduction . . . . .	117
6.1.1	Cellular Automata Agents . . . . .	118
6.1.2	CA and CA-w Models . . . . .	119
6.1.3	Lattice Topology . . . . .	121
6.1.4	The Problem: Routing . . . . .	123
6.2	Minimal Routing in the Triangular Grid . . . . .	125
6.2.1	Topology of $S$ and $T$ . . . . .	125
6.2.2	Minimal Routing Schemes in $S$ and $T$ . . . . .	126
6.2.3	Computing the Minimal Route in $T$ (XYZ-Protocol) . . . . .	127
6.2.4	Deterministic Routing . . . . .	129
6.2.5	Adaptive Routing . . . . .	130
6.3	Modeling the Multi-Agent System . . . . .	130
6.3.1	Dynamics of the Multi-Agent System . . . . .	130
6.3.2	The CA-w and CA <i>Copy-Delete</i> Rules . . . . .	135
6.3.3	Programming Issues . . . . .	137
6.4	Router Efficiency and Deadlocks . . . . .	138
6.4.1	Efficiency of Deterministic Routing . . . . .	139
6.4.2	Efficiency of Adaptive Routing . . . . .	141
6.4.3	Deadlocks . . . . .	143
6.5	Summary . . . . .	144
	References . . . . .	145
<b>7</b>	<b>Multi-Resolution Hierarchical Motion Planner</b>	
	<b>for Multi-Robot Systems on Spatiotemporal</b>	
	<b>Cellular Automata</b> . . . . .	149
	Fabio M. Marchese	
7.1	Introduction . . . . .	149
7.2	MRS Motion Planning Problem . . . . .	150



7.3	Fine Motion Planner . . . . .	151
7.3.1	Spaces . . . . .	151
7.3.2	Motion and Moves . . . . .	154
7.3.3	Interaction Between Spaces and Moves (Planning) . . .	156
7.4	Gross Motion Planning . . . . .	158
7.4.1	Topological Approach . . . . .	158
7.4.2	Examples . . . . .	161
7.5	Multi-Robots Motion Problem . . . . .	166
7.6	Conclusions . . . . .	172
	References . . . . .	172
<b>8</b>	<b>Autonomous Robot Path Planning Techniques Using Cellular Automata . . . . .</b>	<b>175</b>
	Konstantinos Charalampous, Ioannis Kostavelis, Evangelos Boukas, Angelos Amanatiadis, Lazaros Nalpantidis, Christos Emmanouilidis and Antonios Gasteratos	
8.1	Introduction . . . . .	176
8.2	Theoretical Background . . . . .	178
8.2.1	Cellular Automaton Theory . . . . .	178
8.2.2	Path Planning Theory . . . . .	179
8.3	Local Path Planning . . . . .	180
8.3.1	Depth Map Acquisition . . . . .	180
8.3.2	Obstacle Free Ground Plane Modelling . . . . .	181
8.3.3	Polar Transformation of the Depth Map . . . . .	182
8.3.4	Floor Field . . . . .	182
8.3.5	Local Path Estimation . . . . .	184
8.4	Global Path Planning . . . . .	187
8.4.1	Operation in the Continuous Space . . . . .	187
8.4.2	From the Continuous to the Discrete Space . . . . .	188
8.5	Experimental Evaluation . . . . .	189
8.5.1	Local Path Planning . . . . .	189
8.5.2	Global Path Planning . . . . .	190
8.6	Discussion . . . . .	194
	References . . . . .	194
<b>9</b>	<b>Cellular Robotic Ants Synergy Coordination for Path Planning . . . . .</b>	<b>197</b>
	Konstantinos Ioannidis, Georgios Ch. Sirakoulis and Ioannis Andreadis	
9.1	Introduction . . . . .	198
9.2	Cellular Automata and Ant Colony Optimization Principles . . .	203
9.3	Cellular Ants: A Combination of CA and ACO Algorithms for Path Planning . . . . .	207
9.3.1	Proposed Method . . . . .	208

9.4	Simulation Results . . . . .	215
9.4.1	Implementation of the Method in a Simulated Cooperative Robot Team . . . . .	216
9.4.2	Simulator Results . . . . .	221
9.5	Conclusions . . . . .	225
	References. . . . .	225
<b>10</b>	<b>Employing Cellular Automata for Shaping Accurate Morphology Maps Using Scattered Data from Robotics’ Missions . . . . .</b>	<b>229</b>
	Athanasios Ch. Kapoutsis, Savvas A. Chatzichristofis, Georgios Ch. Sirakoulis, Lefteris Doitsidis and Elias B. Kosmatopoulos	
10.1	Introduction . . . . .	230
10.2	CA Based Methodology for Shaping Morphology Maps Using Scattered Data. . . . .	234
10.2.1	Problem Formulation. . . . .	234
10.2.2	Proposed Methodology . . . . .	234
10.2.3	Define Adaptively the “Radius of Influence” . . . . .	236
10.3	Experiments . . . . .	237
10.3.1	Underwater Scenario—Oporto harbor . . . . .	237
10.3.2	Aerial robots Scenario . . . . .	239
10.4	Conclusions and Future Work . . . . .	242
	References. . . . .	244
<b>11</b>	<b>On the Use of Cellular Automata in Vision-Based Robot Exploration . . . . .</b>	<b>247</b>
	Lazaros Nalpantidis	
11.1	Introduction . . . . .	247
11.2	Stereo Vision . . . . .	248
11.2.1	Algorithm Description . . . . .	249
11.2.2	Experimental Evaluation . . . . .	252
11.2.3	Discussion . . . . .	253
11.3	CA Refinement of Simultaneous Localization and Mapping . . . . .	254
11.3.1	SLAM Algorithm Description . . . . .	255
11.3.2	Experimental Evaluation . . . . .	261
11.3.3	Discussion . . . . .	264
11.4	Conclusion. . . . .	265
	References. . . . .	265

<b>12</b>	<b>Modelling Synchronisation in Multirobot Systems with Cellular Automata: Analysis of Update Methods and Topology Perturbations</b>	<b>267</b>
	Fernando Silva, Luís Correia and Anders Lyhne Christensen	
12.1	Introduction	267
12.2	Background and Related Work	269
12.2.1	Topology of the Environment	269
12.2.2	Update Methods	270
12.2.3	Modelling Individual Behaviour with Pulse-coupled Oscillators	271
12.3	Experimental Assessment	275
12.3.1	Characterising the Behaviour of Cellular Automata	275
12.4	Effects of the Update Method on Synchronisation of Behaviour	276
12.4.1	Methods	277
12.4.2	Results	277
12.4.3	Summary	285
12.5	Perturbing the Topology	285
12.5.1	Methods	285
12.5.2	Results	286
12.5.3	Summary	289
12.6	Discussion	290
	References	291
<b>13</b>	<b>Cellular Automaton Manipulator Array</b>	<b>295</b>
	Ioannis Georgilas, Andrew Adamatzky and Chris Melhuish	
13.1	Introduction	295
13.2	Cellular Automata Controller	296
13.3	Hardware Layer Role	298
13.4	Experimental and Simulation Results	298
13.4.1	Scenario 1: No Cilia	299
13.4.2	Scenario 2: Object with Cilia/Surface Without Cilia	301
13.4.3	Scenario 3: Object Without Cilia/Surface with Cilia	305
13.5	Conclusions	307
	References	307
	<b>Index</b>	<b>311</b>

# Chapter 1

## Algorithmic Insights into Finite-State Robots

Arnold L. Rosenberg

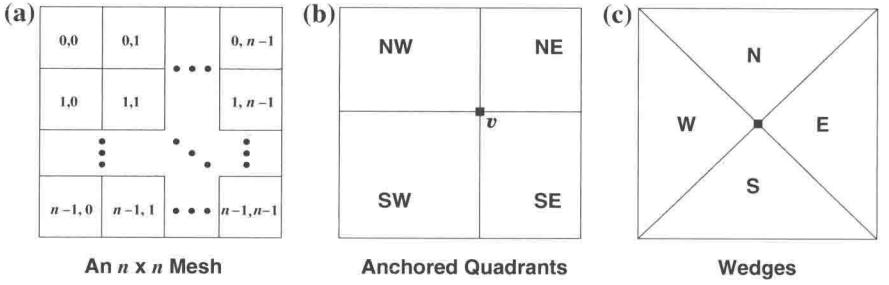
**Abstract** Modern technology has enabled the deployment of small computers that can act as the “brains” of mobile robots. Multiple advantages accrue if one can deploy simpler computers rather than more sophisticated ones: For a fixed cost, one can deploy more computers, hence benefit from more concurrent computing and/or more fault-tolerant design—both major issues with assemblages of mobile “intelligent” robots. This chapter explores the capabilities and limitations of computers that execute simply structured *finite-state programs*. The robots of interest operate within constrained physical settings such as warehouses or laboratories; they operate on tessellated “floors” within such settings—which we view formally as meshes of tiles. The major message of the chapter is that teams of (identical) robots whose “intellects” are powered by finite-state programs are capable of more sophisticated algorithmics than one might expect, even when the robots must operate: (a) *without the aid of centralized control* and (b) using algorithms that are *scalable*, in the sense that they work in meshes/“floors” of arbitrary sizes. A significant enabler of robots’ algorithmic sophistication is their ability to use their host mesh’s edges—i.e., the walls of the warehouses or laboratories—when orchestrating their activities. The capabilities of our “finite-state robots” are illustrated via a variety of algorithmic problems that involve path planning and exploration, in addition to the rearranging of labeled objects.

### 1.1 Introduction

Modern technology has enabled the deployment of small computers that can act as the “brains” of mobile robots. Multiple advantages accrue if one can deploy simpler computers rather than more sophisticated ones: For a fixed cost, one can deploy more computers, hence benefit from more concurrent computing and/or more fault-tolerant design—both major issues with assemblages of mobile “intelligent” robots. This chapter explores the capabilities and limitations of computers that execute simply

---

A.L. Rosenberg (✉)  
Northeastern University, Boston, MA 02115, USA  
e-mail: rsnbrg@ccs.neu.edu



**Fig. 1.1** **a** The  $n \times n$  mesh  $\mathcal{M}_n$ ; **b**  $\mathcal{M}_n$  partitioned into the four *quadrants* determined by *anchor tile*  $v$ ; **c**  $\mathcal{M}_n$  partitioned into its four *wedges*

structured *finite-state programs*, perhaps the simplest type of program that one can expect to enable sophisticated robot behavior. The robots of interest—which we call *finite-state machines* (FSMs, for short), to emphasize the finite-state property—operate within constrained physical settings such as warehouses or laboratories; they operate on tessellated “floors” within such settings—which we view formally as instances of the  $n \times n$  mesh of tiles  $\mathcal{M}_n$  (Fig. 1.1a).

The major message of the chapter is that teams of (identical) FSMs are capable of more sophisticated algorithmics than one might expect, even when the FSMs must operate: (a) *without the aid of centralized control* and (b) using algorithms that are *scalable*, in the sense that they work in meshes/“floors” of arbitrary sizes. A significant enabler of robots’ algorithmic sophistication is their ability to exploit the *edges* of the meshes they operate in—i.e., the walls of the warehouses or laboratories—when orchestrating their activities. The capabilities of finite-state robots are illustrated here via a variety of algorithmic problems that involve path planning and exploration, in addition to the rearranging of labeled objects (that sit within some tiles of the home mesh). We note again the major points that FSMs operate *without centralized control* while executing algorithms that are *scalable*.

Our study focuses on algorithmic problems that emerge from complementary avenues of investigation with histories that span several decades. The literature on automata theory and its applications contains studies such as [3, 5, 8, 10, 24, 27] that focus on the (in)ability of FSMs to explore graphs with goals such as finding “entrance”-to-“exit” paths or exhaustively visiting all of a graph’s nodes or all of its edges. Other studies, e.g., [4, 15, 17, 23, 26, 36], focus on algorithms that enable FSMs that populate the tiles of (multidimensional) meshes—*cellular automata*—to tightly synchronize, a crucial component of many activities that must be performed without centralized control; the cellular automaton model dates back a half-century [38] but remains of interest today [14, 39]. Yet other automata-theoretic studies update the historical string-recognition work of classical finite-automata theory—cf., [25, 28]—to more ambitious domains such as graphics [7, 13, 19, 20, 29]. The robotics literature contains numerous studies—e.g., [1, 2, 11, 18, 35]—that explore ants as a metaphor for simple robots that collaborate to accomplish complex tasks;

the interesting topic of “virtual pheromones” within this metaphor is studied in [11, 18, 31, 35]. Cellular automata appear in many studies of robotic applications of automata-theoretic concepts: application- and implementation-oriented studies as well as theoretical ones [6, 11, 16, 22, 32, 35, 37]. The current chapter melds the automata-theoretic and robotic points of view by studying FSMs that operate within square meshes; most of the problems we discuss are more closely motivated by robotics than automata theory, although a few emerge from the world of language-oriented studies.

The specific algorithmic challenges that we study are inspired by our earlier work on FSMs, which itself emerged from our work on the Cellular ANTOMaton model [32], a marriage of robotics and cellular automata. All of our studies demand algorithms that are *scalable* in the sense that they work in meshes  $\mathcal{M}_n$  of arbitrary size, i.e., for arbitrarily large values of  $n$ . Our first study involving FSMs was [31], which focused on the *Parking Problem* for FSMs; this problem requires each FSM in a mesh to go to its closest corner and has FSMs within each corner organize into a maximally compact formation (i.e., one that minimizes the FSMs’ aggregate distance to their nearest corners). A central component of parking is to have each FSM determine which *quadrant* of  $\mathcal{M}_n$  it resides in (cf. Fig. 1.1b); because the home-quadrant determination problem is treated in detail in [31], we focus here on a kindred, but rather different problem that requires FSMs to determine their home *wedges* (cf. Fig. 1.1c). Our next study of FSMs, in [33], allowed the tiles of  $\mathcal{M}_n$  to be labeled from a given repertoire. The study required FSMs to move to a specified tile  $v_{\varphi,\psi} = \langle \lfloor \varphi n \rfloor, \lfloor \psi n \rfloor \rangle$  of  $\mathcal{M}_n$ , identified by a prespecified pair of positive rational numbers  $\langle \varphi, \psi \rangle$ .

Note that: (a) the rational numbers  $\varphi = a/b$  and  $\psi = c/d$  are *fixed for each specific problem-instance*, and (b) the FSM  $\mathcal{F}$  that solves each instance of the problem is designed so that its state-memory “contains” the four integers  $a, b, c, d$ ; i.e.,  $\mathcal{F} = \mathcal{F}^{(a,b,c,d)}$ . The scalability in our problem solutions refers only to the mesh-size parameter  $n$ .

Every tile  $v_{\varphi,\psi}$  can serve as an *anchor* to induce a partition of  $\mathcal{M}_n$  into quadrants, as in Fig. 1.1b. The added challenge in [33] is to have the FSMs *sweep* the quadrants induced by  $v_{\varphi,\psi}$  to check that each of the mesh’s tiles contains a quadrant-specific label. In the third of our studies, [34], FSMs do more than plan application-specific trajectories and seek specified goal-tiles. They now rearrange objects that occupy  $\mathcal{M}_n$ ’s tiles in various prespecified ways while transporting the objects from  $\mathcal{M}_n$ ’s top row to its bottom row. The specific rearrangements include: (1) reversing the order in which objects appear, (2) cyclically rotating the objects, and (3) sorting the objects by their (ordered) “types.” In addition to being scalable, the algorithms we describe in [34] are *pipelineable* in a way that achieves parallel speedup that is asymptotically linear in the number of FSMs (even as that number approaches  $n$ ). The pipelining that we refer to here has a team of identical copies of an FSM  $\mathcal{F}$  march one after the other, performing different instances of the chores to be performed; see, e.g., [34] for details.

The problems we discuss in this chapter describe, and in several places extend or improve, the material in [32–34]. The problems we discuss require FSMs:

- to determine where they are within  $\mathcal{M}_n$ ;  
We focus on having FSMs determine which *wedge* of  $\mathcal{M}_n$  they reside in (cf. Fig. 1.1c). (Recall that we treat the analogous problem for quadrants in [31].)
- to seek various target tiles of  $\mathcal{M}_n$ ;  
We recapitulate the study in [33], wherein target tiles are specified via pairs of positive rational numbers, specifically using the rational pair  $\langle \varphi, \psi \rangle$  to specify tile  $\{\lfloor \varphi(n-1) \rfloor, \lfloor \psi(n-1) \rfloor\}$  of  $\mathcal{M}_n$ .
- to transport the objects residing in  $\mathcal{M}_n$ 's top row to  $\mathcal{M}_n$ 's bottom row while rearranging the objects in prespecified ways;  
Excerpting from our study in [34], we have FSMs (1) reverse the objects' original order, (2) cyclically rotate the original order, and (3) sort the objects by their (ordered) types.
- to determine whether the objects residing in certain of  $\mathcal{M}_n$ 's rows of tiles have certain patterns.  
We complement the study in [34] by having FSMs check the pattern of objects along  $\mathcal{M}_n$ 's rows rather than effect the pattern. We have FSMs identify *palindromes* (words that read the same forwards and backwards), *perfect squares* (even-length words whose first and second halves are identical), and *rotations* (a pair of words one of which is a cyclic rotation of the other).

The algorithmic tools employed by our FSMs extend to myriad other problems.

A final word of introduction. We noted earlier that various sources—e.g., [11, 18, 35]—discuss “virtual pheromones” as a control mechanism for robotic “ants.” This mechanism assigns registers within each robot’s internal computer to maintain levels of intensity of an array of pheromones, thereby implementing a digital analogue of the volatile organic compounds that are used by nature’s ants. We largely ignore “virtual pheromones” because FSMs do not need them to execute the algorithms we discuss. We note only that in [31] we have shown that “virtual pheromones” do not enhance the power of a single FSM—although they can sometimes be used to decrease the required size of an FSM, as measured in number of states.

## 1.2 Technical Background

### 1.2.1 FSM “Robots” and Their Domains

Our formal model of *FSM-robots* (FSMs, for short) is obtained by augmenting the capabilities of standard finite-state machines (sources such as [30] provide formal details) with the ability to travel around square *meshes* of *tiles*, possibly transporting *objects* from one tile to another (empty) one. We flesh out this informal description.

**Meshes.** We index the  $n^2$  tiles of the  $n \times n$  mesh  $\mathcal{M}_n$  by the set<sup>1</sup>  $[0, n-1] \times [0, n-1]$ ; see Fig. 1.1a. The set of tiles of  $\mathcal{M}_n$  that share the first index-coordinate  $i$ , i.e.,  $R_i \stackrel{\text{def}}{=} \{\langle i, j \rangle \mid j \in [0, n-1]\}$ , is the  $i$ th *row* of  $\mathcal{M}_n$ ; the set of tiles that share the second index-coordinate  $j$ , i.e.,  $C_j \stackrel{\text{def}}{=} \{\langle i, j \rangle \mid i \in [0, n-1]\}$ , is the  $j$ th *column* of  $\mathcal{M}_n$ . Tile  $\langle i, j \rangle$  of  $\mathcal{M}_n$  is:

- a *corner* tile if  $i, j \in \{0, n-1\}$ ;
- an (*internal*) *edge* tile if it is one of:
  - a *bottom* tile Meaning that  $i = 0$  and  $j \in [1, n-2]$ ;
  - a *top* tile Meaning that  $i = n-1$  and  $j \in [1, n-2]$ ;
  - a *left* tile Meaning that  $i \in [1, n-2]$  and  $j = 0$ ;
  - a *right* tile Meaning that  $i \in [1, n-2]$  and  $j = n-1$ ;
- an *internal* tile if  $i, j \in [1, n-2]$ .

We employ the *King's move* adjacency model for meshes, so named for the chess piece (also known as the Moore model). Under this model, each tile  $\langle i, j \rangle$  of  $\mathcal{M}_n$  has up to 8 neighbors, one in each compass direction, abbreviated (in clockwise order)  $N, NE, E, SE, S, SW, W, NW$ . Accordingly, each internal tile of  $\mathcal{M}_n$  has 8 neighbors; each (internal) edge tile has 5 neighbors; and each corner tile has 3 neighbors. Clerical modifications allow any *fixed finite* set of adjacencies, each specified by a pair of signed positive integers  $\langle \pm a, \pm b \rangle$ ; each such pair,  $\langle c, d \rangle$ , indicates that every tile  $\langle i, j \rangle$  of  $\mathcal{M}_n$  has a neighbor at index-point  $\langle i + c, j + d \rangle$ , as long as this point is a valid index for  $\mathcal{M}_n$ , meaning that both  $i + c$  and  $j + d$  are in the range  $[0, n-1]$ . One opts for program compactness at the cost of algorithmic efficiency by choosing a smaller repertoire of adjacencies, such as *NEWS* moves:  $N, E, W, S$  (which are also known as the von Neumann model); one opts for increased efficiency at the cost of larger programs by choosing a larger repertoire of adjacencies, such as the 16 *Knight's + King's* moves. These three alternatives are illustrated in Fig. 1.2, which depicts the world from the viewpoint of an FSM. Whichever adjacency model is implemented: *every edge of every tile  $v$  of  $\mathcal{M}_n$  is labeled to indicate which of  $v$ 's potential neighbors actually exist.* (This enables FSMs to avoid “falling off”  $\mathcal{M}_n$  or “banging into a wall.”)

$\mathcal{M}_n$ 's four *quadrants* are determined by lines that cross at an *anchor* tile  $v$  and are perpendicular to  $\mathcal{M}_n$ 's edges (Fig. 1.1b). The “standard” quadrants—which are anchored at  $\mathcal{M}_n$ 's “center” tile  $\langle \lfloor \frac{1}{2}(n-1) \rfloor, \lfloor \frac{1}{2}(n-1) \rfloor \rangle$ , hence are as close to equal in number of tiles as the parity of  $n$  allows—comprise the following sets of tiles.

<sup>1</sup> For positive integers  $i$  and  $j > i$ , we denote by  $[i, j]$  the set  $\{i, i+1, \dots, j\}$ .



Quadrant	Name	Tile-set
SOUTHWEST	$\mathcal{Q}_{SW}$	$\{(x, y) \mid x \geq \lfloor \frac{1}{2}(n-1) \rfloor; \ y \leq \lfloor \frac{1}{2}(n-1) \rfloor\}$
NORTHWEST	$\mathcal{Q}_{NW}$	$\{(x, y) \mid x < \lfloor \frac{1}{2}(n-1) \rfloor; \ y \leq \lfloor \frac{1}{2}(n-1) \rfloor\}$
SOUTHEAST	$\mathcal{Q}_{SE}$	$\{(x, y) \mid x \geq \lfloor \frac{1}{2}(n-1) \rfloor; \ y < \lfloor \frac{1}{2}(n-1) \rfloor\}$
NORTHEAST	$\mathcal{Q}_{NE}$	$\{(x, y) \mid x < \lfloor \frac{1}{2}(n-1) \rfloor; \ y < \lfloor \frac{1}{2}(n-1) \rfloor\}$

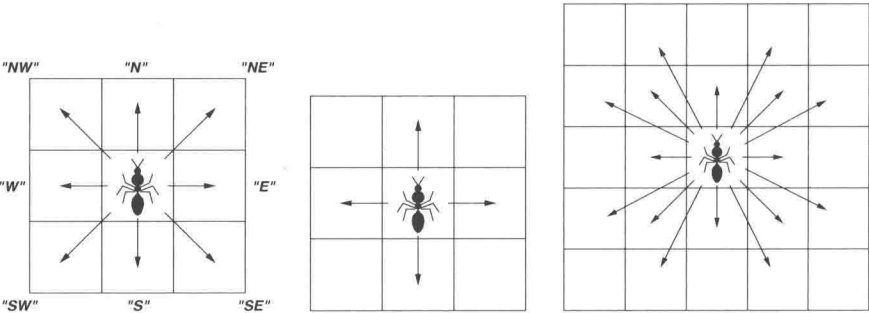
$\mathcal{M}_n$ 's four *wedges* are determined by passing lines with slopes  $\pm 1$  through  $\mathcal{M}_n$ 's "center" tile; see Fig. 1.1c. These lines come as close to connecting  $\mathcal{M}_n$ 's corners as the parity of  $n$  allows.  $\mathcal{M}_n$ 's wedges comprise the following sets of tiles.

Wedge	Name	Tile-set
NORTH	$\mathcal{W}_N$	$\{(x, y) \mid [x \leq y] \text{ and } [x + y \leq n - 1]\}$
SOUTH	$\mathcal{W}_S$	$\{(x, y) \mid [x > y] \text{ and } [x + y \geq n]\}$
EAST	$\mathcal{W}_E$	$\{(x, y) \mid [x \leq y] \text{ and } [x + y \geq n]\}$
WEST	$\mathcal{W}_W$	$\{(x, y) \mid [x > y] \text{ and } [x + y \leq n - 1]\}$

Rounding ensures that each tile has a unique home quadrant and home wedge.

Objects. Each tile  $v$  of  $\mathcal{M}_n$  can be empty—i.e.,  $v$  contains 0 FSMs and 0 objects—or it can hold at most one FSM and at most one object—i.e.,  $v$  contains 0 FSMs and 1 object *or* 1 FSM and 0 objects *or* 1 FSM and 1 object. Each object has a *type* chosen from some *fixed finite ordered* set. Because the number of objects can be commensurate with  $n$  while the number of object-types must be fixed independent of  $n$ , perforce, many objects can have the same type.

FSMs. At any moment, an FSM  $\mathcal{F}$  occupies a single tile of  $\mathcal{M}_n$ , possibly sharing that tile with an object but *not* with another FSM. At each step,  $\mathcal{F}$  can move to any neighbor  $v'$  of its current tile  $v$  (cf. Fig. 1.2), providing that  $v'$  contains no other FSM. Additionally, if  $v'$  contains no object, the  $\mathcal{F}$  can convey the object that resides on  $v$  (if there is one) to  $v'$ .



**Fig. 1.2** Single-step move repertoires for FSMs. (left) The King's-move repertoire; (center) the NEWS (North-East-West-South) repertoire; (right) the Knight's-move + King's-move repertoire