

Exploiting symmetry on parallel architectures

Lewis Benjamin Stiller

A dissertation submitted to The Johns Hopkins University in conformity with the
requirement for the degree of Doctor of Philosophy.

Baltimore, Maryland

1995

Copyright © 1995 by Lewis Benjamin Stiller,

All rights reserved.

Abstract

This thesis describes techniques for the design of parallel programs that solve well-structured problems with inherent symmetry.

Part I demonstrates the reduction of such problems to generalized matrix multiplication by a group-equivariant matrix. Fast techniques for this multiplication are described, including factorization, orbit decomposition, and Fourier transforms over finite groups. Our algorithms entail interaction between two symmetry groups: one arising at the software level from the problem's symmetry and the other arising at the hardware level from the processors' communication network.

Part II illustrates the applicability of our symmetry-exploitation techniques by presenting a series of case studies of the design and implementation of parallel programs.

First, a parallel program that solves chess endgames by factorization of an associated dihedral group-equivariant matrix is described. This code runs faster than previous serial programs, and discovered a number of results.

Second, parallel algorithms for Fourier transforms for finite groups are developed, and preliminary parallel implementations for group transforms of dihedral and of symmetric groups are described. Applications in learning, vision, pattern recognition, and statistics are pro-

posed.

Third, parallel implementations solving several computational science problems are described, including the direct n -body problem, convolutions arising from molecular biology, and some communication primitives such as broadcast and reduce. Some of our implementations ran orders of magnitude faster than previous techniques, and were used in the investigation of various physical phenomena.

Contents

Table of Contents	4
I Foundations	11
1 Introduction	12
1.1 Overview of thesis	16
2 Parallel computing	20
2.1 Parallel computing: Hardware	20
2.2 Parallel computing: Software	24
3 Group theory and parallel processing	29
3.1 Basic definitions	30
3.2 Groups and matrices	34
3.3 Cayley graphs and interconnection networks	37
3.4 Parallel group-theoretical algorithms	42
4 Linear algebra and parallel processing	46
4.1 Background	48
4.2 Graph problems and semirings	51
4.3 Tensors and programs	53

	5
4.3.1	Tensor products: Introduction 54
4.3.2	Code generation: Conversion from factorization to code 56
4.3.3	Example: Matrix multiplication by a tensor product 59
5	Exploiting symmetry: Mathematical framework 63
II	Applications 67
6	Orbit decomposition and its application to the analysis of chess endgames 68
6.1	Motivation and background 70
6.1.1	Search 70
6.1.2	Human analysis 74
6.1.3	Friedrich Amelung and Theodor Molien: A historical note 77
6.1.4	Computer endgame analysis 82
6.2	Tensor products and chess endgames 85
6.2.1	Definitions 87
6.2.2	Group actions 89
6.3	Endgame algorithm 90
6.3.1	Factorizing the unmove operator 90
6.3.2	Exploiting symmetry 93
6.3.3	Control structure 98
6.4	Implementation notes 99
6.4.1	Captures and pawns 99
6.4.2	Database 100
6.5	Results 102
6.5.1	Chess results 102
6.5.2	Timing 108
6.6	Future work 109
6.7	A best play line 111

7	Group fast Fourier transforms and their parallelization	114
7.1	Classical Fourier transforms	117
7.2	Group Fourier transforms: Foundations	121
7.2.1	Basic definitions	121
7.2.2	An algebra viewpoint	123
7.3	Fast group Fourier transform algorithms: Background	126
7.4	Parallel group Fourier transforms	129
7.4.1	Abelian case	129
7.4.2	General case: Background	130
7.4.3	A parallel algorithm for general groups	131
7.4.4	Group circulants	139
7.4.5	Applications	141
7.5	Dihedral group transforms	145
7.6	String matching	149
7.6.1	Background	149
7.6.2	Mathematical formulation	152
7.6.3	Reducing generalized matrix multiplication to matrix multiplication over \mathbb{C}	158
7.6.4	The application of group FFTs to parallel string matching	162
7.7	Future work	164
8	Equivariant factorization of Abelian groups	166
8.1	Fortran 90	167
8.2	n -body simulation	172
8.3	Parallel prefix and an application from computational biology	178
9	Conclusion and future work	183
A	List of symbols	186

Bibliography**189**

Acknowledgments

An interdisciplinary project, such as this one, depends for its success on the cooperation and assistance of many individuals and institutions.

Foremost, the author thanks his advisor, Simon Kasif, who patiently assisted the author in the formulation, presentation, and development of a number of ideas herein.

Noam Elkies provided invaluable chess advice and suggestions throughout the 6-piece project. He also designed the mutual-zugzwangs algorithm. Burton Wendroff made possible much of the work, and also developed and implemented the vectorized Cray algorithm.

The author's collaborators in the work on scientific computing, Luke Daemen, Angel García, and Jim Gubernatis, patiently explained the physics and chemistry background of their research. Hans Berliner and David Waltz were supportive of the chess work before any new results came out of it, and long before the 6-piece code was able to be run. Richard Draper helped to point the author in the direction of a tensorial approach to parallel computing, and helped the author understand more clearly the symmetry aspects of the work. Luke Daemen gave Cray implementations of important parts of the n -body code. Ken Thompson

provided numerous details and timings of his endgame code. John Roycroft provided much of the historical information about endgame analysis. S. Rao Kosaraju and Paul Callahan contributed useful ideas about string-matching.

The author's collaborators in the work on parallelization of shortest-word problems in group, Jon Bright and Simon Kasif, uncomplainingly did the lion's share of work on the paper.

The author is also grateful to the following for many useful conversations and suggestions: Alan Adler, Murray Campbell, Michael Fellows, Charles Fiduccia, Roger Frye, Ralph Gasser, Thomas Hawkins, Feng-Hsiung Hsu, S. Muthukrishnan, Daniel Rockmore, Steven Salzberg, Jonathan Schaeffer, and Johannes Tausch.

Assistance in translating many of the non-English sources was provided by Peter Jansen, Michael Klepikov, George Krotkoff, John Roycroft, Claudia Salzberg, Boris Statnikov, and the staff of the Cleveland Public Library.

Access to most of the manuscripts, rare books and chess journals cited in the paper was obtained during the author's visit to the John Griswold White Collection of Chess, Orientalia and Fine Arts, located at the Cleveland Public Library, 325 Superior Avenue, Cleveland, OH 44414. The author thanks Alice N. Loranth and Motoko B. Reece of the Cleveland Public Library for their assistance during his visits to the collection.

Harold van der Heijden graciously provided all studies in his database of approximately 36,000 endgame studies in which certain pawnless 6-piece endgames arose.

The National Library of Latvia in Riga (Latvijas Nacionālā Bibliotēka) assisted in providing

copies of a number of articles by Friedrich Amelung; Kenneth Whyld helped narrow down the search for these to the city of Riga.

Parallel and high-performance vector computer facilities were graciously provided by the Advanced Computing Laboratory of the Los Alamos National Laboratory, Los Alamos, NM 87545. The author was assisted in the benchmarking by application engineers from Thinking Machines Corporation. The work on parallel group Fourier transforms was supported in part by the Army Research Office contract number DAALO3-89-C-0038 with the University of Minnesota Army High Performance Computing Research Center (AHPARC) and the DoD Shared Resource Center at the AHPARC.

The author was supported during the writing of this thesis under U.S. Army Grant DAAL03-92-G-0345.

Part I

Foundations

Chapter 1

Introduction

Parallel processing has had a major impact on the development of computer science. There is an extensive literature on parallel architectures, parallel programming languages, and parallel algorithms, which has achieved insight into the capabilities and limitations of parallel machines. However, the diversity and complexity of parallel architectures have created a fundamental challenge for programmers, namely, the efficient mapping of algorithms to the parallel environment. Although efficient practical algorithms for the solution of many specific problems on particular parallel architectures are known, the development of a general methodology for the design of parallel algorithms that are practical and efficient on a wide variety of architectures is an active area of research.

This thesis is a step in that direction. We propose a paradigm for structuring and designing programs for parallel computers. Our paradigm is described informally and advocated by means of a collection of case studies on massively parallel architectures.

The intuitive motivation for our paradigm is that we are striving for a uniform method of eliciting the underlying algebraic structure in the problems to be solved. The existence of some algebraic structure is particularly important in the context of parallel computing, insofar as parallel computers typically perform more efficiently on highly structured computations than they do on unstructured computations.

Typically, the algebraic structure takes the form of a group of transformations that leave invariant salient computational characteristics defining the problem. In addition to parallel algorithms for exploiting this symmetry, a theme that permeates this thesis is the primacy of symmetry considerations in a broad range of applications, including string matching, particle simulation, and communication primitives.

Our method comprises the following two main steps:

First: The problem is translated into a generalized matrix multiplication problem; that is, a matrix multiplication problem in which addition and multiplication are replaced by more general operators. The formulation of a problem within the context of matrix multiplication has several advantages. For example, highly optimized linear algebra libraries for parallel machines can be used, and the considerable machinery of multilinear algebra techniques can often be brought to bear upon the new formulation.

Second: Symmetry in the original problem will be captured by symmetry in the matrix into which the problem has been translated; mathematically, the associated matrix will commute with a group of permutation matrices.

These invariant matrices admit several parallelizable techniques for their efficient multiplication.

The first technique is simply *factorization*; namely, we factor the matrix into simpler matrices, for example, into matrices that represent primitive operations supported by the target architecture. This technique emphasizes the use of the tensor product to extract parallelism, and was strongly influenced by the success of the tensor-product formulation of parallel signal-processing algorithms.

The second technique for the manipulation of invariant matrices is called *orbit decomposition*. Orbit decomposition is a formalization of the familiar technique of caching computations in order to reuse the data later. Orbit decomposition can sometimes induce a particular routing pattern in the parallel architecture: a Cayley graph. This Cayley graph is a graphical representation of the symmetry inherent in the original problem. As it happens, the network connecting the individual processors in a number of classes of parallel machines is also, in many cases, a Cayley graph; our methodology thereby gives rise to an interaction between a Cayley graph arising at the *software* level, from symmetry considerations, and a Cayley graph arising at the *hardware* level, from the interconnection network of the processors.

The third technique we use for manipulation of invariant matrices is group Fourier transforms. These generalize the familiar discrete Fourier transformations, the Cooley-Tukey implementation of which has been influential in many areas of computer science. Group Fourier transforms are based on techniques of group representation theory, which can be

loosely viewed as the use of matrices to model symmetry, and have undergone energetic development by a number of earlier researchers. This earlier work has demonstrated many applications for general group Fourier transforms, in areas such as machine learning, VLSI design, vision, random walks, and graph algorithms, and has provided fast algorithms for many classes of finite groups. Although some amount of mathematical machinery, primarily basic representation theory, is necessary to understand these fast algorithms, we have tried to carefully encapsulate areas of this thesis which require such knowledge. Their salient feature for our purposes is that they allow fast parallel algorithms for multiplication by invariant matrices. Work-efficient parallel group Fourier transform algorithms are introduced in this thesis, improving on several suboptimal constructions in the literature.

These three techniques—factorization, orbit-decomposition, and group Fourier transforms—are the tools we use to exploit symmetry.

Before continuing with the overview of the thesis, we emphasize two main points:

First, it should be clear that there are many classes of problems to which the paradigm we propose does not readily apply. For example, problems with unstructured data-access patterns, or data-access patterns that are not known at compile-time, would be a poor match for this approach. Typical examples of such problems include open-ear decomposition in graph problems, forward alpha-beta search with pruning heuristics, and several classes of combinatorial optimization problems. Nevertheless, our paradigm can be used to solve structured subproblems of an unstructured problem. For example, the adaptive multipole method is a dynamic tree algorithm for particle simulation to which we cannot usefully

apply our ideas, but even the adaptive multipole method must, at some point, call a direct “brute-force” particle simulation routine at which point our ideas apply. Similarly, although the parallelization of alpha-beta chess programs is beyond the scope of this work, the leaf-evaluation subroutines of such programs typically rely on a specialized endgame module of the type described in this thesis.

Second, we remark that the field of parallel processing is changing so fast, and with such a complex interconnection between economic, technological, hardware, and software factors, that we cannot claim that our paradigm is the last word on the topic. In Part II, therefore, we will not focus solely on our parallelization techniques. Instead, we will also describe a number of applications that, we hope, will be seen to have a beauty and interest independent of the ultimate viability of our main program; these case studies will also be useful considered only as examples of the successful design and implementation of parallel algorithms.

1.1 Overview of thesis

Part I presents the mathematical underpinnings of our work and gives some background on parallel processing.

Chapter 2 provides a bird’s-eye view of parallel processing. Section 2.1 describes a few common hardware configurations and introduces some interconnection networks. Section 2.2 discusses the problems of programming parallel machines and describes several parallel programming models.

Chapter 3 reviews basic group theory and examines the previous work on the relationship between groups and parallel processing, particularly the work on the application of Cayley graphs as the interconnection networks of a parallel machine. Some parallel group-theoretical algorithms, such as finding a composition series for a group, are also discussed. This chapter also describes related work by Bright, Kasif, and Stiller in the area of exhaustive search using group models, which resulted in the best known theoretical algorithm for parallel knapsack.

Chapter 4 reviews some basic linear algebra and its interaction with parallel processing, highlighting the role of the tensor product in expressing parallel and vector algorithms.

Chapter 5 concludes Part I. It describes the programming methodology whose application will be illustrated in Part II of this thesis.

Part II surveys several applications of the framework described in Part I.

Chapter 6 illustrates the *orbit decomposition*, which is appropriate when both the operator and the data are invariant. Both factorization and orbit-decomposition are used in a parallel program that solves chess endgames. The algorithm entails routing along the Cayley graph of the dihedral group of order 8, one of the smallest non-Abelian groups. Significant speedup over previous implementations on the small problems we studied was observed, but timing comparison for the larger instances versus other techniques is not available, since these problems are currently extremely time-consuming when solved using classical techniques. This chapter also contains a survey of new results discovered by the program, and presents historical information on the development of chess endgame analysis, some of which has

been published here for the first time.

Chapter 7 introduces group Fourier transforms and describes several applications. The group Fourier transform generalizes the classical Fourier transform to the case in which the indexing is performed over an arbitrary finite group. After recapitulating earlier work in the area, an efficient parallelization of group Fourier transform is presented, improving on previous parallel group Fourier transform algorithms. Potential applications for parallel algorithms in learning, VLSI design, pattern-matching, analysis of ranked data in statistics, random walks on groups, and graph theory are briefly presented. In order to illustrate a typical application, the parallel group Fourier transform algorithms are applied to generalized string matching problems, whose associated matrices have entries in a domain with little algebraic structure, and to which, therefore, group representation techniques do not directly apply. Preliminary massively parallel implementations of dihedral and symmetric group Fourier transforms are described.

Chapter 8 illustrates *factorization* by case studies of several scientific-computing applications. The symmetry groups that are considered in this chapter are Abelian, as contrasted with previous chapters, which consider non-Abelian symmetry groups. The specific implementations are:

1. The communication primitives from Fortran 90 are described within a multilinear-algebraic formulation, and it is shown how this formulation led to simple techniques for speeding them up in practice on several parallel architectures.

2. A parallel direct n -body solver, which uses special-purpose microcode and factorization techniques for the modeling of flux vortices in superconductors, is described. Our techniques resulted in speedup of approximately one order of magnitude compared to the previous methods in use for solving this problem on the CM-200.
3. The parallel prefix primitive, a common tool in parallel processing algorithms research, is developed within our framework, and the standard logarithmic time algorithm for parallel prefix is rederived. This derivation is then used in the design of an implementation for computing statistical properties of binary strings arising from a computational biology application. Once again, significant speedup was observed compared to previous methods.

Finally, Chapter 9 presents conclusions and ideas for future work.

Chapter 2

Parallel computing

2.1 Parallel computing: Hardware

Parallel computation refers to the utilization of multiple processors acting in concert to solve a single problem. The earliest reference known to this problem arguably comes from General L. F. Menabrea's 1842 commentary on the analytical engine, quoted in [374, p. 8]:

Likewise, when a long series of identical computations is to be performed, such as those required for the formation of numerical tables, the machine can be brought into play so as to give several results at the same time, which will greatly abridge the whole amount of the processes.

It has been argued as well that the ENIAC, the first general-purpose electronic computer, was highly parallel, insofar as it comprised a large number of independent functional units [339, 495].

In 1959, Holland proposed a collection of independently executing processors that foreshadowed many later developments [376]. The early RW-400 computer also used some

parallelism, as the following quotation from Porter's 1960 *Datamation* article illustrates:

The RW-400 Data System is a new design concept. It was developed to meet the increasing demand for information processing equipment with adaptability, real-time reliability and power to cope with continuously-changing information handling requirements. It is a polymorphic system including a variety of functionally-independent modules. These are interconnectable through a program-controlled electronic switching center. Many pairs of modules may be independently connected, disconnected, and reconnected, in microseconds if need be, to meet continuously-varying processing requirements. The system can assume whatever configuration is needed to handle problems of the moment. Hence it is best characterized by the term 'polymorphic'—having many shapes [606, pp.8–9].

The 1962 Conferences of the American Federation of Information Processing Societies contained a number of parallel computer designs, including the D825, a coarse-grained machine intended for use in military applications [52], and the highly influential SOLOMON system [697]. SOLOMON was intended primarily for use in military applications and utilized an array-based design comprising 2^{10} processors interconnected in a square grid. Its design was extremely influential on further SIMD architectures, such as the DAP and the ILIAC IV [137].

The mid-eighties saw the introduction of the Connection Machine family of massively-parallel architectures. The CM-1 and CM-2/200 each had up to 64K bit-serial processors, and their introduction and support software had a substantial impact on the development of the field. A number of other parallel processors have also been introduced, such as the N-Cube and the Intel Paragon. Several influential vendors have recently stopped producing massively-parallel machines, however, raising questions about the viability of very-large-scale parallel processing [438].

There are several taxonomies by which parallel computers are classified. Determinants characterizing current parallel processors are: MIMD vs. SIMD;¹ shared-memory vs. local-memory; the type of interconnection network used; the number and type of the individual processors.

There are also a large number of theoretical models which purport, to varying degrees, to represent the real complexity of algorithm execution on parallel architectures. These range from models in which communication overhead is entirely neglected, such as the CREW PRAM [786], to models in which the interconnection network becomes paramount, such as on arrays of automata [460]. Most of our results will be based on timing measurements on actual machines, and we will try to avoid excessive formality in the description of a theoretical model; although the algorithms we give are easily parallelizable on a PRAM, they are parallelizable on a wide class of other models as well.

For specificity, we briefly describe the architecture on which many of the algorithms described here were implemented: the CM-2/200 family. The CM-2 comprises 2^{16} bit-serial processors clocked at 7 MHz and driven from a front-end workstation. The CM-200 differs mainly in being clocked at 10 MHz. It comprises 2^{12} chips, interconnected in a 12-cube²

¹ Processors such as the CM-200 are SIMD: each processor executes the same set of instructions (the taxonomic distinction SIMD/MIMD/SISD/MISD seems to be due to Flynn [298]). The NCUBE, Intel Paragon, CM-5, and others are MIMD, in which each processor executes a different control thread; in many applications, however, these processors execute primarily in SIMD mode.

² A k -cube can be thought of as the set of binary strings of length k , with a direct communication link between any two nodes with unit Hamming distance. Alternatively, it is the vertices and edges

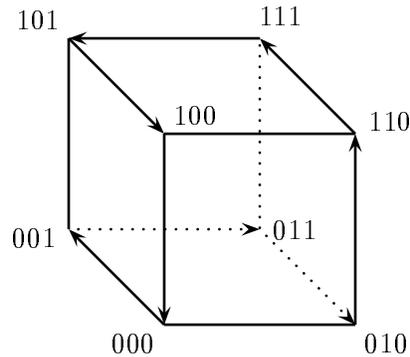


Figure 2.1: Embedding of a one-dimensional torus (i.e., a circle) by Gray coding the coordinates of each element of the grid. This figure illustrates a 3-bit Gray code, which can be thought of as an embedding of a cycle of length 8 into a 3-cube.

with 16 processors for chip. However, nearly all the system software supports the model of there being 2^{16} processors connected in a 16-cube, which is the model we shall assume for the purposes of this thesis.³ Each processor is driven from a single pipelined instruction stream, and contains up to 1 MBIT DRAM. The system-software supports virtual configuration of the processors in a k -dimensional torus using familiar techniques for embedding grids in hypercubes (see Figure 2.1) [333,371].

The CM-2 can also be configured in *slicewise mode*, which permits more efficient utilization of the Weitek floating point units. In this mode, the bit-serial structure is ignored, and the computer is envisioned as comprising 2^{11} 32-bit processors. This mode of operation is particularly interesting because both vectorization and parallelization issues must be

of the unit cube in Euclidean k -space.

³ The on-chip location of the processors only makes a difference in certain fairly obscure details of the chess endgame algorithm; this issue will be addressed in Chapter 6.

addressed in order to attain high bandwidth. Our scientific applications utilized slicewise mode, and our non-scientific applications used bit-serial mode.

The CM-2 is an extreme model of parallelism, insofar as it comprises large numbers of weak, SIMD processors. Therefore, successful implementation of an algorithm on the CM-2 tends to imply that the algorithm is parallelizable as well in stronger computational models, assuming the bandwidth and size is comparable. For example, the CM-5 is a MIMD architecture with a more powerful interconnection network and processors than the CM-2, but our results to apply to this kind of architecture as well [502, 503, 745]. Representative CM-5 and Cray timings are presented in Chapter 8 and CM-5 group Fourier implementations are described in Chapter 7.

2.2 Parallel computing: Software

As mentioned section 2.1, the ENIAC itself had a parallel design. It is interesting and instructive that the ENIAC was soon reconfigured to operate in serial mode, primarily because of the difficulty of its programming. In the words of Burks, one of the original designers:

The ENIAC's parallelism was relatively short-lived. The machine was completed in 1946, at which time the first stored program computers were already being designed. It was later realized that the ENIAC could be reorganized in the centralized fashion of these new computers, and that when this was done it would be much easier to put problems on the machine. . . Thus the first general-purpose electronic computer, built with a parallel decentralized architecture, operated for most of its life as a serial centralized computer [150] [374, p. 10].

Software issues remain the primary obstacle to greater penetration of highly-parallel computing. The plethora of variant high-performance architectures, combined with the rapidity with which architectures and their operating-system interfaces changes, has only increased the magnitude of the problem of attaining high-bandwidth on end-user applications: hence the so-called “parallel software crisis.” A number of tools and programming paradigms have been proposed to alleviate the parallel software crisis.

The parallel programming paradigm most relevant to the methodology advocated in this thesis is data-parallel programming. Data parallelism is a style of coding advocated particularly for SIMD massively parallel architecture; the term was coined in an influential 1986 paper by Hillis and Steele [367], who gave data-parallel solutions for several classic problems such as computing the sum of an array of numbers, general parallel prefix algorithms, regular-language parsing, and several graph algorithms.

The data-parallel coding paradigm is intended to act on large amounts of data with comparatively few threads of control. Many SIMD languages directly support this paradigm, such as CM-LISP [705], Paralation LISP [656], *LISP [743], C* [635], NESL [133], and ILIAS [522], and it is implicitly supported by the array functions of Fortran 90 [658]. Gary Sabot has discussed general methods for adding data-parallelism capabilities to an arbitrary language using the paralation model [657].

The success of the data-parallel paradigm inspired both the implementation of computer languages which directly supported it, and formal semantic treatment of data-parallel constructs.

There have also been a number of formal approaches to data parallel coding, many of which have been implicitly inspired by Backus' famous Turing award lecture, in which he advocated using algebraic primitives to model the interaction of complex operators [77]. Many of the attempts to formalize the data-parallel style have the flavor of a formal version of APL, which contained already most of the key data parallel programming constructs [395].

The Bird-Meertens theory of lists is one of the most well-known of these [127–130]. This posits as its fundamental data-type the list, and gives primitives, such as flatten, reduce, concatenation, and so on to manipulate them. It is possible to derive an algebraic theory of such operations of considerable expressiveness.

The Bird-Meertens theory of lists can be generalized to encompass the theory of categorical data-types [88, 692, 693, 695]. In the theory of categorical data-types, data values are presumed to take their values from a particular category, which is usually presumed to have some additional structure, such as being Cartesian closed. Many constructions in standard programming languages, such as ordered-pair and arrays of arbitrary data-type, can then be expressed as operations on the underlying category. Since programming language functions can be considered to be functors, algebraic tools can be utilized in deriving and verifying programs. There have also been a number of attempts to build a general theory of arrays [109, 404, 493, 568, 569], most of which are closely related to APL and to the Bird-Meertens theory of lists. Although these methodologies are extremely general and powerful, they tend to sacrifice peak performance.

Finally, it is worthwhile asking why the programmer should need to be concerned at all with the parallelization of the code. In fact, the most ambitious parallel programming paradigm is to use parallelizing compilers, which, ideally, reschedule and distribute the computation in such a way that the source code, which can be written as if for a sequential machine, executes efficiently on the target machine [806,829]. Parallelizing compilers include systems such as Fortran D [370], SUPERB [827], and Vienna Fortran [176,828]. A survey, including additional references, is in Amarasinghe et al. [25].

Current parallelizing compilers, however, are fairly limited in the kinds of code that they can efficiently parallelize [220,622,767,805]. Such compilers tend to perform better when the parallel structure of the algorithm is transparent in the source program.

Because our paradigm requires explicit restructuring of the algorithm by the programmer, it is much more limited in scope than a full parallelizing compiler would be. There are several reasons why this smaller goal was pursued, rather than striving directly to build a parallelizing compiler. First, many of the applications we considered were intended to be solved by end-users, such as physicists or scientists, and it was necessary to use the existing production languages, which require parallelization specified by the programmer, in order to solve their problems as rapidly as possible—for deriving fast programs on end-user machines, hand-parallelization is more effective than current parallelizing compilers. Second, parallelizing compilers produce much more efficient code when the algorithm used is as transparently parallel as possible, and this situation seems likely to continue in the near future. Third, we think that many of our techniques readily lend themselves to auto-

matic implementation, using group theoretical software such as GAP [669] and MAGMA, and, thus, our restructuring algorithms could perhaps be implemented as part of the loop-transformation phase of a parallelizing compiler [81,82].

Chapter 3

Group theory and parallel processing

Group theory provides a convenient and powerful language for the analysis of symmetry, and it therefore is one of the fundamental tools of this thesis. This chapter provides basic background material on group theory and also introduces the notion of a Cayley graph, which is a graph associated in particular way with a group. Cayley graphs provide a connection between group theory and graph theory, and have proven to be useful in the analysis of the interconnection networks on parallel processors; in fact we will see that many interconnection networks are Cayley graphs. The Cayley graphs that will arise in the applications discussed in Part II of the thesis, by contrast, will arise at the *software* level; the implementation of these algorithms on a parallel architecture therefore may entail an embedding of a Cayley graph arising from problem symmetry into a Cayley graph arising from the physical interconnection network of the parallel architecture.

3.1 Basic definitions

The material in this subsection comprises some standard facts on finite group theory, most of which are contained in elementary texts on the subject.⁴

A *group* is a triple $\langle \mathfrak{G}, \epsilon, \circ \rangle$ where \mathfrak{G} is a set and \circ is a binary, associative and invertible function $\circ: \mathfrak{G} \times \mathfrak{G} \rightarrow \mathfrak{G}$ with left and right identity $\epsilon \in \mathfrak{G}$.⁵

The history of the definition is long and fascinating. Some authors have claimed that group-theoretical ideas are implicit in symmetrical designs found in geometrical ornaments thousands of years ago. Manuel Moschopoulos used permutation-like operations in his 14th-century analysis of magic squares, and Levi ben Gershon, in “Practice of the Calculator,” 1321, computed the number of permutations of n elements. These early usages are so remote from contemporary understanding, and were so isolated from the mainstream of

⁴ The text by Wielandt on permutation groups is considered a classic, and is relevant to our work [802]. We found Rotman’s monograph to be clear and useful [641]. For more advanced references, particularly to applications of wreath products and multifarious applications to combinatorial problems, we recommend Kerber’s monograph [439]. Deep structural information is contained in the reference [64]. We will only need basic results from the representation theory of finite groups, for which Serre [678] is a standard textbook, and Collins [197] provides a careful and leisurely introduction. Fulton and Harris’ monograph is a lucid and apposite introduction to the topic [310]; see Coleman for a concrete descriptions of induced representations [195]. For a more abstract point of view, the standard reference is Curtis and Reiner [214].

⁵ We usually just call the group itself \mathfrak{G} , since the identity and law of multiplication will normally be clear from the context. Instead of writing $\circ(\mathfrak{g}, \mathfrak{h})$, for $\mathfrak{g}, \mathfrak{h} \in \mathfrak{G}$, we write $\mathfrak{g} \circ \mathfrak{h}$, or even $\mathfrak{g}\mathfrak{h}$.

development, however, that, in my opinion, they should not be considered to be relevant examples or discoveries of groups; if there is a relation to the modern conception of groups, it is an extremely tenuous one [809, pp.18–19]. The modern concept of an abstract group is more properly considered to have arisen from the work of Galois on permutation groups of roots of polynomials, as well as the work of Vandermonde, Lagrange, Ruffini, Abel, Serret, Jordan, Cayley, Klein, and others. For a complete discussion of the issues, the reader is referred to the texts [172, 792, 809].

A group \mathfrak{G} *acts* on a finite set X if to each $\mathfrak{g} \in \mathfrak{G}$ and $x \in X$ is associated a unique $\mathfrak{g}x \in X$ such that $\epsilon x = x$ and $\mathfrak{g}(\mathfrak{g}'x) = (\mathfrak{g}\mathfrak{g}')x$.

If $\mathfrak{H} \subset \mathfrak{G}$ and if \mathfrak{H} forms a group under the inherited multiplication in \mathfrak{G} then \mathfrak{H} is called a *subgroup* of \mathfrak{G} . The group *generated* by a set of elements in \mathfrak{G} is the intersection of all subgroups of \mathfrak{G} containing that set.

Because each \mathfrak{g} in a group acting on X induces a permutation of the elements of X , in this case one can think of \mathfrak{G} as being a permutation group, that is, a group whose elements are permutation and whose multiplication is composition. Although (in accordance with modern conventions) we have defined the notion of action in terms of the abstract axiomatization of group, historically the notions of a permutation and transformation group long preceded any explicit axiomatization [809, pp.230–251]. We will often elide the distinction between the elements of a group and the transformations of X that they induce.

The notions of group and group action are closely associated with the concepts of symmetry and of invariance. Informally, consider a property that may be possessed by the elements

of a set X . Let S be a set of one-to-one transformations from the set to itself each member of which sends any element in the set to another element that has the given property if and only if the first element does. For example, suppose that X is the set of vertices of the unit cube in Euclidean 3-space \mathbb{R}^3 , thus $X = \{(i, j, k): i, j, k \in \{0, 1\}\}$. Color the 4 bottom vertices, $\{(i, j, 0)\}$, red, and color the 4 top vertices blue. Let τ be the operation of rotating the cube counterclockwise 90° about its vertical axis (the line parallel to the z -axis and passing through $(\frac{1}{2}, \frac{1}{2}, 0)$), and let \mathfrak{f} be the operation of reflecting the cube about the plane parallel to the yz -axis and bisecting the cube (i.e., the plane normal to the x axis and passing through the point $(\frac{1}{2}, \frac{1}{2}, 0)$). Then it is easy to see that τ and \mathfrak{f} induce one-to-one mappings from X to itself, and that each sends a red vertex to another red vertex, and a blue vertex to another blue vertex. Therefore, the property of vertex color is invariant under \mathfrak{f} and τ , so that it is invariant under the group generated by the $\{\tau, \mathfrak{f}\}$, which, by the way, is a group of 8 elements called \mathfrak{D}_4 .

If $x \in X$ and \mathfrak{G} acts on X then $\mathfrak{G}x := \{\mathfrak{g}x: \mathfrak{g} \in \mathfrak{G}\}$ is called the *orbit* of x , and $\mathfrak{G}_x := \{\mathfrak{g} \in \mathfrak{G}: \mathfrak{g}x = x\}$ is called the *stabilizer* of x . Any two orbits are disjoint or identical, and the number of elements (order) in $\mathfrak{G}x$ is the order of the group divided by the order of the stabilizer of x . If $\mathfrak{G}_x = \mathfrak{G}$ then x is called *invariant* or *equivariant*. The set of orbits is called the *orbit space* X/\mathfrak{G} . The size of the orbit space might be much less than the size of X , and, in the presence of symmetry, computations on the domain X can often be replaced by equivalent computations on the domain X/\mathfrak{G} . This computational savings represents an important motivation for considering symmetry.

An action that has only one orbit is called *transitive*. In the example above, the \mathfrak{D}_4 action has two orbits, namely the set of red vertices and the set of blue vertices. The stabilizer $\mathfrak{D}_{4\mathbf{v}}$ of any vertex \mathbf{v} contains 2 elements, the identity and a reflection about some plane; however, these stabilizers can be different as \mathbf{v} varies. For example, $|\mathfrak{D}_4| = 8$, and, for any vertex, $|\mathfrak{D}_{4\mathbf{v}}| = 2$; there are, thus, $\frac{8}{2} = 4$ elements in the orbit of \mathbf{v} , which, of course, is correct.

Any subgroup $\mathfrak{h} \subset \mathfrak{G}$ acts on \mathfrak{G} by left multiplication. The orbit of a group element $\mathfrak{g} \in \mathfrak{G}$ is $\mathfrak{h}\mathfrak{g}$ and is called a (right) coset of x and \mathfrak{h} ; the number of these cosets is $\frac{|\mathfrak{G}|}{|\mathfrak{h}|}$. The *conjugate* of \mathfrak{g} by \mathfrak{h} is $\mathfrak{h}^{-1}\mathfrak{g}\mathfrak{h}$; a group acts on itself by conjugation, as well as on its set of subgroups. Any subgroup fixed by the conjugation action is called *normal*. The cosets of a normal subgroup \mathfrak{h} form the *quotient group* $\mathfrak{G}/\mathfrak{h}$ under the natural multiplication of cosets $\mathfrak{g}\mathfrak{h} \circ \mathfrak{g}'\mathfrak{h} = (\mathfrak{g}\mathfrak{g}')\mathfrak{h}$.

If \mathfrak{G} and \mathfrak{G}' are two groups, then a homomorphism $f: \mathfrak{G} \rightarrow \mathfrak{G}'$ is a function from \mathfrak{G} to \mathfrak{G}' that sends the identity of \mathfrak{G} to the identity of \mathfrak{G}' and for which $f(\mathfrak{g} \circ \mathfrak{h}) = f(\mathfrak{g}) \circ f(\mathfrak{h})$. If f is one-to-one and onto then it is called an *isomorphism* and \mathfrak{G} and \mathfrak{G}' are said to be *isomorphic*. Because an isomorphism of groups preserves group structure, it can be thought of as a renaming of the group elements, and isomorphic groups, therefore, are not normally considered to differ in any essential way.

The *direct product* of groups $\mathfrak{G} \times \mathfrak{G}'$ is the group of ordered pairs of elements from \mathfrak{G} and \mathfrak{G}' , with multiplication defined component-wise:

$$(\mathfrak{g}, \mathfrak{g}') \circ (\mathfrak{h}, \mathfrak{h}') = (\mathfrak{g} \circ \mathfrak{g}', \mathfrak{h} \circ \mathfrak{h}')$$

The *cyclic group* \mathfrak{C}_n is any group isomorphic to the group of integers modulo n , \mathbb{Z}_n under addition. The *symmetric group* \mathfrak{S}_n is the group of permutations of the integers $\{0, 1, \dots, n - 1\}$ with multiplication given by composition of permutations.

An *Abelian group* is a group whose multiplication operation is commutative: $\mathfrak{g}\mathfrak{g}' = \mathfrak{g}'\mathfrak{g}$ for all $\mathfrak{g}, \mathfrak{g}' \in \mathfrak{G}$. It can be shown that any finite Abelian group is isomorphic to a direct product of cyclic groups.

Now, suppose that \mathfrak{G} acts on two sets X and Y , and that there is a function $f: X \rightarrow Y$. We say that f is *equivariant* (or *invariant*) if $f(\mathfrak{g}x) = \mathfrak{g}f(x)$ for all $x \in X$ and $\mathfrak{g} \in \mathfrak{G}$. This condition can also be phrased by saying that f commutes with \mathfrak{G} . This definition is important because it formalizes the notion that f “respects” any symmetry in X . If f is equivariant then f induces a well-defined function $f/\mathfrak{G}: X/\mathfrak{G} \rightarrow Y/\mathfrak{G}$, and so f may be replaced, in a way, with the function with a smaller domain.

It is worth remarking that the notion of an invariant f is a special case of the notion of an invariant point of a group action; \mathfrak{G} acts on the set of functions Y^X via conjugation, and f is an invariant of that action.

3.2 Groups and matrices

Recall that a *vector space* \mathfrak{V} over the complex numbers \mathbb{C} is an Abelian group on which the additive and multiplicative groups of \mathbb{C} each act and for which $r(\mathbf{v} - \mathbf{w}) = r\mathbf{v} - r\mathbf{w}$ and $r\mathbf{0} = \mathbf{0}$, where $r \in \mathbb{C}$ and $\mathbf{0}, \mathbf{v}, \mathbf{w} \in \mathfrak{V}$. We consider only finite dimensional vector-

spaces here, each of which has a basis of n elements whose linear combinations uniquely generate the space. The linear transformations from a vector space \mathfrak{V}_n of dimension n to a vector space \mathfrak{V}_m of dimension m is, given a basis for the spaces, uniquely associated with an invertible $n \times m$ complex matrix \mathbf{M} . In the sequel, therefore, we will frequently identify a matrix and its associated linear transformation, and we will frequently identify n -dimensional spaces of \mathbb{C} with complex n -tuples.

Let $\{\mathbf{e}_i^n\}_{i=0}^{n-1}$ and $\{\mathbf{e}_i^m\}_{i=0}^{m-1}$ be bases for \mathfrak{V}_n and \mathfrak{V}_m respectively. The *direct sum* $\mathfrak{V}_n \oplus \mathfrak{V}_m$ is the $n+m$ dimensional vector space of ordered pairs of elements from \mathfrak{V}_n and \mathfrak{V}_m . The *tensor product* $\mathfrak{V}_n \otimes \mathfrak{V}_m$ is the mn -dimensional space whose basis elements are $\{\mathbf{e}_i^n \otimes \mathbf{e}_j^m\}_{i,j=0}^{n-1,m-1}$ [530, 531]. The *tensor power* $\otimes^j \mathfrak{V}$ is $\mathfrak{V} \otimes \cdots \otimes \mathfrak{V}$, with j factors. The *symmetric power* $\text{Sym}^j \mathfrak{V}$ is the subspace of $\otimes^j \mathfrak{V}_n$ whose basis is given by all elements of the form

$$\left\{ \mathbf{e}_{i_1}^n \otimes \cdots \otimes \mathbf{e}_{i_j}^n : 0 \leq i_1 \leq i_2 \leq \cdots \leq i_j \leq n-1 \right\}.$$

Let \mathfrak{M}_m^n be the set of $m \times n$ complex matrices, and let $\mathbf{M} \in \mathfrak{M}_m^n$ be an $m \times n$ matrix, $\mathbf{M} = (\mathbf{M}_{ij})_{i=0,j=0}^{m-1,n-1}$. Let us write I for the index set $0, 1, \dots, m-1$ of the rows of \mathbf{M} , and J for the index set $0, 1, \dots, n-1$ of the columns. Now suppose that \mathfrak{G} acts on both I and J . We say that \mathbf{M} is \mathfrak{G} -equivariant if

$$(\forall i \in I) (\forall j \in J) (\forall \mathfrak{g} \in \mathfrak{G}) \mathbf{M}_{ij} = \mathbf{M}_{\mathfrak{g}i, \mathfrak{g}j}.$$

If $I = J = \mathfrak{G}$ and \mathfrak{G} acts on itself by left multiplication, then a \mathfrak{G} -equivariant matrix is called a \mathfrak{G} -circulant. A \mathfrak{C}_n -circulant is thus the usual circulant matrix—constant on the diagonals with wraparound [223].

In order to see the relationship between the definition of \mathfrak{G} -equivariant matrices and our earlier definitions, we define the concept of group representation.

Let \mathfrak{V} be a vector space over \mathbb{C} of dimension n ; \mathfrak{V} can be thought of as the space of n -tuples of complex numbers. An invertible linear transformation from \mathfrak{V} onto itself corresponds to a nonsingular matrix in a natural way, and the linear transformation and its associated matrix will often be identified in the sequel. The nonsingular $n \times n$ matrices form a group GL_n under matrix multiplication. A group homomorphism ρ from \mathfrak{G} to GL_n is called a *representation* of \mathfrak{G} of degree n . The matrix corresponding to the linear transformation that permutes the basis elements has precisely one “1” in each row and column with its other entries being 0; such a matrix is called a permutation matrix. A representation ρ such that $\rho(\mathfrak{g})$ is always a permutation matrix is called a permutation representation. A representation can also be thought of as an action by a group \mathfrak{G} on \mathfrak{V} in which the map induced by each $\mathfrak{g} \in \mathfrak{G}$ is a linear transformation of \mathfrak{V} (or an $n \times n$ matrix).

Now given two permutation representations of \mathfrak{G} , ρ and σ , of degrees n and m with associated spaces \mathfrak{V} and \mathfrak{W} respectively, and given a linear map $M: \mathfrak{V} \rightarrow \mathfrak{W}$, the following are equivalent:

- M is equivariant with respect to the \mathfrak{G} actions on \mathfrak{V} and \mathfrak{W} .
- $M \cdot \rho(\mathfrak{g}) = \sigma(\mathfrak{g}) \cdot M$, for all $\mathfrak{g} \in \mathfrak{G}$.
- M is \mathfrak{G} -equivariant.

Representation theory will be discussed in more detail in Chapter 7, where it will be used in the context of fast group Fourier transforms.

3.3 Cayley graphs and interconnection networks

Group theory has been used extensively in the design and modeling of interconnection networks for parallel computers. These interconnection networks can connect processors to memory modules or, in the cases we consider here, processors to one another.

An interconnection can be viewed graphically as follows: each processor, together with its local memory, is represented as a vertex in the interconnection graph, and wires physically linking processors are represented as edges connecting the corresponding vertices. More complicated models are also possible, in which some vertices represent, for example, switches.

It is often of interest to know how difficult it is to route a given permutation of the processors given a specified interconnection network. In a series of classic papers from the early 1960s, Vaclav E. Beneš, whose original motivation was the analysis of telephone switching networks, showed that group theoretical concepts could be used to analyze the set of permutations routed by a network [102–104].

The *Cayley graph* of a group \mathfrak{G} with respect to a subset $S \subset \mathfrak{G}$ is a directed graph in which each edge is colored from a set of $|S|$ colors [800]. The vertices of the graph are the elements of \mathfrak{G} , and there is an edge from vertex \mathfrak{g} to vertex \mathfrak{g}' whenever there is some $\mathfrak{h} \in S$ such

that $hg = g'$; in this situation, the edge between g and g' is colored h . In practice the term Cayley graph is normally reserved for the situation when S generates \mathfrak{G} , which is equivalent to requiring that the Cayley graph be connected; this is reasonable, since a disconnected interconnection network is rarely useful in a multiprocessing environment.

Beginning from at least the 1960s a number of interconnection networks were proposed and implemented. These interconnection networks were designed to minimize cost and maximize the number of permutations that could be routed within a fixed time. As early as 1984, Carlsson, Sexton, Shensa, and Wright showed that Cayley graphs could be used to model interconnection networks [162]. Later, Carlsson, Cruthirds, Sexton, and Wright observed that important classes of networks were Cayley graphs and thereby were able to simplify analysis and modeling of networks [160, 163].

Group theory was explicitly proposed as a foundation for the modeling and analysis of interconnection networks in an unpublished manuscript of Carlsson, Fellows, Wright and Sexton (1985) [161], where the utility of a group-theoretical approach to network design, network description, network simulation, and scheduling was advocated. Fellows' 1985 dissertation discussed many of these matters in more detail, including analyses of families of interconnection networks including hypercubes, tori, cube-connected cycles, butterfly networks [282].

Akers and Krishnamurthy (1984, 1987) [14, 15] observed that group graphs had good fault-tolerance capabilities.

We now consider several concrete examples of Cayley graphs.

The hypercube was introduced in section 2.1. It is a popular interconnection network in which processors are at the vertices of a unit cube in \mathbb{R}^k . In fact, the hypercube of dimension k is simply the Cayley graph for the cross product of k copies of the cyclic group \mathcal{C}_2 with respect to the generating set $\{(1, 0, 0, \dots, 0), (0, 1, 0, \dots, 0), \dots, (0, \dots, 0, 1)\}$.

A Cayley graph of a non-commutative group is shown in, Figure 3.1, where the Cayley graph for the order 8 dihedral group \mathcal{D}_4 (see section 3.1) with respect to the generators $\{\tau, f\}$. The problem of embedding this \mathcal{D}_4 Cayley graph into a hypercube arises in Chapter 6; (compare figure 6.4).

The cube-connected cycles, which were shown to be capable of a wide class of computations, particularly the so-called ASCEND-DESCEND algorithms (which include fast Fourier transforms) in the early 1980s by Franco P. Preparata and Jean Vuillemin [610] in a famous paper, are wreath products of cyclic groups.

Fred Annexstein, Marc Baumslag, and Arnold L. Rosenberg generalized the Cayley graph construction to group action graphs. If \mathcal{G} acts on X , and if $S \subset \mathcal{G}$ generates \mathcal{G} , then group action graph corresponding to \mathcal{G}, S and X is the graph whose nodes are the elements X and for which, for each $x \in X$ and $g \in S$, there is an edge from x to gx . They studied many embedding, routing, and simulation problems and presented formal methods for simulating the certain kinds of group action graphs by others; they also studied the de Bruijn [609] and shuffle-exchange networks [721], in addition to the networks described above [54–56, 636].

Based on this early work, a wide class of Cayley graphs (and group action graphs) has been proposed as possible interconnection networks. Richard N. Draper initiated the study of

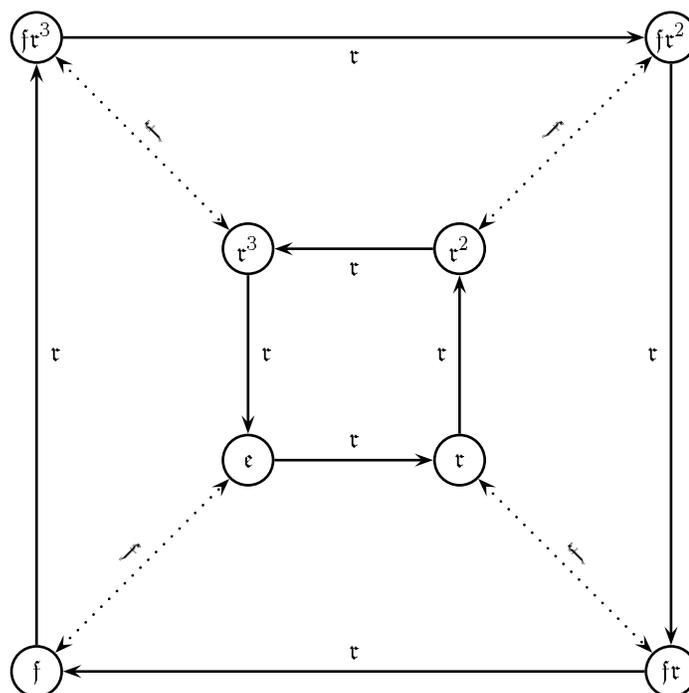


Figure 3.1: Cayley graph for \mathcal{D}_4 . Solid lines correspond to rotation 90° and dotted lines correspond to reflection about the horizontal bisector. The generators are $\{t, f\}$ with relations $\{t^4 = e, f^2 = e, ft = f^3t\}$. Each node is labeled with a word whose product is the group element at that node.

supertoroidal networks, which are networks that are the Cayley graph of the semidirect product of cyclic groups [248]. Routing algorithms and simulation algorithms were given; the methods display an interesting mixture of group-theoretical, graph-theoretical, and computational techniques [147, 148, 245, 246]. The twisted hypercube network has also been shown to be a group graph [261, 267, 826].

Cayley graphs have been particularly studied from the point of view of being classes of graphs with small diameter, a common concern in interconnection network design [184, 250, 274]. Lowell Campbell et al. have shown that even almost “randomly chosen” Cayley graphs of non-Abelian linear groups (groups of matrices whose elements lie in a finite fields) have surprisingly low diameter for their size [156].

Other applications of Cayley graphs, including some to game-playing, are described in Cooperman, Finkelstein, and Sarawagi [202].

Our work differs from much of the work on interconnection networks insofar as we are concerned primarily with Cayley graphs that arise at the *software* level. Our Cayley graphs arise when the problem to be solved is phrased as a \mathfrak{G} -equivariant matrix \mathbf{M} ; it will be seen to be necessary to embed this Cayley graph into the target architecture, which may have an interconnection network that is also a Cayley graph, but normally of a group that has no relation to the group of symmetries of the problem (see Chapters 6 and 7) [708]. The embedding of Cayley graphs in Cayley graphs is a special case of the problem of finding Hamilton cycles in Cayley graphs [55], and is related to the class of problems analyzed by Fellows in his dissertation [282].

On the other hand, it is worth noting that there are many classes of architectures which are not Cayley graphs or group action graphs. The most notorious of these is, perhaps, the definitely unsymmetric Internet architecture. The fat-tree topology of the CM-5 is another well-known example [502,503]. Many of the graphs considered in the theoretical community, such as the immensely complex network of expanders used in the sorting algorithm of Ajtai, Komlós and Szemerédi, are highly nonsymmetric as well. An analysis of relationships between group theoretical techniques and expander graphs is contained in the survey article by Bien (1989) [124].

3.4 Parallel group-theoretical algorithms

We now briefly discuss the subject of parallelization of computations in finite groups.

There is a considerable body of literature for sequential algorithms for computation in finite (permutation) groups [152,684], particularly permutation groups, which we can only very briefly touch upon here. The kinds of questions that have been addressed in the sequential literature involve finding structural aspects of a group (such as finding a composition series, a Sylow subgroup, the centralizer, the commutator subgroup, and so on) and solving word problems in the group, particularly testing the membership of a permutation in a permutation group. These algorithms tend to rely on finding a sequence of generators of the group element for which any group element has a reasonably short and effectively computable representation [251]. The notion of a strong generating set has proven to be especially useful for such computations [144,203,405]. Group-theoretical algo-

rithms have been especially important in two applications: polynomial-time testing of graph isomorphism [70,317,523] and exploiting symmetry in backtrack searches of combinatorial objects [143,153,266,479,483,508]. The proof of the nonexistence of a projective plane of order 10 was one of the most dramatic illustrations of the utility of the “isomorph rejection” method in backtrack search [451,481,482]. Many of these algorithms have been implemented in the two premier contemporary group-computation systems: GAP and MAGMA (MAGMA is the latest version of Cannon’s famous CAYLEY program).

The first parallel algorithm for permutation group algorithms seems to be implicit in the logspace solutions of certain word problems due to Lipton and Zlaczstein in 1977 [515]; there is a close connection between problems solvable in log space and parallelizable algorithms in certain theoretical models of parallelization. In a classic and fairly deep paper by László Babai, Eugene M. Luks, and Ákos Seress, it was shown that a number of permutation group algorithms were in NC, that is, they could be solved in polylogarithmic time on a polynomial number of processors [71]. The algorithms include finding the order of a permutation group, finding the center, and so on, and rely on the classification theorem for finite simple groups. S. Waack has explored parallel complexity of classes of linear groups [791] and surveys other results in [790], and Jin-Yi Cai has given efficient algorithms for another class of groups [154]. However group-theoretical algorithms are notoriously difficult to analyze because of their sensitivity to choice of data representation and distribution; this matter is discussed in more detail in Sims’ monograph [685]. In the parallel case the situation is even trickier to analyze because of the added complexity of accounting for different architectures and data-layouts.

In a more computational vein, we remark that even the backtrack searches by Clement Lam et al. for projective planes of a given order were concerned with vectorization, and many of their speedup techniques would apply in a parallel environment as well [482, 741]. Group-theoretical algorithms were implemented on a Connection Machine by Bryant W. York and Ottorino Ori (1993) and used to enumerate buckminster fullerenes [817]. York et al. have exploited potential parallelism in group-theoretic computation, which, roughly speaking, can come at a high level from parallelization of the main algorithm itself, or at a low level from parallelizing the manipulations of individual large permutations, with particular reference to the Connection Machine family [204, 474, 816].

Finally, in joint work with Jonathan Bright and Simon Kasif, the author has explored parallel algorithms for solving the shortest word problem in groups.⁶ [142] This problem is easy to phrase. The input is a finite set S of permutations from a permutation group \mathfrak{G} and an additional permutation \mathfrak{g} . The output is the shortest expression of \mathfrak{g} as a composition of permutations from S . For example, if S was the set of permutations corresponding to single moves applied to a Rubik's cube, and \mathfrak{g} was the permutation corresponding to a particular mixed-up state of the cube, then the algorithm would output, in theory, the shortest sequence of moves that would unmix the cube. The method used was a fairly straightforward parallelization of the sequential algorithm Fiat, Shamir, Moses, and Shimshoni [289]. This algorithm is, in turn, a generalization to non-Abelian groups of the well-known algorithm of Schroepel and Shamir for solving certain classes of NP-complete problems [671]. The idea

⁶ Of course this problem, in general, is recursively unsolvable.

is fairly simple and can be, perhaps, most easily understood through an example. Suppose we wish to find out if a particular state of Rubik's cube can be solved in 20 moves. We generate all states that can be reached in 10 moves from the initial state, and all states that can be reached in 10 moves from the target state, and see if they intersect; this is a parallelization of bi-directional search [604]. Each of these two sets can be generated in sorted order by computing the permutations corresponding to all states reachable within 5 moves, and composing them appropriately; a parallel algorithm only needs to be careful about generating the sets in parallel. Surprisingly, this simple idea, when applied to the case of an Abelian group, resulted in the fastest known parallel algorithm from a theoretical point of view for the knapsack problem, into the parallelization of which considerable effort had been directed [174, 177, 285–287, 423, 498, 513].

Chapter 4

Linear algebra and parallel processing

In order to apply group-theoretical ideas in a uniform setting, the methodology advocated in this thesis first attempts to convert the problem into a linear algebra setting, namely to the multiplication by a matrix \mathbf{M} over an appropriate domain. There are a number of potential advantages and disadvantages inherent in this formulation. First, we consider some potential disadvantages:

1. No such formulation may be possible. For example, the problem of playing full-chess appears to be difficult to formulate in this way.
2. The formulation might obfuscate the underlying computational structure of the problem. For example, in certain general unstructured graph problems, it is probably considerably simpler to act directly on the graph, rather than on a sparse-matrix representation of the adjacency matrix. Similarly, tree algorithms are usually much more

easily understood in terms of trees directly than in terms of their adjacency matrices.

3. Additional mathematical machinery may be required, namely the tools of linear algebra. This tends to be less of a problem for physical scientists, who are familiar with the matrix algebra terminology, but could be a problem in other domains.

On the other hand, there are compensating advantages that in some cases make the formulation useful:

1. Library implementations of linear algebra primitives tend to be highly tuned and very efficient in practice. Rapid execution of such primitives tends to be a motivating factor in design tradeoffs in the architecture, so that the user can normally expect they will execute at good bandwidth. We will demonstrate the utility of this observation when we describe some almost paradoxical, but nonetheless quite fast, parallel programs for the n -body problem (see Chapter 8).
2. A large repertoire of familiar and powerful mathematical techniques is available for the manipulation of particular classes of matrices. These include numerous factorization identities, group representation theory, and the huge body of work on matrix algebra; much of this is probably more familiar to end users than, for example, more general techniques from formal semantics.
3. Conversion to any one notation tends to elicit the common computational themes of disparate applications.

4. Code generation from a factorization of a matrix, as we shall see below, is straightforward, and parallel algorithms targeted to different classes of machines can easily be derived by changing the factorization.

The specific tradeoffs depend on the problem, although we strive, in our selection of case studies, to emphasize that the approach, although not universal, is more general than it might appear.

Section 4.1 briefly reviews the development of linear algebra primitives on parallel machines. Section 4.2 describes the generalization of linear-algebra ideas to semirings, and outlines some brief examples from graph theory. Section 4.3 informally defines the matrix language in which many of our algorithms will be expressed. The fundamental primitive for expressing parallelism will be the *tensor product* \otimes .

4.1 Background

Linear algebra has long been recognized as one of the cornerstones of parallel processing, as the following excerpt, taken from a 1962 article by Slotnick, Borck, and McReynolds, describing the SOLOMON computer, exemplifies [697, p. 97]:

The SOLOMON (Simultaneous Operation Linked Ordinal MODular Network), a parallel network computer, is a new system involving the interconnections and programming, under the supervision of a central control unit, of many identical processing elements (as few or as many as a given problem requires), in an arrangement that can simulate directly the problem being solved.

The parallel network computer shows great promise in aiding progress in certain critically important areas limited by the capabilities of current computing

systems. Many of these technical areas possess the common mathematical denominator of involving calculations with a matrix or mesh of numerical values, or more generally involving operations with sets of variable which permit simultaneous independent operation on each individual variable within the set. This group is typified by the solution of linear systems, the calculation of inverses and eigenvalues of matrices, correlation and autocorrelation, and numerical solution of systems of ordinary and partial differential equations. Such calculations are encountered throughout the entire spectrum of problems in data reduction, communication, character recognition, optimization, guidance and control, orbit calculations, hydrodynamics, heat flow, diffusion, radar data processing, and numerical weather forecasting.

Since then, of course, a tremendous body of research has centered on the parallelization of matrix primitives, and one of the earliest envisioned applications of APL was, in fact, the modeling of microcode-level parallelization [394, 396].

An important characteristic of modern linear-algebra manipulation is its reliance on a standard library of subroutines, the so-called BLAS (Basic Linear Algebra Subprograms) set [243, 520], which has been extended to LINPACK and LAPACK [520, 521].

Linear algebra primitives are particularly important for the effective programming of parallel computers, both because they are the computational bottleneck for large classes of codes, and because of the difficulty of programming in a lower-level. The LAPACK library has been extended to ScaLAPACK, a scalable high-performance computing library intended to provide explicit support for programmers who use matrix primitives in their code [181, 244, 410].

Matrix multiplication is easily parallelizable in the PRAM model in logarithmic time and with optimal work. Even subcubic algorithms of Strassen [725, 726] and Copper-

smith and Winograd [205] may be efficiently parallelized in this model [586,588]. The Coppersmith-Winograd $O(n^{2.376})$ time algorithm is not practical for reasonable matrix sizes, but Strassen's algorithm has been shown to result in real speedups on vector machines [78].

Much of our work depends on fast parallel algorithms and implementations for matrix multiplication [789]. Most efficient matrix multiplications algorithms have their basis in a simple systolic algorithm given by Cannon [159]. Practical algorithms must contend with issues of matrix layout [229,510], pipelining, and constant factors in the minimization of communication overhead [115,304,324]. Implementations of hypercube algorithms on the Connection Machine, on which we have relied, in large measure, for our speedups, must take account of a number of complications, such as local vectorization, memory hierarchies, and the simultaneous utilization of multiple interconnection wires from a processor [354, 372,372,412].

We also make use of BLAS functions such as outer-product routines, based on all-to-all broadcast [146] and transpositions [263]. It is a point worth reemphasizing that even a routine like all-to-all broadcast, which sends a copy of the data in each processor to all the other processors and which is, theoretically speaking, quite trivial, is extremely difficult to implement at peak bandwidth on a parallel architecture, primarily because of constant-factor pipelining and microcode-level-parallelism issues that are almost always elided in theoretical models [414,540]. Thus, in the factorization portion of our main program, we typically factor down to one of the BLAS routines, but it would not be efficient, from a

software-engineering point of view, to recode fundamental BLAS operations at the end-user level.

On the other hand, because the original motivation for BLAS was scientific computation, BLAS routines are typically designed to operate over the field of complex numbers, whereas many of our applications required this functionality over semirings. However, since most of the BLAS complexity for the routines that we have discussed involve data-movement, these are easy to modify for the case when the matrix entries are from a semiring, and this observation, implemented in custom microcode, was part of the reason for the speed of our N -body code (see Chapter 8); in other cases we implicitly embedded our semiring into \mathbb{C} and acted there. The next section discusses these matters in greater detail.

4.2 Graph problems and semirings

There are several methodologies for the study of graph problems which use linear-algebraic concepts, and which demonstrate the utility of linear algebraic methods in unstructured problems. We discuss here two of these: algebraic graph theory and path-algebra theory.

In algebraic graph theory, graph-theoretical questions are explored by analyzing the adjacency matrix of the graph as a complex matrix. For example, bounds on the eigenvalues of the adjacency matrix for a graph can yield a bound on the graph's diameter [185], and bounds on the eigenvalues of the Laplacian of a graph [145, pp.38–43], which is related to its adjacency matrix, can be used in graph partitioning algorithms [352]. There are several ex-

cellent surveys of the field, and the related area of algebraic combinatorics, by Biggs [125], by Godsil [335], and by Brualdi and Ryser [145]. If the spaces \mathfrak{V} and \mathfrak{W} are considered as vector spaces over \mathbb{C} whose dimension is the number of vertices in the graph, then the adjacency matrix \mathbf{M} becomes a complex linear transformation, thereby reducing to the formulation that we give. When the graph has some symmetry given by an automorphism group \mathfrak{G} , then \mathfrak{G} will induce a symmetry of \mathbf{M} . By applying group Fourier transformations to \mathbf{M} , information on graph-theoretical properties of certain graphs has been derived, using serial algorithms [476].

Path-algebra theory, a form of semiring theory, like algebraic graph theory, analyzes graphs via their adjacency matrices. However, whereas algebraic graph theory views the entries in the adjacency matrix as lying in a field, such as \mathbb{C} , in path-algebra theory the matrix entries are considered to be elements of a weaker domain, such as a semiring.

Formally, a *semiring* is defined as an ordered triple $(R, +, *)$ such that R is a set, $(R, +)$ is a commutative monoid, $(R, *)$ is a monoid, and $*$ distributes over $+$.⁷ The most commonly arising example of a semiring is a boolean algebra, although the semiring $(\mathbb{R}, \min, +)$ frequently arises in the context of optimization problems [504].

⁷ There are many slightly different definitions of “semiring.” The introduction to Golan’s survey contains a description of some of the variants. In particular, what we call “semiring” is called a “quasiring” in the popular textbook by Cormen, Leiserson, and Rivest. On the other hand, some authors use the term “quasiring” to mean something else entirely [288, 688] A *path algebra* is a semiring with some additional infinitary closure conditions, whose precise form will not concern us here.

Semirings, in various ways, have been proposed as a way to unify a number of path-finding algorithms [1, 499, 736, 737, 814]. For example, consider the problem finding the smallest-weight path between two nodes in a weighted directed graph of n nodes. It is easy to see that this is reducible to $\log n$ matrix multiplications in the semiring $(\mathbb{R}, \min, +)$. Since matrix-multiplication is inherently parallelizable, this algorithm is easier to parallelize, and has fewer threads of control, than a parallelization of straightforward breadth-first search. Our work in the applications of chess endgames and some of our work in string-matching use matrices whose entries are in a semiring, and so path-algebra ideas are implicit here.

Semirings have been used in a number of parallel graph algorithms; see the surveys [541, 587, 589, 640]. The utility of the path algebra/semiring formulation in parallel algorithms is a natural consequence of our formulation of the generalized operator \mathbf{M} .

4.3 Tensors and programs

We have seen that BLAS routines provide efficient execution of certain primitive matrix operations, and that the algorithms can, by generalizing the domain in which the matrix entries lie, encode wide classes of graph-theoretical algorithms. This section explores in more detail the relationship between a matrix factorization, whether over a semiring or over the complex numbers, and parallel processing. This material is taken directly from the body of work in signal processing pertaining to the relationship between the tensor product and parallelism, and identities useful in the derivation of factorization.

The general theme is quite simple, namely, a factorization of our operator matrix \mathbf{M} in terms of primitive operators like \otimes , $+$, and \cdot , is mapped to a code sequence suitable for execution on a parallel architecture. By modifying the factorization, code suitable for variant architectural parameters can be derived.

4.3.1 Tensor products: Introduction

Let \mathfrak{V}_n be the space of length n vectors with entries in a field \mathfrak{F} . We let $\{\mathbf{e}_i^n\}_{i=1}^n$ be the “standard basis”, that is, \mathbf{e}_i^n is the vector whose i th component is 1 and whose other components are 0.

An element of \mathfrak{V}_n may be thought of as a length n array whose elements are in \mathfrak{F} , or as an $n \times 1$ matrix over \mathfrak{F} [530].

The mn basis elements of $\mathfrak{V}_n \otimes \mathfrak{V}_m$, $\{\mathbf{e}_i^n \otimes \mathbf{e}_j^m\}_{i=0, j=0}^{n-1, m-1}$, are ordered by

$$\mathbf{e}_i^n \otimes \mathbf{e}_j^m \mapsto \mathbf{e}_{mi+j}^{mn}.$$

In this manner an element of $\mathfrak{V}_n \otimes \mathfrak{V}_m$ may be considered to be a vector of length mn with elements drawn from \mathfrak{F} . Let \mathfrak{M}_m^n be the space of $n \times m$ matrices over \mathfrak{F} . In the following, a linear transformation will be identified with its matrix representation in the standard basis.

Let $\mathfrak{M}_n = \mathfrak{M}_n^n$. Let l_n be the $n \times n$ identity transformation of \mathfrak{V}_n .

Write $\text{diag}(\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_{n-1}) \equiv \text{diag}(\mathbf{v})$ for the diagonal matrix in \mathfrak{M}_n whose diagonal elements are taken from the coordinates of \mathbf{v} .

If $\mathbf{A} \in \mathfrak{M}_m^n$ and $\mathbf{B} \in \mathfrak{M}_{m'}^{n'}$, the matrix of the tensor product $\mathbf{A} \otimes \mathbf{B} \in \mathfrak{M}_{mm'}^{nn'}$ is given by

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} \mathbf{A}_{11}\mathbf{B} & \mathbf{A}_{12}\mathbf{B} & \cdots & \mathbf{A}_{1m}\mathbf{B} \\ \mathbf{A}_{21}\mathbf{B} & \mathbf{A}_{22}\mathbf{B} & \cdots & \mathbf{A}_{2m}\mathbf{B} \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{A}_{n1}\mathbf{B} & \mathbf{A}_{n2}\mathbf{B} & \cdots & \mathbf{A}_{nm}\mathbf{B} \end{pmatrix} \quad (4.1)$$

The importance of the tensor-product to our work in parallel processing inheres in the following identity, for $\mathbf{B} \in \mathfrak{M}_m$: [408]

$$(\mathbf{I}_n \otimes \mathbf{B}) \begin{pmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{nm-1} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \cdot \begin{pmatrix} \mathbf{v}_0 \\ \vdots \\ \mathbf{v}_{m-1} \end{pmatrix} \\ \mathbf{B} \cdot \begin{pmatrix} \mathbf{v}_m \\ \vdots \\ \mathbf{v}_{2m-1} \end{pmatrix} \\ \vdots \\ \mathbf{B} \cdot \begin{pmatrix} \mathbf{v}_{(n-1)m} \\ \vdots \\ \mathbf{v}_{nm-1} \end{pmatrix} \end{pmatrix} \quad (4.2)$$

Suppose $n = ml$. The n -point stride l permutation matrix \mathbf{P}_l^n is the $n \times n$ matrix defined by

$$\mathbf{P}_l^n (\mathbf{v} \otimes \mathbf{w}) = \mathbf{w} \otimes \mathbf{v},$$

where $\mathbf{v} \in \mathfrak{V}_m$ and $\mathbf{w} \in \mathfrak{V}_l$. The effect of \mathbf{P}_l^n on a vector is to stride through the vector, taking m steps of size l . For example, taking $m = 3$, $l = 2$, and $n = 6$, we have:

$$\mathbf{P}_2^6 \begin{pmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \\ \mathbf{v}_5 \end{pmatrix} = \begin{pmatrix} \mathbf{v}_0 \\ \mathbf{v}_2 \\ \mathbf{v}_4 \\ \mathbf{v}_1 \\ \mathbf{v}_3 \\ \mathbf{v}_5 \end{pmatrix} \quad (4.3)$$

Stride permutations are important due to the following *Commutation Theorem* [758]:

Theorem 1

$$P_l^n(A \otimes B)P_m^n = B \otimes A$$

where $A \in \mathfrak{M}_m$, $B \in \mathfrak{M}_l$, and $n = ml$.

This theorem, which is easy to prove even when the entries are from a semiring, allows the order of evaluation in a tensor product to be varied. We shall see in the next subsection that some evaluation orders naturally correspond to vectorization, and some to parallelizations; the Commutation Theorem will be the method by which one type of execution is traded off for another.

4.3.2 Code generation: Conversion from factorization to code

This subsection describes the relationship between the matrix representation of a formula and the denoted machine code. Because many of the algorithms to be presented will be presented in the tensorial manner, with the code-generation phase only represented implicitly, this subsection is fundamental to this dissertation.

The matrix notation we use is nothing more than an informal notation for describing algorithms. It differs from standard notations primarily in its explicit denotation of data distribution, communication, and operation scheduling. Whereas most high-level languages, and even special-purpose parallel languages, leave the distribution of data over the processors and the scheduling of operations within processors to the discretion of the compiler, the notation we use, at least potentially, encodes all such scheduling. This has both advan-

tages and disadvantages: although it gives the programmer a finer level of control, which can be important for time-critical applications, it requires some conscious decision-making over data-distribution that is unnecessary in some other languages. On the other hand, the functional nature of the notation does make it potentially amenable to compiler reordering. The most serious disadvantage is its narrowness of application. Originally developed for signal processing codes, this work demonstrates its wider application, but there are many applications which would not easily fall under its rubric.

The target architecture of the language is a machine comprising m parallel processors, each with shared memory. However, it is easy to see that the results go through also, with an extra communication step or two, on local-memory machines. Each processor may also have vector capabilities, so that computations within the processors should be vectorized. We do not assume restrictions on the vector length capability of the processors.

User data is always stored conceptually in the form of a vector

$$\begin{pmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{n-1} \end{pmatrix}.$$

Assuming that m divides n , elements $\mathbf{v}_0, \dots, \mathbf{v}_{\frac{n}{m}-1}$ are stored in processor 0, elements $\mathbf{v}_{\frac{n}{m}}, \dots, \mathbf{v}_{\frac{2n}{m}-1}$ are stored in processor 1, and so on. Matrices are stored in column-major order. It is assumed that certain general classes of specific matrices are already implemented on the architecture, in particular, the stride permutations and any specific permutations corresponding to the interconnection network.

Let $\mathbf{B} \in \mathfrak{M}_l$ and let $\mathbf{code}(\mathbf{B})$ be any sequence of machine instructions that computes the

result of left-multiplication by \mathbf{B} . That is, $\mathbf{code}(\mathbf{B})$ is a program that takes as input an array \mathbf{v} of l elements of \mathfrak{F} , and returns as output the array $\mathbf{B} \cdot \mathbf{v}$ of l elements of \mathfrak{F} , where vectors are identified with their coordinates in the standard basis.

Given $\mathbf{code}(\mathbf{B})$ and $\mathbf{code}(\mathbf{B}')$ for two matrices \mathbf{B} and \mathbf{B}' , it is easy to compute some $\mathbf{code}(\mathbf{B} + \mathbf{B}')$. Simply let $\mathbf{code}(\mathbf{B} + \mathbf{B}')$ be the program that, given its input array \mathbf{v} , first runs as a subroutine $\mathbf{code}(\mathbf{B})$ on \mathbf{v} (saving the result), then runs $\mathbf{code}(\mathbf{B}')$ on \mathbf{v} , and then returns the coordinate-wise sum of the arrays that are returned by these two subroutine calls.

Similarly, given $\mathbf{code}(\mathbf{M})$ and $\mathbf{code}(\mathbf{M}')$, it is easy to find $\mathbf{code}(\mathbf{M} \cdot \mathbf{M}')$, assuming the dimensions of \mathbf{M} and \mathbf{M}' are compatible: run $\mathbf{code}(\mathbf{M})$ on the result of running $\mathbf{code}(\mathbf{M}')$ on the argument \mathbf{v} .

Of course, $\mathbf{code}(I_l)$ is the code that returns its argument, an l -vector.

Consider a parallel processor with m processors, p_1, \dots, p_m , each with some local memory. We make the convention that a length ml array will be stored with its first l elements in processor p_1 , its second l elements in processor p_2 , and so on.

Given this convention, one can interpret $\mathbf{code}(I_m \otimes \mathbf{B})$ as code that runs on this m -processor architecture. To construct $\mathbf{code}(I_m \otimes \mathbf{B})$, load $\mathbf{code}(\mathbf{B})$ in each p_i . When called on a length ml array \mathbf{v} , p_i runs $\mathbf{code}(\mathbf{B})$ on the l elements of \mathbf{v} that are stored in its local memory, and outputs the result to its local memory. Equation 4.2 shows that this will compute the tensor product. Similar rules can be derived when the number of processors is different

from m .

The code corresponding to $\mathbf{A} \otimes \mathbf{l}_l$, for $\mathbf{A} \in \mathfrak{M}_m$, is a bit more subtle. The interpretation of $\mathbf{code}(\mathbf{A} \otimes \mathbf{l}_l)$ is as the code corresponding to \mathbf{A} , except that it operates on l -vectors rather than on scalars. This code can be constructed (loosely speaking) from $\mathbf{code}(\mathbf{A})$ by interpreting the length ml argument array \mathbf{v} as being an element of the m -module over the ring \mathfrak{F}^l . This corresponds closely to hardware primitives on certain vector architectures.

The relation

$$\mathbf{A} \otimes \mathbf{B} = (\mathbf{A} \otimes \mathbf{l}_l)(\mathbf{l}_m \otimes \mathbf{B}) \quad (4.4)$$

can be used to compute general tensor products.

By combining a fixed set of transformations reflecting the hardware primitives of the underlying architecture with combining rules like $+$, \cdot and \otimes , and some simple tensor product identities, concise expressions that can be translated into efficient code for certain classes of functions can be defined [345].

4.3.3 Example: Matrix multiplication by a tensor product

In order to illustrate the process of translation from tensor-product formulas into code we will describe a simple example taken from [345].

As above, let $\mathbf{A} \in \mathfrak{M}_m$ and $\mathbf{B} \in \mathfrak{M}_l$. Let $\mathbf{M} = \mathbf{A} \otimes \mathbf{B} \in \mathfrak{M}_n$, where $n = ml$. This subsection describes the development of fast algorithms to compute $\mathbf{w} = \mathbf{M}\mathbf{v}$, where $\mathbf{v}, \mathbf{w} \in \mathfrak{M}_n$, on various architectures. Except for the fact that the entries in the matrices considered here

lie in arbitrary semirings, the exposition in this section closely follows the signal-processing paper of Granata, Conner, and Tolimieri [345, pp.44–47].

We consider several target architectures. For each target architecture, we give a factorization and generated code.

The first target architecture to be considered is a *serial* machine. The associated factorization is

$$\mathbf{M} = \mathbf{A} \otimes \mathbf{B} \quad (4.5)$$

$$= (\mathbf{A} \mathbf{I}_m) \otimes (\mathbf{I}_l \mathbf{B}) \quad (4.6)$$

$$= (\mathbf{A} \otimes \mathbf{I}_l) (\mathbf{I}_m \otimes \mathbf{B}) \quad (4.7)$$

$$= (\mathbf{P}_l^n (\mathbf{I}_m \otimes \mathbf{A}) \mathbf{P}_m^n) (\mathbf{I}_l \otimes \mathbf{B}) \quad (4.8)$$

The term $\mathbf{I}_m \otimes \mathbf{B}$, as explained earlier, represents a loop of the function \mathbf{B} over l contiguous length m segments of its length n argument.

The term

$$\mathbf{P}_l^n (\mathbf{I}_m \otimes \mathbf{A}) \mathbf{P}_m^n$$

represents a similar loop for \mathbf{A} , except that the argument is reordered before being passed to \mathbf{A} . Of course, on a serial machine this reordering can always be folded into the computation for \mathbf{A} , but for clarity we write it out in the serial code for $\mathbf{w} = \mathbf{M}\mathbf{v}$, below:

```

for  $i = 0$  to  $l - 1$  /* For each segment* /
     $temp(im : im + l - 1) = \mathbf{B} \cdot \mathbf{v}(im : im + l - 1)$  /* Call  $\mathbf{B}$  on that segment */

```

```

temp = P_m^n · temp /*Compute the stride permutation*/
for i = 0 to m - 1
    w(il : il + m - 1) = A · temp(il : il + m - 1)
w = P_l^n · w

```

Note that this method requires time proportional to $m^2 + l^2$, which is normally much smaller than the n^2 brute-force method; more to the point, the method uses existing, and possibly highly optimized, software for **A** and for **B** in order to generate the code for **M**.

The next target architecture is a vector machine. The vector machine is presumed, as primitives, to be able to perform arbitrary operations on vectors, and we presume a primitive load-stride operation.

The expression $\mathbf{B} \otimes l_i$ corresponds to the routine named by **B** performed on vectors of length s instead of scalars, assuming that the code for computing **B** uses only *linear* operations, namely multiplication by a constant and vector addition. Because the focus of this thesis is on parallelism and not vectorization, we will skip the code for this section.

The third architecture that we consider is that of a 16-processor SIMD machine, and assume that $m = l = 64$, so that $\mathbf{M} \in \mathfrak{M}_{4096}$.

The associated factorization is

$$\mathbf{M} = \mathbf{P}_{64}^{4096} (\mathbf{l}_{64} \otimes \mathbf{A}) \mathbf{P}_{64}^{4096} (\mathbf{l}_{50} \otimes \mathbf{B}) \quad (4.9)$$

$$= \left(\mathbf{P}_{64}^{4096} (\mathbf{l}_{16} \otimes \mathbf{l}_4 \otimes \mathbf{A}) \right) \left(\mathbf{P}_{64}^{4096} (\mathbf{l}_{16} \otimes \mathbf{l}_4 \otimes \mathbf{B}) \right) \quad (4.10)$$

The terms $l_4 \otimes \mathbf{A}$ and $l_4 \otimes \mathbf{B}$ correspond to looping calls that are executed within each of the 16 processors. The l_{16} terms indicate that the identical code is executed in each processor. The stride terms can be implemented via a reindexing on a shared-memory machine, and via permutation on a local memory machine. It is clear that when each processor is itself a vector processor, then the $l_4 \otimes \mathbf{A}$ term can be rewritten using the vectorization methods described above.

Another simple class of examples is provided by classical fast Fourier transform algorithms, which will be discussed in more detail in section 7.1.

Chapter 5

Exploiting symmetry: Mathematical framework

This chapter describes in more detail the programming methodology that will be adopted in Part II of this thesis.

The problems we will solve will have the following two parts:

1. A function $M: \mathfrak{V} \rightarrow \mathfrak{W}$. Here \mathfrak{V} and \mathfrak{W} are simply some domains of objects.
2. A group \mathfrak{G} acting on \mathfrak{V} and on \mathfrak{W} . $\mathfrak{V}, \mathfrak{W}$ are spaces in which the data items lie.

The group will express the symmetry of M in the sense that we are guaranteed that,

$$\forall \mathfrak{g} \in \mathfrak{G}, \mathfrak{g} \circ M = M \circ \mathfrak{g}.$$

The methodology used to solve a problem presented in this manner may be divided into two main techniques:

First, we *set up* the problem in a generalized linear-algebraic formulation. We suppose that

the data items in \mathfrak{X} are uniquely denoted by some set of n features, which take values in some algebraic structure \mathfrak{F} . Similarly, we suppose that data items in \mathfrak{Y} are denoted by m distinct features taking values in \mathfrak{F} . The operator \mathbf{M} is considered to be an $m \times n$ matrix over some algebraic structure \mathfrak{F} . Of course, this formulation is often not possible, but it may be possible for some portion of the computational problem to be solved. The matrix \mathbf{M} normally has a special structure that reflects the structure of the problem.

Second we *exploit the symmetry* in \mathbf{M} . Because \mathbf{M} arose from a problem with a symmetry expressed by a symmetry group \mathfrak{G} , if the \mathfrak{G} action is linear in \mathfrak{F} , then we get a representation $\rho_{\mathfrak{X}}$ and $\rho_{\mathfrak{Y}}$ of \mathfrak{G} on \mathfrak{X} and \mathfrak{Y} . Then we have the formulas

$$(\forall \mathfrak{g} \in \mathfrak{G}) \rho_{\mathfrak{Y}}(\mathfrak{g}) \circ \mathbf{M} = \mathbf{M} \circ \rho_{\mathfrak{X}}(\mathfrak{g})$$

We mainly consider the case where $\rho_{\mathfrak{X}}$ and $\rho_{\mathfrak{Y}}$ are *permutation representations*; that is, they only permute the features to be considered, and their matrices are permutations matrices.

We use three main tools to exploit the symmetry. Which of these tools to use, and in what contexts, depends on details of the problem. These tools are:

Factorization The matrix \mathbf{M} will be factored using the operators of matrix multiplication, direct sum, and Kronecker product. Each such factorization induces a parallel algorithm on a particular architecture; by modifying the factorization, we can derive algorithms for variant architectures.

Orbit decomposition By choosing only a single element from each *orbit* of the \mathfrak{G} action, we can reduce the computational requirements of computing $\mathbf{M}\mathbf{v}$ for $\mathbf{v} \in \mathfrak{X}$. This

decomposition induces a new operator, and in some cases induces a parallel algorithm whose communication pattern is characterized by a Cayley graph of \mathfrak{G} .

Group Fourier transform For certain groups \mathfrak{G} a fast *group Fourier transform* exists.

This is a generalization of the ordinary discrete Fourier transform whose application block-diagonalizes the operator \mathbf{M} ; by computing the \mathfrak{G} -Fourier transform of \mathbf{M} and the operand \mathbf{v} , we will be able to compute the product $\mathbf{M}\mathbf{v}$ faster than we would otherwise have been able to do.

Although our machinery and problem statement might seem overly limiting, we hope to illustrate, by means of several case studies, that symmetry considerations are implicit in a broad range of problems.

Briefly, though, here is how the symmetry groups \mathfrak{G} arise, for various applications.

In the chess endgame code, the operator \mathbf{M} acts on *sets* of chess positions. These sets correspond to binary vectors in a natural way. Now, consider a chess position P having no pawns. Suppose we list all the positions that White could move to from the position P , and then we apply some symmetry transformation of the chessboard—rotation or reflection—to each of these positions, and call the resulting set S . Now, *first* apply the same transformation to P , and *then*, find each position to which White could move, from the transformed P , and call the resulting set S' . It is a consequence of the rules of chess that (ignoring castling) when there are no pawns, $S = S'$. Thus, our resulting matrix \mathbf{M} will commute with the symmetry group of the square, and \mathbf{M} is \mathfrak{D}_4 -invariant.

In the case of n -body simulation, that is, the problem of simulating the motions of n -particles moving through space, the symmetry group is $\mathfrak{G} = \mathfrak{C}_n$. It arises because if the particles are cyclically shifted, then the forces on them are cyclically shifted as well. Thus, the matrix representing the sum of forces on each particle is \mathfrak{C}_n invariant.

In the case of string-matching, the symmetry arises as follows. If the text is rotated, then the sets of positions at which a given pattern string occurs in the text string will be rotated in the same way, assuming we allow the pattern to wraparound appropriately. Thus, the match matrix \mathbf{M} in the case of classical string-matching will be \mathfrak{C}_n -invariant.

The next part will discuss these and other applications in greater detail, but it should be clear that our formulation, although undoubtedly highly limited in scope, is not as limited as it might appear at first.

Part II

Applications

Chapter 6

Orbit decomposition and its application to the analysis of chess endgames

The simplest way to exploit the symmetry of a problem has probably been used, in one sense, by almost every programmer, and consists simply in avoiding the recomputation of expressions that have already been computed.

Specifically, we are given some set of data that is symmetric, and we wish to compute some function of the data. If the function “respects” the symmetry of the data, informally speaking, then we only need to store a part of the data, namely, one orbit-representative from each orbit of the action of the symmetry group on the data. We call this “orbit decomposition.”

Unfortunately this symmetry-exploitation paradigm can result in less regular data-access patterns compared to the algorithms that do not exploit symmetry. While this constitutes

only a minor inconvenience on scalar machines, such irregularity can severely undermine the performance of parallel or vector codes.

This chapter describes the methods for efficiently implementing an orbit decomposition problem on a parallel architecture. The domain of application comes from chess endgames—the problem is to determine whether a particular chess endgame with a small number of pieces can be won for White—and the salient symmetry characteristic was the invariance under a non-commutative group. The resulting algorithm reduces to an embedding of the Cayley graph for the symmetry group of the problem into the parallel architecture.

Section 6.1 provides an overview of some of the background and motivation for the problem. Although the analysis of chess endgames has a long history, much of the historical information is unfamiliar to many people, and previous researchers had overlooked or underestimated the significance of the work of a number of key contributors to the early stages of computer endgame analysis. Subsection 6.1.1 contrasts the chess endgame problem with other classic search problems in artificial intelligence. Subsection 6.1.2 provides a brief overview of some of the pre-computer work in these endgames. Subsection 6.1.3 focuses particularly on two endgame analysts from the early 1900s whose work had been ignored or underestimated by previous researchers: Friedrich Amelung and Theodor Molien. Amelung was the first to carry out detailed investigations of one of the six-piece endgames we solved, and Molien was the first to provide an (approximate) statistical characterization of a pawnless endgame. Interestingly, comparatively recent research by Hawkins and Kanunov has shown that Molien was also extremely influential in the development of group representa-

tion theory. Subsection 6.1.4 surveys previous computer analysis of the type we perform. Although most of this material was known, we feel that previous researchers in the area may have given insufficient credit to Richard Bellman, whose papers from as early as 1961 sketched a retrograde analysis algorithm and predicted its applicability to the complete analysis of checkers.

Section 6.2 outlines the relationship between tensor products and chess endgames, and introduces the basic notation. Section 6.3 describes the main endgame algorithm, and subsection 6.3.2 specifically discusses symmetry exploitation.

Implementation issues are considered in section 6.4 and section 6.5 presents some of the results that were discovered by the program.

Section 6.6 concludes with ideas for future work.

6.1 Motivation and background

6.1.1 Search

Search is one of the oldest and most fundamental problems in artificial intelligence. In its most general form, we are given some object and a set of operations, each of which changes the state of the object, and we want to find a sequence of operations to apply to the object, satisfying certain criteria (such as having least cost) that takes the object into some goal state. For example, an automated theorem prover might prove a theorem by

continually deriving new results until the theorem itself is proven; an airline computer may wish to plan its schedule of flights by continually adding new flights until some coverage and cost requirements are met; a vision system might try a number of transformations of some digital representation of an object until a match against a fixed template is attained; a puzzle solver might try to find the shortest sequence of moves that solve a puzzle, such as the 15-puzzle.

In their full generality, most of these problems that arise in artificial intelligence are NP-hard and are, thus, unlikely to be solved exactly in sub-exponential time. In order to make progress on these problems, artificial intelligence practitioners deploy an array of simplifications and tricks.

One approach is approximation: look for suboptimal solutions, instead of trying to find the *best* solution.

Another approach is to use heuristics—rules-of-thumb—to guide the search into promising areas. The most popular such algorithm is A* and its variants [231, 457, 596]. There are several parallelizations of A*-like algorithms [213, 231, 272]. This approach can work well if good heuristics are available, but, of course, that is not always the case.

One might also try techniques such as simulated annealing, genetic hillclimbing, or neural-networks to try and guide the search.

The full-chess problem is simply to program a computer to beat humans at chess. It dates back at least to Babbage, as is discussed later, although modern approaches follow

a combination of heuristics and alpha-beta pruning. The heuristics assign to each node in the game-tree a value approximating its goodness. The alpha-beta pruning techniques eliminate approximately a factor of square-root of the number of nodes in the tree by improving on the brute-force minimax algorithm. Production chess programs deploy an array of auxiliary sophisticated pruning techniques, such as singular extensions, null-move cutoffs, killer-move, history, and many others [595,596].⁸ Alpha-beta search is simply a refinement of the classic brute-force strategy of Shannon [679] which considers, from any given position, each move; then each of the opponents replies; then each reply to each such reply, and so on. The algorithm dynamically prunes nodes that can be proven not to affect the value of the original position. Even though it is called “brute-force,” the algorithm only examines roughly the square-root of the number of all possibilities up to a certain depth.

The upshot of all this is that, given the root node, it can be complicated to determine which nodes in the game tree need to be searched, and, therefore, it is difficult to avoid searching redundant nodes in a parallel program [80]. There are a number of theoretical models of game-tree search in which speedup has been predicted to occur (e.g. [16,23,284,292]), but the many difficult-to-model vagaries of the distribution of chess-tree node values and their interaction with complex tree-pruning and move-evaluation heuristics have obviated much of this work [158,533–535]. Nevertheless, there are successful coarse-grained parallel chess-searchers, such as the ParaPhoenix work of Schaeffer [663] and the Cray codes of Hyatt et al. and of Warnock and Wendroff [382,388,389]. There are only a few massively

⁸ The origins of alpha-beta are somewhat controversial. The reader is referred to Knuth and Moore’s article (1975) for historical background [449].

parallel full-chess players. The first was the NCUBE WAYCOOL of Felten and Otto [283]; currently the CM-5 programs of Berliner, Leiserson, Kuszmaul, McConnell, Kaufmann et al. [473] and the Transputer's Zugzwang of Feldmann, Monien, Mysliwicz, and Vornberger are the strongest [277–281], although a custom VLSI massively parallel machine is under development and is predicted to be very strong [380–382]. On the other hand, there is considerable implicit parallelism in the VLSI architectures of Thompson and, on a different level, of Berliner and Ebeling [113, 198, 258].

Despite its superficial similarity, the work reported here takes a sharply different approach to parallelism than do parallel tree searchers. First, we generate moves backward, starting from the mating positions; second, we do not use heuristics; third, we do not use pruning. Generating operators backwards has also been used in the context of symbolic protocol evaluation [377], and is an important feature of bi-directional search [604].

Finally, we say a few words about the beautiful work on exploiting symmetry in forward-search problems using techniques of isomorph-rejection [153]. This work uses backtracking combined with symmetry considerations to avoid exploring a state that can be proven by symmetry to have been already eliminated [479, 482, 483]. It achieved its most spectacular success in Lam's proof of the nonexistence of a finite projective plane of order 10 [481]. Some attention has been given as well to parallelization and vectorization of this style of backtracking, which impinges upon computational group theory [741, 817]. Parallel algorithms for bi-directional search problems in the presence of symmetry were discussed in section 3.4 [142].

It is noteworthy that many games, such as chess, checkers, Hex, and others, have been proven to be PSPACE or even EXP-TIME complete [305,619,620,627,724], using the machinery of alternating Turing machines, a powerful game-like generalization of non-deterministic Turing machines [173,720]. This suggests that an algorithm that scales to $n \times n$ boards cannot improve substantially on the one we give of analyzing all nodes in the state-space [763].

6.1.2 Human analysis

Endgame analysis appears to date from at least the 9th century, with al-‘Adlī’s analysis of positions from ♔♚♜♞⁹ [6, plate 105] and ♔♚♞♜♚♞ [6, plate 112]. However, the rules were slightly different in those days, as stalemate was not necessarily considered a draw. The oldest extant collection of compositions, including endgames, is the Alfonso manuscript, ca. 1250, which seems to indicate some interest during that time in endgame study [600, pp.111–112].

Modern chess is generally considered to have begun roughly with the publication, probably in 1497, of Luis Ramirez de Lucena’s *Repetición de amores y arte de ajedrez* [226].¹⁰ Ruy Lopez de Sigura’s 1561 book briefly discusses endgame theory, but its main impact on this

⁹ In listing the pieces of an endgame, the order will be White King, other White pieces, Black King, other Black pieces. Thus, ♔♚♜♞ is the same as ♔♚♞♜, and comprises the endgame of White King and White Rook against Black King and Black Knight

¹⁰ Ironically, this work does not contain the famous “Lucena position” from ♔♚♜♞♚♞, which seems to have been first published by Alessandro Salvio in 1634, who attributed it to Scipione Genovino.

work would be the introduction of the controversial 50-move rule, under which a game that contains 50 consecutive moves for each side without the move of a pawn or a capture could be declared drawn [227, pp.55–56] [646].

Pietro Carrera's 1617 *Il gioco de gli scacchi* discussed a number of fundamental endgames such as ♔♚♔♙♙, and certain 6-piece endgames such as ♔♚♚♔♚♙ and ♔♚♚♔♚♙ [164, Book 3, p. 176–178]. A number of other authors of the time, such as Philip Stamma (1737), François-André D. Philidor (1749), and Gioacchino Greco (1624,) began developing the modern theory of endgames [347, 602, 702]. Giovanni Lolli's monumental *Osservazioni teorico-pratiche sopra il giuoco degli scacchi* (1763) would be one of the most significant advances in endgame theory for the next 90 years [518, 643]. Lolli analyzed the endgame ♔♚♔♙♙, and he agreed with the earlier conclusion of Salvio (1634) that the endgame was a general draw for White [660]. This assessment would stand substantially unchanged until Kenneth Thompson's computer analysis demonstrated the surprising 71 move win [753]. Notwithstanding this error, Lolli did manage to discover the unique ♔♚♔♙♙ position in which White to play draws but Black to play loses [518, pp.431–432].

Bernhard Horwitz and Josef Kling's 1851 *Chess Studies* contained a number of influential endgame studies, although their analysis of ♔♙♙♔♙ was questioned by A. John Roycroft (1972) [445, pp.62–66] [643, p. 207]. The Horwitz and Kling assessment was definitively shown to be incorrect by the independent 1983 computer analyses of Thompson and Ofer Comay [645, 751].

Alfred Crosskill (1864) [212] gave an analysis of ♔♚♙♔♚ in which he claimed a win in

more than 50 moves was required; this was confirmed by computer analysis of Thompson. The Crosskill analysis was the culmination of a tradition of analysis of ♔♚♙♔♚ beginning at least from the time of Philidor [602, pp.165–169].

A generation later, Henri Rinck and Aleksei Troitzky were two of the most influential endgame composers of their time. Troitzky is well-known for his analysis of ♔♙♙♙♔♙—he demonstrated that > 50 move wins were at times required [766]. Rinck was a specialist in pawnless endgames, composing more than 500 such studies [625, 626], including some with 6 pieces. Troitzky summarized previous work in the area of ♔♙♙♙♔♙, beginning with a problem in ♔♙♙♙♔♙ from the 13th-century Latin manuscript *Bonus Socius* [228], and reserved particular praise for the systematic analysis of this endgame in an 18th-century manuscript by Chapais [175]. (An early version of the program reported in this chapter resulted in the first published solution for the entire endgame [707].)

The 20th century saw the formal codification of endgame theory by scholars such as Johann Berger (1890) [106], André Chéron (1960) [178], Machgielis [Max] Euwe (1940) [269], Reuben Fine (1941) [291], Yuri L. Averbakh (1982) [68], and many others. Some work focusing particularly on pawnless 6-piece endings has also appeared, for example, [107, 456, 642].

Currently the Informator *Encyclopedia of Chess Endings* series [539], which now uses some of Thompson’s computer analysis, is a standard reference. John Nunn has written several books based on that work [581, 583].

Additional historical information can be found in the references [341, 378, 572, 643].

6.1.3 Friedrich Amelung and Theodor Molien: A historical note

This subsection discusses the work of Friedrich Amelung and Theodor Molien as it pertains to pawnless chess endgame analysis.

Friedrich Ludwig Amelung (March 11, 1842–March 9, 1909) was a Latvian chess player and author who edited the chess column of the Riga newspaper *Düna-Zeitung*. He studied philosophy and chemistry at the University of Dorpat from 1862 to 1879, and later became a private teacher and director of a mirror factory [507, p.11] [45,391]. He published a number of endgame studies and analyses of endgames, and began a systematic study of pawnless endgames. For example, he explored the endgame ♔♚♚♚♚♚ in detail [36,37]; this endgame was shown to have unexpected depth, requiring up to 46 moves to win, in later work by the author [707]. He also published an article on ♔♚♚♚♚♚ and ♔♚♚♚♚♚ [28], which were not exhaustively analyzed until the 1980s [707, 753].

However, his main interest to our work actually inheres in two major projects: an analysis of the 4-piece endgame ♔♚♚♚, which appeared in 1900 [30–35], and his studies of certain pawnless 6-piece endgames [38–44].

Amelung’s 1900 analysis of ♔♚♚♚ was significant because it contained the first histogram known to the author of a pawnless endgame or, for that matter, of any endgame [34, pp.265–266]. This table listed the approximate number of positions in ♔♚♚♚ from which White could win and draw in 2–5 moves, 5–10 moves, 10–20 moves, and 20–30 moves. Such tables have been a mainstay of computer-age endgame analysis, of course. The existence of this

early analysis does not appear to have been known to contemporary workers, although it appeared in a widely read and influential publication, *Deutsche Schachzeitung*.

Even more intriguing, however, is Amelung’s comment that an even earlier, exact numerical analysis, containing the number of win-in- k moves for each k of a four-piece chess endgame was known, and was due to “Dr. Th. Mollien, der Mathematiker von Fach ist”; that is, to the professor “Th. Mollien.”

Theodor Molien (September 10, 1861–December 25, 1941)¹¹ was born in Riga.¹² His father, Eduard, was a philologist and teacher, and Theodor eventually became fluent in a number of languages, including Hebrew, Greek, Latin, French, Italian, Spanish, Portuguese, English, Dutch, Swedish, and Norwegian, as well as German and Russian, of course. “If you read a hundred novels in a language,” Molien liked to say, “you will know that language. [422, p.9]”¹³ He studied celestial mechanics at Dorpat University (1880–1883) and also took courses from Felix Klein in Leipzig (1883–1885). His doctoral dissertation, which was published in *Mathematische Annalen* [556, 557] proved a number of the fundamental

¹¹ There are a number of variant English spellings of Molien’s name: Molin [86], Mollin [35, p.5], Mollien [34, p.265], and Molien [27, 29]. His biography gives his name as Федор Эдуардович Молин (Fedor Eduardovich Molin) [422]. We will refer to him as Theodor Molien in conformity with his publications [556–559, 562].

¹² Molien’s biographical information has been taken from Kanunov [422], which was translated for this project by Boris Statnikov.

¹³ «Прочитайте сто романов на каком-либо языке,—любил говорить он позднее,—и Вы будете знать этот язык.»

structure theorems of group representation theory, including the decomposability of group algebras into direct sums of matrix algebras, which is crucial to our own work in Chapter 7.

Molien's early papers on group representation theory [556–559, 562], despite their importance, were obscure and difficult to understand. Indeed, his papers anticipated Frobenius' classic paper on the determinant of a group-circulant matrix [308], a fact which Frobenius readily admitted [359], although he had tremendous difficulty understanding Molien's work (letter to Alfred Knezer, May 6, 1898). In a letter to Dedekind, February 24, 1898, Frobenius wrote:

You will have noticed that a young mathematician, Theodor Molien in Dorpat, has independently of me considered the group determinant. He has published, in volume 41 of the *Mathematische Annalen* a very beautiful but difficult to read work "On systems of higher complex numbers [557]," in which he investigated non-commutative multiplication and obtained important general results of which the properties of the group determinant are special cases.¹⁴

Despite these results, and despite Frobenius' support, Molien was rejected from a number of Russian academic positions, partly because of the Czarist politics of the time (according to Kanunov) and, at least in one case, because the committee considered his work too theoretical and without practical applications [422, pp.35–36]. After studying medieval

¹⁴ "Sie werden bemerkt haben, daß sich ein jungerer Mathematiker Theodor Molien in Dorpat unabhängig von mir mit der Gruppensdeterminante beschäftigt hat. Er hat im 41. Bande der Mathematischen Annalen eine sehr schöne, aber schwer zu lesende Arbeit 'Ueber Systeme höherer complexer Zahlen' veröffentlicht, worin er die nicht commutative Multiplication untersucht hat und wichtige allgemeine Resultate erhalten hat, von denen die Eigenschaften der Gruppensdeterminant specielle Fälle sind. [Excerpt from a transcription by Walter Kaufmann Bühler of a letter from Frobenius to Dedekind dated February 24, 1898. A copy of this transcription was kindly provided by Thomas Hawkins, Department of Mathematics, Boston University, and is excerpted here with the permission of Springer-Verlag.]"

mathematical manuscripts at the Vatican Library in 1899 [422, p.35], he accepted a post at the Tomsk Technological Institute in Siberia where he was cut off from the mathematical mainstream and became embroiled in obscure administrative struggles (he was, in fact, briefly fired). His remaining mathematical work had little influence and he spent most of his time teaching.

Thus, Molien's work was unknown or underestimated in the West for a long while, for example, Wussing's classic 1969 text barely mentions him [809]. With the publication of Thomas Hawkins series of articles on the history of group representation theory [357–359], the significance of Molien's contributions became better-known, and van der Waerden's 1985 history of algebra gives Molien due credit [792, pp.206–209,237–238].

Although it is not mentioned in Kanunov's biography, before Molien moved to Tomsk, he was one of the strongest players in Dorpat and was particularly known for his blindfold play (Ken Whyld, personal communication, 1995). He was president of the Dorpat chess club, and several of his games were published in a Latvian chess journal, *Baltische Schachblätter*, edited, for a time, by Amelung [27] [96, p.8]; one of his games (which he lost) won a “best-game” prize in the main tournament of the Jurjewer chess club in 1894 [563].

Molien's numerical studies of $\text{♔} \text{♚} \text{♗} \text{♘}$ are alluded to several times in the chess journals of the time (about 1900) [29,560] [35, p.5] [34, p.265]. In 1898 he published four chess studies [561] based on his research into the endgame $\text{♔} \text{♚} \text{♗} \text{♘}$ [35, p.5]. However, we have not been able to locate a publication of his complete results, despite the historical significance of such a document.

In any case, it seems to me to be an interesting coincidence that within a span of a few years Molien performed groundbreaking work in two apparently unrelated areas: group representation theory and quantitative chess endgame analysis, although his work in both areas was mostly ignored for a long time. Furthermore, major parts of my thesis continue along two paths first blazed by Molien: this chapter reports on work that continued Molien’s quantitative analysis of pawnless chess endgames; Chapter 7 continues Molien’s work on the decomposition of finite group algebras.

We now continue with our discussion of chess endgame history proper, and in particular, Amelung’s work on pawnless endgames, of which his work on $\text{♔} \text{♚} \text{♛} \text{♜} \text{♝} \text{♞}$ deserves special mention. Partly in response to the first edition of Johann Berger’s influential 1890 manual of endings [106, 167–169], in 1902 Amelung published a three-part series in *Deutsche Schachzeitung*, perhaps the premier chess journal of its time, analyzing the endings of King, Rook and minor piece (♞ or ♝) against King and two minor pieces [38–40], and represented a continuation of Amelung’s earlier work with Molien on the endgame $\text{♔} \text{♚} \text{♜} \text{♝}$ [34]. Amelung indicated that the endgame $\text{♔} \text{♚} \text{♛} \text{♜} \text{♝} \text{♞}$ was particularly interesting, and in 1908 he published a short article on the topic in *Für Haus und Familie*, a biweekly supplement to the Riga newspaper *Düna-Zeitung*, of which he was the chess editor [41]. Amelung’s interest in this endgame was so great that he held a contest in *Düna-Zeitung* for the best solution to a particular example of this endgame [43]. A solution was published the next year [44], but Amelung died that year and was unable to continue or popularize his research. Consequently, succeeding commentators dismissed many of his more extreme claims, and his work

seemed to pass into oblivion. It is discussed in the 1922 edition of Berger [107, p.223–233], but Amelung’s work was criticized by the mathematician and chess champion Machgielis [Max] Euwe in his titanic 1940 study of pawnless endgames [270, pp.50–53].¹⁵

Indeed, *Düna-Zeitung* turned out to be an elusive newspaper; I was not able to locate any references to it in domestic catalogues and indices; the only copy I was able to find was archived at the National Library of Latvia. In addition to the remark about Molien, the research reported here argues for a renewed appreciation of the accuracy and importance of Amelung’s work.

6.1.4 Computer endgame analysis

Although some have dated computer chess from Charles Babbage’s brief discussion of automated game-playing in 1864, his conclusion suggests that he did not appreciate the complexities involved:

In consequence of this the whole question of making an automaton play any game depended upon the possibility of the machine being able to represent all the myriads of combinations relating to it. Allowing one hundred moves on each side for the longest game at chess, I found that the combinations involved in

¹⁵ Euwe wrote “Dit eindspel [] biedt de sterkste partij zeer goede winstkansen. F. Amelung ging zelfs zoo ver, dat hij de verdediging als kansloos beschouwde, maar deze opvatting schijnt ojuist te zijn [270, p.50]”, i.e., “This endgame [] offers the stronger side excellent winning chances. F. Amelung went so far as to say that the defense was hopeless, but this assessment seems to be untrue.” (Translation from the Dutch is by Peter Jansen; translation into German is available in Euwe [271, Volume 5, Page 55].)

the Analytical Engine enormously surpassed any required, even by the game of chess. [74, p. 467]

Automated endgame play appears to date from the ♔♚♔ construction of Leonardo Torres-Quevedo. Although some sources give 1890 as the date in which the automaton was designed, it was exhibited at about 1915 [97, 683].¹⁶ Quevedo’s automaton, which, unlike most later work, could move its own pieces, used a rule-based approach [731, 761], like that of Barbara J. Huberman’s 1968 thesis [385]. By contrast, we are concerned with exhaustive analysis of endgames, in which the value of each node of the state-space is computed by backing up the game-theoretic values of the leaves.

The mathematical justification for the retrograde analysis chess algorithm was already implicit in the 1912 paper of Ernst Zermelo [819]. Additional theoretical work was done by John von Neumann and Oskar Morgenstern (1944) [788, pp.124-125].

The contemporary dynamic programming methodology, which defines the field of retrograde endgame analysis, was discovered by Richard Bellman in 1965 [101].¹⁷ Bellman’s work was the culmination of his work reported as early as 1961:

Checkers and Chess. Interesting examples of processes in which the set of all possible states of the system is indescribably huge, but where the deviations are reasonably small in number, are checkers and chess. In checkers, the number of possible moves in any given situation is so small that we can confidently expect a complete digital computer solution to the problem of optimal play in this game.

¹⁶ “Torres believes that the limit has by no means been reached of what automatic machinery can do, and in substantiation of his opinions presents his automatic chess-playing machine” [731, p. 298].

¹⁷ Bellman’s article, strangely enough, is not generally known to the computer game community, and it is not included in Herik’s bibliography. [780]

In chess, the general situation is still rather complex, but we can use the method described above to analyze completely all pawn-king endings, and probably all endings involving a minor piece and pawns. Whether or not this is desirable is another matter [100, p.3].

Bellman had considered game theory from a classical perspective as well [98,99], but his work came to fruition in his 1965 paper [101], where he observed that the entire state-space could be stored and that dynamic programming techniques could then be used to compute whether either side could win any position. Bellman also sketched how a combination of forward search, dynamic programming, and heuristic evaluation could be used to solve much larger state spaces than could be tackled by either technique alone. Bellman predicted that checkers could be solved by his techniques, and the utility of his algorithms for solving very large state spaces has been validated by Jonathan Schaeffer et al. in the domain of checkers and Ralph Gasser in the domain of Nine Men's Morris [320,478,664]. On the other hand, $4 \times 4 \times 4$ tic-tac-toe has been solved by Patashnik (1980) using forward search and a variant of isomorph-rejection based on the automorphism group computation of Silver (1967) [591,681].

The first retrograde analysis implementation was due to Thomas Ströhlein, whose important 1970 dissertation described the solution of several pawnless 4-piece endgames [667,727,728].

E. A. Komissarchik and A. L. Futer (1974) studied certain special cases of , although they were not able to solve the general instance of such endgames [453]. J. Ross Quinlan (1979) analyzed  from the point of view of a machine learning testbed [611, 612]. Hans Berliner and Murray S. Campbell studied the Szén position of three connected

passed pawns against three connected passed pawns by simplifying the promotion subgames [112]. Campbell has begun to extend this idea to wider classes of endgames [157]. Peter J. Jansen has studied endgame play when the opponent is presumed to be fallible [400–402]. H. Jaap van den Herik et al. have produced a number of retrograde analysis studies of various 4-piece endgames, or of endgames with more than 4 pieces whose special structure allows the state-space size to be reduced to about the size of the general 4-piece endgame [234,778,781]. Danny Kopec has written several papers in the area as well [454].

The first retrograde analysis of general 5-piece endgames with up to one pawn was due to Thompson (1986) [753]. The significance of this work was twofold. First, many more moves were required to win certain endgames than had previously been thought. Second, the Thompson work invalidated generally accepted theory concerning certain 5-piece endgames by demonstrating that certain classes of positions that had been thought to be drawn were, in fact, won. The winning procedure proved to be quite difficult for humans to understand [553]. The pawnless 5-piece work of Thompson was extended to all pawnless 5-piece endgames and many 5-piece endgames with one pawn by an early version of the program discussed in this paper.

6.2 Tensor products and chess endgames

This section describes the chess endgame algorithm in a generalization of the tensor product formalism described in Section 4.3. The parallel chess endgame algorithm should be contrasted with the vast body of work on parallel forward search and in particular parallel

alpha-beta chess searching (references). The problem considered is easier to parallelize than forward search because of the absence of pruning. Some progress toward the problem of using combined forward and backward search in parallel searching in the context of symmetry is reported in work by Jonathan Bright, Simon Kasif, and the author, where the best-known bound on parallel knapsack algorithms is also improved [142].

Small chess endgames present a particularly interesting challenge to our multilinear-algebraic parallel-program design methodology:

- The formalism for the existing multilinear algebra approach had been developed to exploit parallelization of linear transformations over a module. This formalism needed to be generalized so that it would work over Boolean algebras.
- The symmetry under a noncommutative crystallographic group had to be exploited without sacrificing parallelizability.
- The state-space size of $7.7 \cdot 10^9$ nodes was near the maximum that the target architecture could store in RAM.

The remainder of this chapter describes the resolution of these problems, and reports on the following two main domain results:

1. Table 1 gives equations defining the dynamic programming solution to chess endgames. Using the techniques described in this paper, the factorizations can be modified to produce efficient code for most current parallel and vector architectures.

2. Table 2 presents a statistical summary of the state space of several 6-piece chess endgames. This table could not have been generated in a practicable amount of time using previous techniques.

6.2.1 Definitions

For the sake of simplicity of exposition, captures, pawns, stalemates, castling, and the 50-move rule will be disregarded unless otherwise stated.

Let S be an ordered set of k chess pieces. For example, if $k = 6$ then one could choose $S = \langle \♞, \♚, \♛, \♜, \♝, \♞ \rangle$.

An S -*position* is a chess position that contains exactly the k pieces in S . We write $S = \langle S_1, S_2, \dots, S_k \rangle$. An S -position can be viewed as an assignment of each piece $S_i \in S$ to a *distinct* square of the chessboard (note that captures are not allowed).

\mathfrak{B}_n is the space of length n Boolean vectors. The space of 8×8 Boolean matrices is thus $\mathfrak{C} \equiv \mathfrak{B}_8 \otimes \mathfrak{B}_8$. Let $\{\mathbf{e}_i\}_{i=1}^8$ be the standard basis for \mathfrak{B}_8 .

Let $\otimes^j \mathfrak{B}$ be the j th tensor power of \mathfrak{B} , *i.e.*, $\mathfrak{B} \otimes \dots \otimes \mathfrak{B}$, with j factors.

Let $\mathfrak{B} \equiv \otimes^k \mathfrak{C}$. \mathfrak{B} is called the *hyperboard* corresponding to S . It can be thought of as a cube of side-length 8 in \mathbb{R}^{2k} . Each of the 64^k basis elements corresponds to a point with integer coordinates between 1 and 8.

Each basis element of \mathfrak{C} is of the form $\mathbf{e}_i \otimes \mathbf{e}_j$ for $1 \leq i, j \leq 8$. Any such basis element, therefore, denotes a unique square on the 8×8 chessboard. Any element of \mathfrak{C} is a sum of

distinct basis elements, and therefore corresponds to a set of squares [801].

Each basis element of \mathfrak{B} is of the form $\mathbf{c}_1 \otimes \mathbf{c}_2 \otimes \cdots \otimes \mathbf{c}_k$, where each \mathbf{c}_s is some basis element of \mathfrak{C} . Since each \mathbf{c}_s is a square on the chessboard, each basis element of \mathfrak{B} can be thought of as a sequence of k squares of the chessboard. Each position that is formed from the pieces of S is thereby associated with a unique basis element of \mathfrak{B} . Any set of positions, each of which is formed from pieces of S , is associated with a unique element of \mathfrak{B} : the sum of the basis elements corresponding to each of the positions from the set.

This correspondence between sets of chess positions and elements of \mathfrak{B} forms the link between the chess algorithms and the tensor product formulation. In the following, the distinction between sets of chess positions formed from the pieces in S and elements of the hyperboard \mathfrak{B} will be omitted when the context makes the meaning clear.

If $p \in \{\text{♔}, \text{♕}, \text{♖}, \text{♗}, \text{♘}, \text{♙}\}$ is a piece, then the *unmove operator* $\text{III}_{\mathbf{p},s}$ is the function that, given an S -position P returns the set of S -positions that could be formed by unmoving S_s in P as if S_s were a p .

$\text{III}_{\mathbf{p},s}$ can be extended to a linear¹⁸ function from elements of \mathfrak{B} to itself, and thereby becomes an element of \mathfrak{M}_{64^k} .

The core of the chess endgame algorithm is the efficient computation of the $\text{III}_{\mathbf{p},s}$. The following subsections describe a factorization of $\text{III}_{\mathbf{p},s}$ in terms of primitive operators. The

¹⁸ Technically the unmove operators are only quasilinear, since the Boolean algebra is not a ring, and thus \mathfrak{B} is not a module.

ideas of subsection 4.3.2 may then be used to derive efficient parallel code from this factorization.

6.2.2 Group actions

This subsection introduces a few group actions [310]. We will use the group-theoretic terminology both to give concise descriptions of certain move operators and to describe the exploitation of symmetry. There is a close correspondence between multilinear algebra, combinatorial enumeration, and group actions which motivates much of this section [547–550].

The symmetric group on k elements \mathfrak{S}_k acts on \mathfrak{B} by permuting the order of the factors:

$$\mathfrak{s} \bigotimes_{s=1}^k \mathbf{c}_s = \bigotimes_{s=1}^k \mathbf{c}_{\mathfrak{s}s},$$

for $\mathfrak{s} \in \mathfrak{S}_k$ and $\mathbf{c}_s \in \mathfrak{C}$.

The dihedral group of order 8, \mathfrak{D}_4 , (see section 3.1) acts on \mathfrak{C} by

$$\mathfrak{r}(\mathbf{e}_i \otimes \mathbf{e}_j) = \mathbf{e}_{8-j+1} \otimes \mathbf{e}_i \tag{6.1}$$

$$\mathfrak{f}(\mathbf{e}_i \otimes \mathbf{e}_j) = \mathbf{e}_i \otimes \mathbf{e}_{8-j+1} \tag{6.2}$$

Thus, \mathfrak{r} rotates the chessboard counterclockwise 90° and \mathfrak{f} flips the chessboard about the horizontal bisector.

\mathfrak{D}_4 acts diagonally on \mathfrak{B} :

$$\mathfrak{d} \bigotimes_{s=1}^k \mathbf{c}_s = \bigotimes_{s=1}^k \mathfrak{d}\mathbf{c}_s$$

Let \mathfrak{C}_4 be the cyclic group generated by τ .

A group \mathfrak{G} acting on \mathfrak{V}_n and \mathfrak{V}_m acts on \mathfrak{M}_n^m by conjugation: $(\mathfrak{g}M)\mathbf{v} = \mathfrak{g}(M\mathfrak{g}^{-1}(\mathbf{v}))$. We let

$$\int_{\mathfrak{G}} x = \sum_{\mathfrak{g} \in \mathfrak{G}} \mathfrak{g}x.$$

The notation $\int_{\mathfrak{G}} x$ is intended to represent the group average of x with respect to \mathfrak{G} [310, p. 6]. It is a fixed point of the \mathfrak{G} action: $\mathfrak{g}\int_{\mathfrak{G}} x = \int_{\mathfrak{G}} x$ for all $\mathfrak{g} \in \mathfrak{G}$.

6.3 Endgame algorithm

This section presents the endgame algorithm using the notation developed in Section 6.2. Subsection 6.3.1 gives the fundamental factorization. Subsection 6.3.2 describes the modification of the equations of Table 6.1 to exploit symmetry. Subsection 6.3.3 describes the control structure of the algorithm.

6.3.1 Factorizing the unmove operator

We define E_8 to be the unit one-dimensional 8×8 end-off shift matrix. The unit multidimensional shift along dimension s is defined by

$$U_s \in \mathfrak{M}_{64^k} \equiv I_{64^{s-1}} \otimes (E_8 \otimes I_8) \otimes I_{64^{k-s}}.$$

Such multidimensional shifts are commonly used in scientific computation.

$$\text{III}_{\boxplus,s} = \int_{\mathfrak{C}_4} \text{LU}_s (l_{64^k} + \text{LU}_s)^6 \quad (6.4)$$

$$\text{III}_{\boxdot,s} = \text{L} \int_{\mathfrak{D}_4} \text{U}_s \cdot (\tau(\text{U}_s^2)) \quad (6.5)$$

$$\text{III}_{\boxminus,s} = \int_{\mathfrak{D}_4} \text{LU}_s (l_{64^k} + \text{LU}_s \tau \text{U}_s)^6 \quad (6.6)$$

$$\text{III}_{\boxtimes,s} = \text{L} \int_{\mathfrak{C}_4} \text{U}_s + \text{U}_s \tau \text{U}_s \quad (6.7)$$

$$\text{III}_{\boxplus,s} = \text{III}_{\boxplus,s} + \text{III}_{\boxminus,s} \quad (6.8)$$

$$(6.9)$$

Table 6.1: These equations define the core of a portable endgame algorithm. By modifying the factorizations, code suitable for execution on a wide range of high-performance architectures can be derived.

Fix a basis $\{\mathbf{c}_i\}_{i=1}^{64}$ of \mathfrak{C} , and define

$$\text{L} \in \mathfrak{M}_{64^k} \equiv \text{diag} \left(\int_{\mathfrak{C}_k} \sum_{i_1 < \dots < i_k} \mathbf{c}_{i_1} \otimes \dots \otimes \mathbf{c}_{i_k} \right) \quad (6.3)$$

Certain basis elements of \mathfrak{B} do not correspond to legal S -positions. These “holes” are elements of the form $\bigotimes_{s=1}^k \mathbf{c}_s$ such that there exist distinct s, s' for which $\mathbf{c}_s = \mathbf{c}_{s'}$. If $\mathbf{v} \in \mathfrak{B}$ then $\text{L}\mathbf{v}$ is the projection of \mathbf{v} onto the subspace of \mathfrak{B} generated by basis elements that are not holes.

Table 6.1 defines the piece-unmove operators.

Figure 6.1 illustrates the computation of the integrand in the expression for $\text{III}_{\boxplus,1}$ in Table 6.1. This corresponds to moving the \boxplus to the right. The average over \mathfrak{C}_4 means that the \boxplus must be moved in 4 directions. For example, conjugation by τ of the operation of moving the \boxplus right corresponds to moving the \boxplus up: if one rotates the chessboard clockwise 90° , moves the \boxplus right, and then rotates the chessboard counterclockwise 90° , the result will be

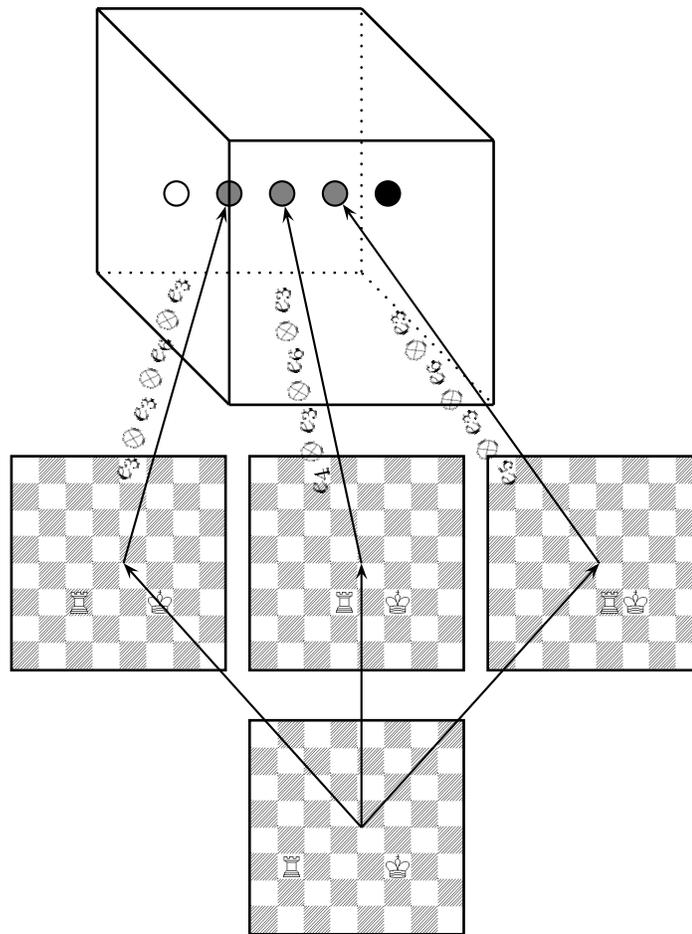


Figure 6.1: Unmoving the ♔ to the right from the position at bottom results in the three positions center. Here, $S = \langle \text{♖}, \text{♔} \rangle$. Each position corresponds to a point in the hyperboard, top. The bottom position is $\mathbf{e}_2 \otimes \mathbf{e}_3 \otimes \mathbf{e}_6 \otimes \mathbf{e}_3$. The new positions are $\mathbf{e}_3 \otimes \mathbf{e}_3 \otimes \mathbf{e}_6 \otimes \mathbf{e}_3$, $\mathbf{e}_4 \otimes \mathbf{e}_3 \otimes \mathbf{e}_6 \otimes \mathbf{e}_3$, and $\mathbf{e}_5 \otimes \mathbf{e}_3 \otimes \mathbf{e}_6 \otimes \mathbf{e}_3$. $\mathbf{e}_6 \otimes \mathbf{e}_3 \otimes \mathbf{e}_6 \otimes \mathbf{e}_3$ is illegal and is zeroed out by L .

the same as if the ♚ had been moved up to begin with.

As explained in section 4.3, by varying the factorization, code suitable for varying architectures can be derived. For example, if the interconnection architecture is a 2-dimensional grid, then only U_s for $s = 1$ can be directly computed. By using the relations $U_s = (1\ s)U_1$ and $\text{III}_{\mathbf{p},s} = (1\ s)\text{III}_{\mathbf{p},1}$, equations appropriate for a grid architecture can be derived. Here $(1\ s) \in \mathfrak{S}_k$ interchanges 1 and s .

These equations are vectorizable as well [698]. The vectorized implementation of Table 1 by Burton Wendroff et al. has supported this claim [799].

Other factorizations appropriate for combined vector and parallel architectures, such as a parallel network of vector processors, can also be derived [434].

6.3.2 Exploiting symmetry

The game-theoretic value of a chess position without pawns is invariant under rotation and reflection of the chessboard. Therefore, the class of positions considered can be restricted to those in which the ♚ is in the lower left-hand octant, or fundamental region, of the chessboard (Figure 6.2).

The chess positions with the ♚ in its fundamental region correspond to points in a triangular wedge in the hyperboard.

Algebraically, because each $\text{III}_{\mathbf{p},s}$ is a fixed point of the \mathfrak{D}_4 action, we need only consider

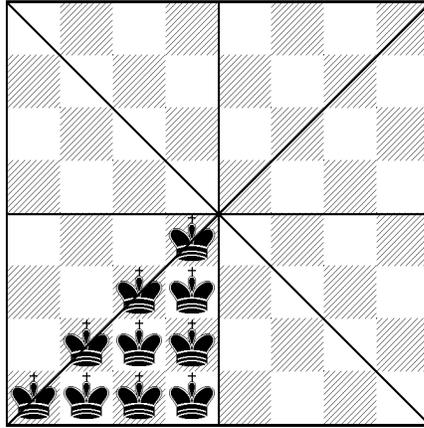


Figure 6.2: The chessboard may be rotated 90° or reflected about any of its bisectors without altering the value of a position without pawn. Therefore, the location of the ♔ may be restricted to the 10 squares shown, called a fundamental region.

the $10 \cdot 64^{k-1}$ -space:

$$\mathfrak{B}' \equiv \mathfrak{C}/\mathfrak{D}_4 \otimes \bigotimes^{k-1} \mathfrak{C}$$

rather than the 64^k -space \mathfrak{B} . We suppose that the first piece of S , the piece corresponding to the first factor in the expression for \mathfrak{B}' , is the ♔.

When pieces other than the ♔ are moved, the induced motion in the hyperboard remains within the wedge. Thus, the induced functions $\text{III}'_{\mathfrak{p},s}: \mathfrak{B}' \mapsto \mathfrak{B}'$ have the same form as Table 6.1 when $s \geq 1$.

However, when the ♔ is moved outside its fundamental region, the resulting position must be transformed so that the ♔ is in its fundamental region. This transformation of the chessboard induces a transformation on the hyperboard (Figure 6.3).

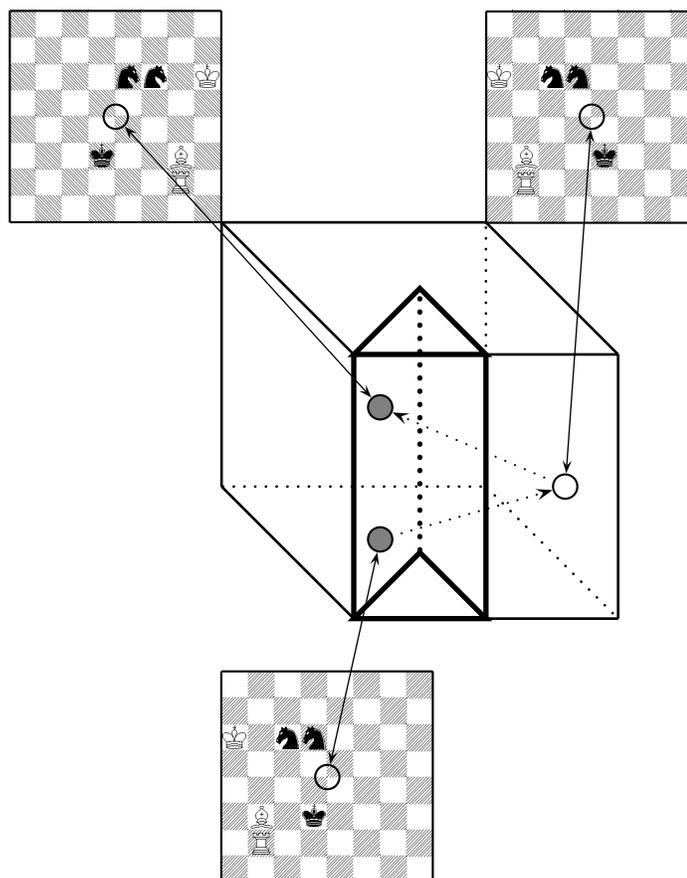


Figure 6.3: Only a wedge in the hyperboard is physically stored. To compute the effect of moving the ♔ outside the squares to which it is restricted, a communication pattern is induced in the hyperboard. In this example, the ♔ in the lower chessboard is moved, reaching the position at the upper right. This position must be reflected about the vertical bisector, yielding the position at upper left. These three positions correspond to three points in the hyperboard, only the first and third of which are physically stored. The Black-to-move position at bottom requires 222 moves against best play for White to win (see Table 2).

Algebraically,

$$\mathbb{H}'_{\mathfrak{g},1} = \sum_{\mathfrak{d} \in \mathfrak{D}_4} \mathbb{H}'_{\mathfrak{g},1_{\mathfrak{d}}} \otimes \bigotimes^{k-1} \mathfrak{d} \quad (6.10)$$

where $\mathbb{H}'_{\mathfrak{g},1_{\mathfrak{d}}} \in \mathfrak{M}_{10}$.

The sum over $\mathfrak{d} \in \mathfrak{D}_4$ corresponds to routing along the pattern of the Cayley graph of \mathfrak{D}_4 (see Figure 6.4).

This is a graph whose elements are the 8 transformations in \mathfrak{D}_4 , and whose edges are labeled by one of the generators \mathfrak{r} or \mathfrak{f} . An edge labeled \mathfrak{h} connects node \mathfrak{g} to node \mathfrak{g}' if $\mathfrak{h}\mathfrak{g} = \mathfrak{g}'$. The communication complexity of the routing can be reduced by exploiting the Cayley graph structure [708]. The actual communication pattern used is that of a group action graph, which looks like a number of disjoint copies of the Cayley graph, together with some cycles [800].

The problem of parallel application of a structured matrix to a data set invariant under a permutation group has been studied in the context of finite-element methods by Danny Hillis and Washington Taylor as well. Although their terminology is different from our terminology, their general ideas are similar [368]. The method we use turns out to be similar to the orbital exchange method, which is used to compute the FFT of a data set invariant under a crystallographic group [49, 50, 757].

It is interesting to note that exploiting symmetry under interchange of identical pieces can be handled in this notation: j identical pieces correspond to a factor $\text{Sym}^j \mathfrak{C}$ in the expression for \mathfrak{C} , where Sym^j is the j th symmetric power of \mathfrak{C} . [310, pp.472–475]

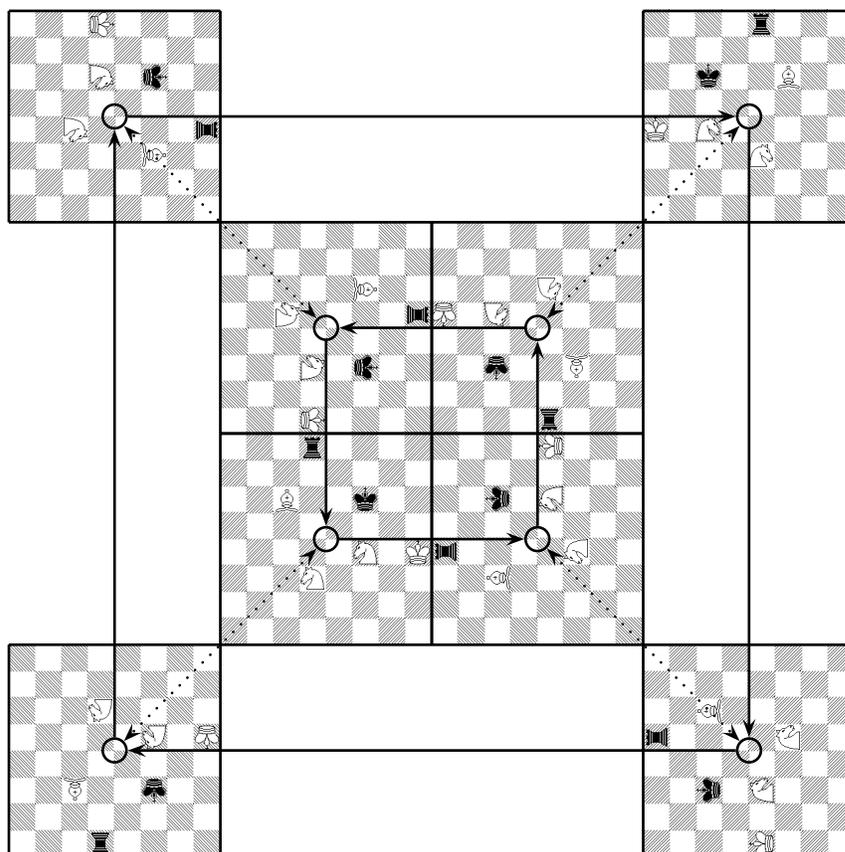


Figure 6.4: The Cayley graph for \mathcal{D}_4 . Each node is pictured by showing the effect of its corresponding transformation on a position in $\textcircled{\text{K}}\textcircled{\text{Q}}\textcircled{\text{N}}\textcircled{\text{B}}\textcircled{\text{P}}\textcircled{\text{R}}\textcircled{\text{K}}\textcircled{\text{Q}}\textcircled{\text{N}}\textcircled{\text{B}}\textcircled{\text{P}}\textcircled{\text{R}}\textcircled{\text{K}}$; thus, the chess value of each of these nodes is the same. Solid lines correspond to τ , and rotate the board counterclockwise 90° . Dotted lines correspond to f , and flip the board horizontally. The position shown arose during a game between Anatoly Karpov and Gary Kasparov in Tilburg, October 1991.

There are efficient algorithms, in general, for performing the purely algebraic operations required, as well as languages, such as GAP, MAGMA, and AXIOM, that are suitable for the denotation of the algebraic structures used [152, 684, 685]. The groups encountered here are so small, however, that computer-assisted group-theoretic computation was not required.

6.3.3 Control structure

For $i \geq 1$ we define $\mathbf{v}_i \in \mathfrak{B}$ to be vector of positions from which White to move can checkmate Black within i moves (i.e., i White moves and i Black moves). Thus, \mathbf{v}_1 is the vector of positions from which White can checkmate Black on the next move. \mathbf{v}_2 is the set of positions from which White can either checkmate Black in one move or can move to a position from which any Black reply allows a mate-in-one, and so on.

The overall structure of the algorithm is to iteratively compute the sets $\mathbf{v}_1, \mathbf{v}_2, \dots$ until some i is reached for which $\mathbf{v}_i = \mathbf{v}_{i+1}$. Then $\mathbf{v} = \mathbf{v}_i$ is the set of positions from which White can win, and i is the *maximin* value of the set S : the maximum, over all positions from which White can win, of the number of moves required to win that position [727, 753].

The method for computing \mathbf{v}_i from \mathbf{v}_{i-1} is called the backup rule. Several backup rules have been used in various domains [478, 664]. They are all characterized by the use of an *unmove generator* to “unmove” pieces, or move them backward, possibly in conjunction

with more traditional move generators. We let

$$\mathbb{III}_{\text{White}} \equiv \sum_{\{s:S_s \text{ is White}\}} \mathbb{III}_{S_s,s} \quad (6.11)$$

$$\mathbb{III}_{\text{Black}} \equiv \sum_{\{s:S_s \text{ is Black}\}} \mathbb{III}_{S_s,s} \quad (6.12)$$

The backup rule used is:

$$\mathbf{v}_{i+1} = \mathbb{III}_{\text{White}}(\overline{\mathbb{III}_{\text{Black}}(\overline{\mathbf{v}}_i)}). \quad (6.13)$$

Here, $\overline{\mathbf{v}}$ denotes the complement of \mathbf{v} .

6.4 Implementation notes

6.4.1 Captures and pawns

The algorithms developed so far must be modified to account for captures and pawns.

Each subset of the original set of pieces S induces a *subgame*, and each subgame has its own hyperboard [101]. Without captures, moving and unmoving are the same, but when captures are considered they are slightly different. The equations for $\mathbb{III}_{\mathbf{p},s}$ developed in the preceding section refer to *unmoving* pieces, not to moving them [753]. Unmoving pieces cannot capture, but they can uncapture, leaving a piece in their wake. This is simulated via interhyperboard communication.

The uncapture operation can be computed by using tensor products, corresponding to the parallel broadcast (see section 8.1). An uncapture is the product from left to right of an

unmove operator in the parent game, a diagonal matrix, a sequence of stride matrices, and a broadcast. The broadcast is a tensor product of copies of an identity matrix with the 1×64 matrix of 1's.

Each pawn position induces a separate hyperboard. Pawn unpromotion induces communication between a quotient hyperboard and a full hyperboard, which is implemented again by multiplication by \mathfrak{D}_4 .

6.4.2 Database

There are two values that can be associated with a position: *distance-to-mate* and *distance-to-win*.

The distance-to-mate is the number of moves by White required for White to checkmate Black, when White plays so as to checkmate Black as soon as possible, and Black tries to avoid checkmate as long as possible [819]. Although the distance-to-mate might seem like the natural metric to use, it can produce misleadingly high distance values because the number of moves to mate in trivial subgames, like $\text{♔} \text{♖} \text{♗}$, would be included in the count of something like $\text{♔} \text{♖} \text{♗} \text{♘}$. In fact, in $\text{♔} \text{♖} \text{♗} \text{♘}$, it does not matter for most purposes how many moves are required to win the subgame $\text{♔} \text{♖} \text{♗}$, once White captures the ♘ , as long as the ♘ is captured safely [616].

The more usual distance-to-win metric is simply the number of moves required by White to force conversion into a winning subgame. In practice, this metric is more useful when the

position has no pawns. It also is the metric of relevance to the 50-move rule. If a particular position has a distance-to-win of m , then against perfect play the win value would be altered by an m' move rule for $m' > m$. Although our program has implemented distance-to-mate metric for 5-piece endgames, the results presented here use the more conservative distance-to-win metric.

The *max-to-win* for a set of pieces (i.e., an endgame) is the maximum, over all positions using those pieces from which White can win, of the distance-to-win of that position.

The distance-to-win of each point in the hyperboard can be stored so that a 2-ply search permits optimum play.

By Gray coding this distance, the increment of the value can be done by modifying only one bit.

Curiously, the motif of embedding Cayley graphs into Cayley graphs arises several times in this work. Gray codes, which can be viewed as embedding the Cayley graph for \mathbb{Z}_2^n into that of \mathbb{Z}_2^n , are used both for implementing \mathbf{U} (and, therefore, $\mathbf{III}_{\mathbf{P},s}$) and for maintaining the database. Embedding the Cayley graph for \mathfrak{D}_4 in that of \mathbb{Z}_2^n arises during unpromotion and moving the $\text{\textcircled{c}}$. Because many interconnection networks are Cayley graphs or group action graphs [56,246,248,636], this motif will reappear on other implementations.

6.5 Results

6.5.1 Chess results

The combinatorially possible pawnless 5-piece games and many 5-piece games with a single pawn were solved using an early version of the current program. This work resulted in the first publication of the 77-move ♔♕♖♗♘ max-to-win, which at the time was the longest known pawnless max-to-win [707]. Some endgames were solved under the distance-to-mate metric as well. The distance-to-mate results were not particularly illuminating. The state space size is approximately $121 \cdot 10^6$ nodes for a single pawnless 5-piece endgame under the Thompson symmetries, in which one representative from the 462 orbits of the \mathfrak{D}_4 action on the nonadjacent positions of the two kings is stored.

Several pawnless 6-piece endgames were also solved. The state-space size per endgame was 6,185,385,360 nodes, although the size of each hyperboard is $462 \cdot 64^4$, or about $7.7 \cdot 10^9$.

Table 6.2 presents statistical information about some pawnless 6-piece endgames.

The percent-win can be misleading because of the advantage of the first move in a random position—White can often capture a piece in one move—and because it includes positions in which Black is in check.

The max-to-win values were significantly higher than previously known endgames. No 5-piece endgame had a max-to-win over 100, and most of the nontrivial ones had max-to-wins of approximately 50. ♔♖♗♘♙♚ has the longest known max-to-win of 243, although it is

Game	W	Wins	%W	Z	Game	W	Wins	%W	Z
	243	4821592102	78	18176		63	5257968414	85	6670
	223	5948237948	96	456		54	4529409548	73	1030
	190	4433968114	72	8030		52	1015903231	65	256
	153	5338803302	86	1858		51	5058432960	82	2820
	140	4734636964	77	1634		49	3302327120	53	1270
	101	5843483696	94	1520		48	5689213742	92	32
	99	4242312073	69	1010		46	4079610404	66	22
	98	5359504406	87	1478		44	5122186896	83	32
	92	5324294232	86	6300		44	1185941301	75	396
	92	5125056553	83	243		38	981954704	63	1662
	86	5834381682	94	12918		37	1483910528	94	26
	85	5707052904	92	342		36	4213729734	68	78
	82	5935067734	96	388		35	4626525594	75	17688
	75	1123471668	72	95		32	3825698576	62	6
	73	5365200098	87	1410		32	3789897570	61	35
	73	5023789102	81	1410		31	6130532196	99	58
	72	5808880660	94	2228		29	3920922433	63	152
	71	5553239408	90	1780		27	3533291870	57	3
	69	4944693522	80	48		18	4136045492	67	16
	68	1497242834	95	83		12	970557572	62	18
	65	6054654948	98	6					

Table 6.2: Endgame description, maximin, number of wins, percent-win, and number of mutual zugzwangs for certain 6-piece endgames. The symmetries considered are the Thompson symmetries in which the one representative from each of the 462 orbits of the non-adjacent king-positions is stored, except for endgames with a \star , which indicates that the two bishops were constrained to lie on squares of opposite colors. A state-space size of 6,185,385,360 for normal endgames and 1,570,867,920 for endgames with a \star was used. Thus, for example, there is really only a single mutual zugzwang in , but it is counted 6 times.

not a general win.

We remark that ♔♚♙♗♘♘ is a general win, with 223 moves required to win in the worst case. ♔♚♙♗♘♘ was called “known to be a draw” by Roycroft, a leading endgame expert, in 1972 (♔♙♙♗♘♘), which was considered a draw by most players, was only “controversial or unknown” according to the same source). Most of the standard works concurred with the opinion that ♔♚♙♗♘♘ was not a general win [270, pp.50–53], [178, p. 417], [106, pp.167–169], [291, p. 521]. Chéron, however, seems to reserve judgment.

The 50-move rule would affect the value of each endgame listed with max-to-win of 50 or more. The 92-move win in ♔♚♚♚♚♚♚ is somewhat surprising too.

A mutual zugzwang is closely related to a game whose Conway value is 0: it is a position in which White to move can only draw, but Black to move loses. Such positions seem amusing because, particularly when no pawns are involved, chess is a very “hot” game in the sense of *Winning Ways* [110].

Unlike the “maximin” positions (see appendix) whose analysis is fairly impenetrable, the mutual zugzwangs can sometimes be understood by humans.

An example may help clarify this concept. Figure 6.5 shows a mutual zugzwang discovered by the program, in ♔♙♘♘♗♗♚. The Black ♚ is trapped on **h8**,¹⁹ since **g8** is guarded by the ♙ on **a2**, and the ♘’s guard each other. If the Black ♚ were to capture a ♘, then

¹⁹ The columns of a chessboard are conventionally lettered from left to right with letters going from **a** to **h**; the rows of the chessboard are numbered from **1** to **8** reading up the page. Thus, **h8** is the square on the upper right corner of the board.

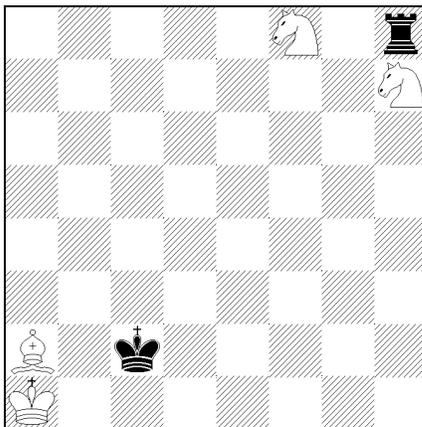


Figure 6.5: Mutual zugzwang: White to play draws, but Black to play loses

it would in turn be captured, and the resulting subgame of ♔♚♜♔ would be winning for White. The position seems to be a race between Kings to see who will reach the upper right corner area first. If the Black ♔ reaches **g7** or **e8** first, the Black ♖ can sacrifice itself for a White ♞, and then the Black ♔ captures the other White ♞, leaving the drawn endgame ♔♚♔. On the other hand, if the White ♔ reaches **g7** first, it simply captures the Black ♖**h8**. Note also that neither ♞ can move, as the ♖ would immediately capture the other ♞.

It is not difficult to see that Black to play loses: White gets in first. For example, **1 ... ♔c3**
2 ♔b1 ♔d4 **3 ♔c2 ♔c5** (If **3... ♔e5?** **4 ♞g6+** wins the ♖) **4 ♔d3 ♔d6** **5 ♔e4 ♔c7** (If
 ♔e7? **6 ♞g6+** wins) **6 ♔f5 ♔d8** **7 ♔g6 ♔e8** **8 ♔g7** and White wins.

However, White to move from the position in Figure 6.5 must move the ♚. **1 ♚b1+** ♔c3
 forces **2 ♚a2 ♔c2**, since other moves by White on the second move allow the ♖**h8** to
 escape via **g7**. Chess theory, confirmed by the program, shows that this general position in

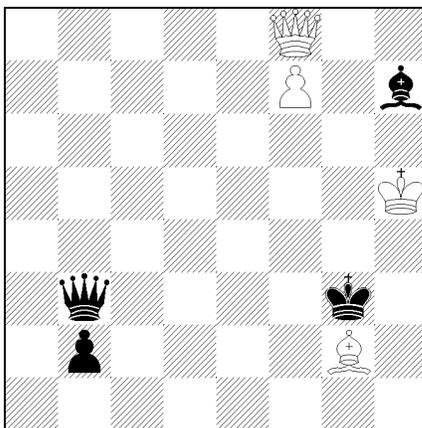


Figure 6.7: Elkies, *American Chess Journal* 1(2) 1994. White to play and win. “1 ♖g7+ Not 1 ♜d6+? ♙xg2 2 f8/♜ (interpolating further checks does not help) when 2...♜h3+ 3 ♘g5 ♜e3+ forces either perpetual check or a queen trade, drawing. 1...♚h2 2 f8/♜. If 2 ♜e5+ ♙xg2 3 f8/♜ ♜h3+ 4 ♘g5 b1/♜ with ♚h1 and ♙e4 draws, but now 2...b1/♜ loses to 3 ♜f4+ ♙g1 4 ♙e4+ and mate. Thus, Black tries for perpetual check, and not with 2...♜d1+? 3 ♙f3. 2...♜b5+ 3 ♚h6 ♜b6+ 4 ♙c6! Not yet 4 ♙xh7 b1/♜+ 5 ♚h8 ♜b8! drawing. Now Black must take the bishop because 4...♜e3+ 5 ♜g5 ♜xg5+ 6 ♙xg5 b1/♜ 7 ♜f2+ mates. 4...♜xc6+ 5 ♙xh7 b1/♜+. So Black does manage to give the first check in the four-queen endgame, but he is still in mortal danger. 6 ♙h8 ♙h1! Black not only cannot continue checking, but must play this modest move to avoid being himself checked to death! For instance, 6...♜g2 7 ♜c7+ ♚g1 8 ♜fc5+ ♚h1 9 ♜h5+ and the Black king soon perishes from exposure. But against the quiet 6...♚h1 White wins only with 7 ♜fg8!!, a second quiet move in this most tactical of endgames, bringing about” the rotated version of the ♚♜♜♜♜ mutual zugzwang. (Quotation from Elkies’ analysis)

was unclear whether a draw could have been obtained. In fact, however, pawnless 6-piece endgames almost never arise in tournament play.

6.5.2 Timing

The implementation was on a 64K processor CM-2/200 with 8 GBytes RAM. The processors were interconnected in a hypercube and clocked at 7MHz (10 MHz for the CM-200). The CM-2 6-piece code required approximately 1200 seconds for initialization and between 111 and 172 seconds to compute K_{i+1} from K_i . Exact timings depend on S (for instance, as is clear from Table 1, $\text{III}_{\text{w},s}$ is slower than either $\text{III}_{\text{m},s}$ or $\text{III}_{\text{d},s}$) as well as run-time settable factorization choices and load on the front end.

Per-node time per endgame (time to solve the endgame divided by number of nodes in the state-space) is faster by a factor of approximately $6 \cdot 10^3$ than timings *of different endgames* reported using classical techniques [579,753,779,781] based on the 5-piece timings of the code reported here.

In unpublished personal communication Thompson has indicated that the per-node time of the fastest serial endgame code is currently only a factor of approximately $7 \cdot 10^2$ times slower than that of the code reported in this paper (depending on the endgame) [754].

Unfortunately, direct comparison of 6-piece timing against other work is, of course, not currently possible since 6-piece endgames could not have been solved in a practicable amount of time using classical techniques on previous architectures. However, with larger and

faster serial machines, and with enough spare cycles, 6-piece endgames are in fact coming within reach of classical solution techniques. This would permit a more informative timing comparison.

Thus, although per-node timing comparisons based on radically differently sized state-spaces are not very meaningful, the large per-node timing differential of the current program compared to classical programs does tend to support the hypothesis that the techniques reported here lend themselves to efficient parallel implementation.

The only program with per-node time of comparable speed to the author's CM-200 implementation is Burton Wendroff's et al. vectorized implementation of Table 1 [799], although this implementation currently solves only a single 4-piece endgame.

The CM-200 source code implementing Table 1 is currently available from `ftp.cs.jhu.edu:pub/stiller/snark`.

6.6 Future work

The main historical open question is to find out what was Molien's exact contribution to the history of numerical chess endgame analysis, and to locate and check his analysis of ♔♙♗♘♚. Kanunov [422, p.6] refers to private papers held by Molien's daughter; currently we are trying to locate these papers in the hope that they might shed light on the questions raised in subsection 6.1.3. Amelung himself is also a figure about whom little is known, and the remarks here would seem to suggest that a detailed reassessment of his contribution to

the endgame study would be desirable.

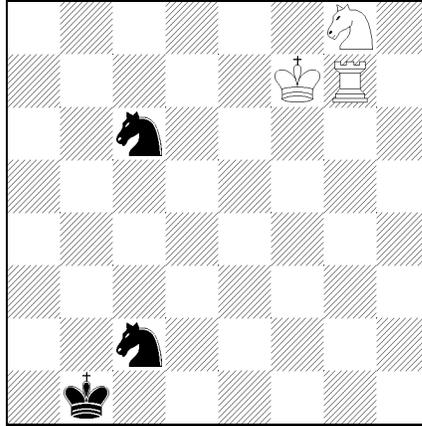
The question of Molien and Amelung's contributions to quantitative endgame analysis is part of the larger historical question of pre-digital precursors to computer chess algorithms. In addition to the work of Babbage, Molien, Amelung, Zermelo, and Quevedo, we remark that K. Schwarz, in a little-known 1925 article in *Deutsche Schachzeitung*, argued for a positional evaluation function similar to the squares-attacked heuristic used in some full-chess programs [673].

From a computational point of view, it might seem that the next logical step in the evolution of the current program should be the exhaustive solution of pawnless 7-piece endgames. In fact, in my opinion a more promising approach would be to follow up on the suggestions first made by Bellman [100, 101, 112] and solve endgames with multiple pawns and minor pieces. Such an approach would combine heuristic evaluation of node values corresponding to promotions with the exhaustive search techniques described here. Although the use of heuristics would introduce some errors, the results of such a search would, in my opinion, have considerable impact on the evaluation of many endgames arising in practical play.

Even more speculatively, it is also possible to search for certain classes of endgames considered artistic by endgame composers; such endgames typically depend on a key mutual-zugzwang or domination position some moves deep in the tree.

6.7 A best play line

Distance-to-win (conversion) metric is used. Equioptimal moves are parenthesized. For technical reasons the last move in the line is omitted.



1 ♔f7-e6 ♘c6-b4 2 ♖e6-e5 ♗b4-d3 3 ♕e5-e4 ♞d3-f2 4 ♗e4-f3 ♝f2-d3
 5 ♜f3-e2 ♘c2-b4 6 ♖e2-e3 ♗b1-b2 7 ♕e3-d4 ♞d3-f4 8 ♗d4-c4 ♗b4-d5
 9 ♞g7-h7 ♘d5-e3 10 ♕c4-d4 ♗e3-c2 11 ♗d4-e4 ♗f4-e6 12 ♖e4-e5 ♗e6-g5
 13 ♞h7-h5 ♘c2-e1 14 ♖e5-f5 ♗g5-f3 15 ♕f5-e4 (♗f5-f4) ♗f3-d2 16 ♖e4-e3
 ♘d2-b3 17 ♞h5-h1 ♗e1-c2 18 ♖e3-d3 ♗b3-c1 19 ♗d3-e4 ♗c1-b3 20 ♞h1-
 h3 ♗b3-c5 21 ♖e4-e5 ♘c2-e1 22 ♗g8-f6 ♗e1-d3 23 ♖e5-d6 ♗c5-b7
 24 ♗d6-c7 ♗b7-c5 25 ♖c7-c6 ♗b2-c2 26 ♞h3-h2 ♖c2-b3 (♖c2-c3) 27 ♖c6-d5
 ♖b3-b4 28 ♗d5-d4 (♞h2-h4) ♘d3-f4 29 ♞h2-h4 ♖b4-b5 30 ♗f6-e8 ♗c5-
 b3 31 ♗d4-e4 ♗f4-g6 32 ♞h4-h7 ♗b3-c5 33 ♖e4-d4 ♗g6-f4 34 ♗e8-d6
 ♖b5-c6 35 ♞h7-h6 ♗c5-b3 36 ♗d4-e4 ♗f4-e6 37 ♖e4-e5 ♗e6-d4 38 ♞h6-
 h3 ♗b3-c5 39 ♘d6-c8 ♗d4-c2 40 ♞h3-c3 ♗c2-b4 41 ♖e5-d4 ♗b4-a6
 42 ♞c3-c2 ♖c6-d7 43 ♗c8-b6 ♖d7-d6 44 ♗b6-c4 ♖d6-c6 45 ♗c4-e3 ♖c6-
 d6 46 ♗e3-f5 ♖d6-e6 47 ♗f5-g7 ♖e6-f7 48 ♗g7-h5 ♗c5-e6 49 ♗d4-e5
 ♗a6-b4 50 ♞c2-e2 ♗b4-d3 51 ♖e5-e4 ♗d3-b4 52 ♞e2-b2 ♖f7-g6 53 ♗h5-
 g3 ♗e6-g5 54 ♖e4-d4 ♗g5-e6 55 ♗d4-c4 ♗b4-a6 56 ♞b2-f2 ♗e6-g5
 57 ♞f2-f1 ♗a6-c7 58 ♗g3-e2 ♗g5-f7 59 ♗e2-f4 ♖g6-g5 60 ♖c4-d4 ♗c7-
 b5 61 ♗d4-c5 ♗b5-d6 62 ♗f4-e6 ♖g5-g6 63 ♗e6-f8 ♖g6-g5 64 ♖c5-d5
 ♗d6-f5 65 ♞f1-b1 (♞f1-a1) ♗f5-g3 66 ♞b1-b7 ♗f7-h6 67 ♞b7-g7 ♖g5-f4
 68 ♗f8-e6 ♖f4-f3 69 ♞g7-b7 ♗g3-h5 70 ♞b7-b4 ♗h5-f6 71 ♗d5-d4 ♗f6-
 h5 72 ♖d4-d3 ♗h6-g4 73 ♗e6-g5 ♖f3-g3 74 ♗g5-e4 ♖g3-h4 75 ♞b4-a4
 ♗h5-f4 76 ♖d3-d4 ♗f4-e6 (♗f4-e2) 77 ♗d4-d5 ♗e6-f4 78 ♖d5-d6 ♗f4-h3
 79 ♞a4-a8 ♗g4-f2 80 ♗e4-c5 ♖h4-g5 81 ♗d6-e5 ♗f2-g4 82 ♖e5-d4 ♗h3-f4
 83 ♗c5-e4 ♖g5-g6 84 ♞a8-a6 ♖g6-f5 85 ♞a6-a5 ♖f5-e6 86 ♗e4-c5 ♖e6-e7

87 ♠a5-a7 ♣e7-f6 88 ♠d4-e4 ♣f6-g5 89 ♠a7-a5 ♠f4-h5 90 ♠c5-e6 ♣g5-g6
 91 ♠a5-b5 ♣g6-f7 92 ♠e6-c5 ♣f7-e7 93 ♠b5-b2 ♣e7-d6 94 ♠c5-b7 ♠d6-e7
 95 ♠b2-a2 ♠h5-g7 96 ♠a2-e2 ♣e7-d7 97 ♠e2-g2 ♠g7-e8 98 ♠e4-f4 ♠g4-f6
 99 ♠f4-e5 ♠d7-e7 100 ♠g2-e2 ♣e7-d7 101 ♠b7-a5 ♠f6-g4 102 ♠e5-f5 ♠g4-
 h6 103 ♠f5-g6 ♠h6-g8 104 ♠a5-c4 ♠e8-c7 105 ♠g6-f7 ♠g8-h6 106 ♠f7-f6
 ♠h6-g8 107 ♠f6-e5 ♠g8-e7 108 ♠e2-d2 ♠d7-c6 109 ♠d2-c2 ♠c7-a6 (♠e7-g6)
 110 ♠c4-e3 ♠c6-d7 111 ♠c2-d2 ♠d7-c6 112 ♠d2-d6 ♠c6-b5 113 ♠d6-h6
 ♠e7-c8 114 ♠e5-d4 (♠h6-h5) ♠a6-b4 115 ♠h6-h5 ♠b5-c6 116 ♠e3-c4 ♠c8-e7
 117 ♠h5-h6 ♠c6-c7 118 ♠h6-h7 ♠c7-d7 119 ♠d4-e5 ♠b4-d5 120 ♠c4-d6
 (♠c4-d2) ♠d7-c6 121 ♠d6-e4 ♠e7-g6 122 ♠e5-f5 ♠g6-f8 123 ♠h7-h6 ♠c6-c7
 124 ♠h6-h1 ♠f8-d7 125 ♠h1-b1 ♠d7-b8 126 ♠f5-e5 ♠d5-e3 127 ♠e5-d4 ♠e3-
 f5 128 ♠d4-d5 ♠f5-e3 129 ♠d5-c5 ♠b8-d7 130 ♠c5-d4 ♠e3-g4 131 ♠b1-c1
 ♠c7-d8 132 ♠c1-e1 ♠g4-f6 133 ♠e4-g5 (♠e4-d6) ♠d8-c7 134 ♠g5-f7 ♠d7-f8
 135 ♠e1-f1 ♠f6-g4 136 ♠f1-g1 ♠g4-f6 137 ♠g1-e1 ♠c7-d7 138 ♠d4-e5 ♠f6-e8
 139 ♠f7-h8 ♠d7-e7 140 ♠e5-d5 ♣e7-d7 141 ♠e1-f1 ♠e8-c7 142 ♠d5-e5 ♠f8-
 e6 143 ♠h8-g6 ♠e6-c5 144 ♠f1-b1 ♠d7-c6 145 ♠g6-e7 ♠c6-d7 146 ♠e7-f5
 ♠d7-c6 147 ♠f5-d4 ♠c6-d7 148 ♠b1-d1 ♠c7-a6 149 ♠d4-f5 ♠d7-c6 150 ♠d1-
 h1 ♠a6-b4 151 ♠h1-h6 ♠c6-d7 152 ♠e5-d4 ♠c5-e6 153 ♠d4-c4 ♠b4-a6
 154 ♠h6-h7 ♠d7-c6 155 ♠h7-h1 ♠a6-c7 156 ♠h1-d1 ♠c7-e8 157 ♠f5-e7
 ♠c6-c7 158 ♠c4-d5 ♠e6-f8 159 ♠e7-g8 ♠c7-d7 160 ♠d5-c5 ♠d7-e6 (♠d7-
 c7) 161 ♠d1-e1 ♠e6-d7 162 ♠e1-e7 ♠d7-d8 163 ♠e7-a7 ♠f8-d7 164 ♠c5-c6
 ♠d7-e5 165 ♠c6-d5 ♠e5-g6 166 ♠a7-h7 ♠e8-c7 167 ♠d5-c6 ♠g6-e5 168 ♠c6-
 d6 ♠e5-c4 169 ♠d6-c5 ♠c4-e5 170 ♠h7-h5 ♠e5-f7 171 ♠c5-c6 ♠c7-e6
 172 ♠h5-a5 ♠d8-e8 173 ♠g8-f6 ♠e8-e7 174 ♠f6-d5 ♠e7-f8 175 ♠c6-d7 ♠e6-
 d4 176 ♠d5-f4 ♠f7-h6 177 ♠a5-d5 ♠d4-f5 178 ♠d7-e6 ♠f5-g7 179 ♠e6-f6
 ♠h6-g8 180 ♠f6-e5 ♠g8-h6 181 ♠d5-a5 ♠h6-g4 182 ♠e5-d4 (♠e5-d5) ♠f8-
 f7 183 ♠a5-a7 ♠f7-f6 184 ♠d4-e4 ♠g7-e8 185 ♠a7-a6 ♠f6-g7 186 ♠a6-b6
 (♠f4-g2) ♠g4-f6 187 ♠e4-f5 ♠f6-d7 188 ♠f4-e6 ♠g7-f7 189 ♠e6-g5 ♠f7-f8
 190 ♠b6-a6 ♠e8-g7 191 ♠f5-g6 ♠d7-e5 192 ♠g6-h7 ♠g7-e8 193 ♠a6-e6
 ♠e5-f7 194 ♠g5-f3 ♠f7-d6 195 ♠h7-g6 ♠d6-f5 (♠d6-c8) 196 ♠e6-e1 ♠f5-
 e7 197 ♠g6-g5 ♠f8-f7 198 ♠f3-e5 ♠f7-g7 199 ♠e5-g4 ♠g7-f8 200 ♠g4-h6
 ♠e7-d5 201 ♠h6-f5 ♠f8-f7 202 ♠e1-e2 (♠e1-e4 ♠e1-e5) ♠d5-b6 203 ♠e2-e7
 ♠f7-f8 204 ♠e7-e1 ♠b6-d5 205 ♠e1-e5 ♠d5-b6 (♠e8-c7) 206 ♠g5-g6 ♠e8-
 c7 207 ♠f5-d6 ♠b6-d5 208 ♠e5-e1 ♠c7-e6 (♠d5-f4 ♠d5-e7 ♠d5-b4) 209 ♠g6-f5
 ♠e6-c7 210 ♠f5-e5 ♠d5-b4 (♠f8-e7) 211 ♠e1-f1 ♠f8-e7 212 ♠f1-f7 ♠e7-d8
 213 ♠d6-b7 ♠d8-c8 214 ♠b7-c5 ♠c7-b5 215 ♠f7-g7 (♠f7-h7) ♠c8-d8 216 ♠g7-
 b7 ♠b4-c6 217 ♠e5-e6 ♠d8-c8 218 ♠b7-h7 ♠c6-b4 219 ♠c5-a4 ♠b4-a6
 220 ♠e6-d5 ♠b5-c7 221 ♠d5-d6 ♠c7-e8 222 ♠d6-e7 ♠e8-c7 223 ♠h7-h6
 ♠a6-b8 224 ♠a4-b6 ♠c8-b7 225 ♠b6-c4 ♠b8-c6 226 ♠e7-d6 (♠e7-d7) ♠c6-b4
 227 ♠h6-h8 ♠b4-a6 228 ♠h8-h7 ♠b7-c8 229 ♠c4-a5 ♠c8-d8 230 ♠a5-c6
 ♠d8-c8 231 ♠c6-e7 ♠c8-d8 232 ♠e7-d5 ♠c7-e8 233 ♠d6-c6 ♠a6-b8 234 ♠c6-
 b5 ♠e8-d6 235 ♠b5-c5 ♠d6-c8 236 ♠h7-h8 ♠d8-d7 237 ♠d5-f6 ♠d7-c7
 238 ♠h8-h7 ♠c7-d8 239 ♠h7-b7 ♠b8-a6 240 ♠c5-c6 ♠c8-e7 241 ♠c6-b6

 a6-b4 242  b7-d7

Chapter 7

Group fast Fourier transforms and their parallelization

One of the most powerful and elegant tools for symmetry exploitation is the group Fourier transform. This chapter describes new fast algorithms for parallel group Fourier transforms and presents several new applications to string matching.

The group Fourier transform is a generalization of the classical Fourier transform to the case in which the index set, instead of being \mathbb{Z}_m , is a general finite group, \mathfrak{G} . This notion will be explained in detail later in the chapter, but for now, let us try to understand, in general terms, one kind of relationship between symmetry and the classical Fourier transform.

Suppose we want to compute some (linear) function M on a vector of points \mathbf{v} . Assuming that M is an $n \times n$ matrix, then brute-force multiplication of M by \mathbf{v} will take n^2 operations. But if we have the following additional information on M then we can do better; suppose, for instance, that if we cyclically shift \mathbf{v} to the left, getting a new vector \mathbf{w} , then $M\mathbf{w}$ is the

cyclic shift of $M\mathbf{v}$; these cyclic shifts, of course, form a cyclic group of order n . In that case, we can compute $M\mathbf{v}$ in $O(n \log n)$ time by operating in the frequency domain; that is, by first Fourier transforming both \mathbf{v} and the first column of M and then pointwise multiplying the results. The classical Fourier transform can therefore be thought of as a transform that elicits the underlying symmetry of the operator M *when this symmetry is a cyclic group*.

In general, however, one can imagine transformations on \mathbf{v} far more general than cyclic shifts. When the set of transformations form a group \mathcal{G} commuting with M , then, once again, it is more efficient to operate in the spectral domain. We take the \mathcal{G} -Fourier transform of the first column of M and of \mathbf{v} , and “pointwise” multiply these, although the points, in this case, are in fact matrices. In this way, the \mathcal{G} -Fourier transform can be thought of as the function that exposes the underlying symmetry of M . Just as the *classical* Fourier transform is often used to model time-invariant properties of a signal, so the \mathcal{G} -Fourier transform can be used to model the \mathcal{G} -invariant properties of an input.

The format of this chapter will be as follows.

Section 7.1 provides a brief overview of the fascinating history of the classical Fourier transform, which is simply a group Fourier transform for Abelian groups. Then we will briefly present some of the classical fast Fourier transform (FFT) algorithms in the tensor product notation developed in section 4.3.

Section 7.2 provides a brief review of group representation theory and the basics of group Fourier transform theory. The concept of an irreducible group representation is introduced, and the group Fourier transform is formally defined. Subsection 7.2.2 then redefines the

group Fourier transform in an entirely equivalent way using the concepts of group algebras; this formulation lets us more easily understand the matricial structure of group Fourier transforms.

Section 7.3 describes the *fast* group Fourier transforms algorithms. Although these algorithms are not as fast as classical FFT algorithms, they are still considerably faster than the brute-force n^2 algorithm for many classes of groups. In the case where the group \mathfrak{G} is Abelian, we see that the new algorithms reduce to the Cooley-Tukey FFT algorithm.

Section 7.4 describes the parallelization of the group FFT algorithms. Subsection 7.4.1 reviews parallelization of classical FFT algorithms, and subsection 7.4.2 outlines previous work on parallel group Fourier transforms for general finite groups. Subsection 7.4.3 presents a parallelization of the group fast Fourier transforms, and describes a preliminary parallel \mathfrak{S}_n transform. Subsection 7.4.4 discusses in more detail the application to group circulants. The section concludes in subsection 7.4.5, where applications of group FFTs to learning, random walks on groups, analysis of ranked data, and group filters are presented.

Section 7.5 illustrates the ideas developed so far by describing the development of a simple parallel dihedral group FFT implementation.

Section 7.6 describes a new class of applications for group FFT algorithms: generalized string matching. Many earlier string-matching algorithms are brought under a single unifying rubric of generalized matrix multiplication, and the symmetry of the original problem is modeled by the symmetry of the matrix. These generalized matrix multiplication problems are particularly interesting because, despite their almost complete lack of alge-

braic structure, the group FFT algorithms can still be used. Subsection 7.6.1 presents a brief background of string-matching problems, subsection 7.6.2 describes the mathematical underpinnings of our model for string matching, subsection 7.6.3 gives algorithms for performing certain kinds of generalized matrix multiplication algorithms, and subsection 7.6.4 describes a class of new parallel string-matching algorithms.

Section 7.7 concludes with some open problems and suggestions for future work.

7.1 Classical Fourier transforms

The history of fast Fourier transforms is somewhat convoluted, so to speak, but nevertheless worthy of remark. Three summaries of the history of varying focus are contained in the references by Goldstine [340], by Cooley, Lewis, and Welch [199] and by Heidemann, Johnson and Burrus [361].

Abelian fast Fourier transform theory has its roots in Carl Friedrich Gauss' 19th-century posthumous paper [323]. Gauss considered the problem of interpolating a trigonometric series as part of a general investigation of interpolation techniques. Unfortunately, due to language, notational, and terminological difficulties, it is difficult for the contemporary reader to discern the modern FFT in Gauss' presentation, although it was pointed out by Goldstine [340, p.247–258].

An $O(n \log n)$ FFT was rediscovered by Runge, in the early 1900s [652], and refined into a usable algorithm by Danielson and Lanczos in a famous 1942 paper, which, however, had

been forgotten [218,219]. Danielson and Lanczos were interested in X-ray analysis, and the following excerpt from the introduction to their first paper [218, pp.365–366] gives a vivid an entertaining picture of the computational conditions under which they labored.

One of the more recent applications of Fourier analysis occurs in the quantitative investigation of liquids by x-rays following the theory of Zernike and Prins. Although the following method was developed for this application, it is equally applicable to any problem requiring a Fourier analysis. If a modern mechanical analyzer (e.g. Henrici [362] or Michelson [552]²⁰) is available, the evaluation of a Fourier integral presents no difficulty. It is our purpose to show that, for occasional analysis at least, one need not depend upon such costly instruments, even when the required number of coefficients is very large. Like all other arithmetical methods we make use of the symmetry of the trigonometric functions in the four quadrants of a circle. The great reduction in the number of operations, which this allows, has been pointed out by Runge [652]. Since, however, the labor varies approximately as the square of the number of ordinates, the available standard forms become impractical for a large number of coefficients. We shall show that, by a certain transformation process, it is possible to double the number of ordinates with only slightly more than double the labor.

In the technique of numerical analysis the following improvements suggested by Lanczos were used: (1) a simple matrix scheme for any even number of ordinates can be used in place of available standard forms; (2) a transposition of odd ordinates into even ordinates reduces an analysis for $2n$ coefficients to two analyses for n coefficients; (3) by using intermediate ordinates it is possible to estimate, before calculating any coefficients, the probable accuracy of the analysis; (4) any intermediate value of the Fourier integral can be determined from the calculated coefficients by interpolation. The first two improvements reduce the time spent in calculation and the probability of making errors, the third tests the accuracy of the analysis, and the fourth improvement allows the transform curve to be constructed with arbitrary exactness. Adopting these improvements the approximate times for Fourier analyses are: 10 minutes for 8 coefficients, 25 minutes for 16 coefficients, 60 minutes for 32 coefficients, and 140 minutes for 64 coefficients. . .”

The algorithms of Gauss, Runge, and Danielson-Lanczos were fundamentally the famous

²⁰ The devices of Henrici and Michelson-Stratton to which the authors allude were based on enormous mechanical Fourier analyzer due to Kelvin dating from the 1800s [362].

Cooley-Tukey algorithm, and should be distinguished from the 1958 prime-factor algorithm of Good and the (independently) 1963 work of Thomas, which reduced the case where n was a product of distinct primes to the FFTs of the prime factors of n [343, 749].

In any case, the effect of the paper of Cooley-Tukey, backed up by their implementation, was significant. Until then most FFT implementations used quadratic algorithms, and considerable CPU time was used computing FFT's. The discovery and simple exposition of their $O(n \log n)$ algorithm had a considerable impact on the development of digital signal processing and Fourier analysis techniques.

The modern formulation of the discrete Fourier transform (DFT) is given an input vector of complex numbers $(\mathbf{v}_0, \dots, \mathbf{v}_{n-1})^T$, compute the product $\mathbf{F}_n \mathbf{v}$, where \mathbf{F}_n is the $n \times n$ DFT-matrix whose rs th component is ω^{rs} , where ω is a primitive n th root of unity (for example, $e^{2\pi i/n}$). Our presentation in the remainder of this section will closely follow the exposition in [759, pp.16–20].

Let $n = ml$. The Singleton (1967) [689] mixed-radix version of the Cooley-Tukey (1965) fast Fourier transform [200] can be expressed recursively,

$$\mathbf{F}_n = (\mathbf{F}_m \otimes \mathbf{I}_l) \mathbf{T}_l (\mathbf{I}_m \otimes \mathbf{F}_l) \mathbf{P}_m^n. \quad (7.1)$$

where \mathbf{T}_l is a diagonal matrix encoding the twiddle factors:

$$\mathbf{T}_l = \bigoplus_{j=0}^{m-1} \left(\text{diag} \left(1, \omega, \dots, \omega^{l-1} \right) \right)^j.$$

This can be interpreted as a mixed parallel/vector algorithm (see section 4.3). Given an input vector \mathbf{v} , $\mathbf{P}_m^n \mathbf{v}$ forms a list of m segments, each of length l . The $\mathbf{I}_m \otimes \mathbf{F}_l$ term performs

m l -point FFTs in parallel on each segment. T_l just multiplies each element by a twiddle factor. Finally, the $F_m \otimes I_l$ term performs an m -point FFT on vectors of size l .

The commutation theorem can be used to derive a parallel form

$$F_n = P_m^n (I_l \otimes F_m) P_l^n T_l (I_m \otimes F_l) P_m^n, \quad (7.2)$$

and a vector form

$$F_n = (F_m \otimes I_l) T_l P_m^n (F_l \otimes I_m). \quad (7.3)$$

In 1968, Marshall C. Pease developed an FFT that could be derived by unrolling the recursion in Equation 7.2 [598].

The vectorized Korn-Lambiotte FFT (1979) can be derived by unrolling Equation 7.3 [459, 598].

By using the commutation theorem and varying the factorization, many different FFT algorithms have been derived, with different tradeoffs between parallelization and vectorization [66, 69, 170, 344, 407].

The Fourier transform diagonalizes circulant matrices: given an $n \times n$ circulant matrix \mathbf{M} it can be shown that $F_n \mathbf{M} F_n^{-1}$ is diagonal. Therefore, computation of $\mathbf{M} \mathbf{v}$ reduces to performing a backward and forward Fourier transform; that is, Fourier transforms implement circular convolution [719].

7.2 Group Fourier transforms: Foundations

This section briefly introduces some of the mathematical underpinnings of the group Fourier transform theory. For a more leisurely introduction, we recommend the survey [189]. An introduction from an engineering perspective is in [771]. Clifford theory and the theory of induced representations is clearly introduced in [195]. The exposition here follows the approach of [240].

7.2.1 Basic definitions

A *representation* of a finite group \mathfrak{G} is a group-homomorphism ρ from \mathfrak{G} into the group of invertible linear transformations of an n -dimensional complex vector-space \mathfrak{V}_n (also see subsection 3.2). This latter group, GL_n , may be identified with the multiplicative group of nonsingular $n \times n$ matrices over the complex numbers. If \mathfrak{V} has a subspace \mathfrak{W} such that $\rho(\mathfrak{g})(\mathfrak{W}) = \mathfrak{W}$ for all group elements \mathfrak{g} , then \mathfrak{W} is said to be *invariant* under ρ . A representation whose only invariant subspaces are \mathfrak{V} and $\mathbf{0}$ is said to be *irreducible*.

Suppose ρ and ρ' are two representations of degrees n and n' associated with vector-spaces \mathfrak{V}_n and $\mathfrak{V}_{n'}$. Then we say that ρ and ρ' are *equivalent* if there is an isomorphism $f : \mathfrak{V}_n \rightarrow \mathfrak{V}_{n'}$ such that $f \circ \rho(\mathfrak{g}) = \rho'(\mathfrak{g}) \circ f$ for all $\mathfrak{g} \in \mathfrak{G}$. The *direct sum* of ρ and ρ' is the representation of degree $n + n'$ with associated vector-space $\mathfrak{V}_n \oplus \mathfrak{V}_{n'}$ such that $(\rho \oplus \rho')(\mathfrak{g})(\mathbf{v} \oplus \mathbf{v}') = \rho(\mathfrak{g})(\mathbf{v}) \oplus \rho'(\mathfrak{g})(\mathbf{v}')$.

It is a fact that any representation ρ is equivalent to the direct sum of irreducible represen-

tations. There are only a finite number (up to isomorphism) of irreducible representations of any finite group \mathfrak{G} . We write d_ρ for the degree of the representation ρ , and we let \mathcal{R} be a complete set of inequivalent irreducible representations over \mathfrak{G} . The following relation is fundamental:

$$\sum_{\rho \in \mathcal{R}} d_\rho^2 = |\mathfrak{G}|. \quad (7.4)$$

Most representation-theoretic questions over finite groups thereby reduce to the study of irreducible representations.

Any representation of \mathfrak{G} is equivalent to a direct sum of irreducible representations of \mathfrak{G} ; the representation theory of finite groups is thereby normally reduced to the study of irreducible representations.

Let f be any function from \mathfrak{G} into \mathbb{C} . Then the *Fourier transform* of f is the function \hat{f} that assigns to the degree d_ρ representation ρ the $d_\rho \times d_\rho$ matrix:

$$\hat{f}(\rho) = \sum_{\mathfrak{g} \in \mathfrak{G}} f(\mathfrak{g})\rho(\mathfrak{g}).$$

The \mathfrak{G} -*Fourier transform problem* is to compute $\hat{f}(\rho)$ for a complete set of irreducible representations \mathcal{R} .

Suppose that f and f' are two functions from \mathfrak{G} into \mathbb{C} . Their \mathfrak{G} -*convolution* (or just convolution), $f \star_{\mathfrak{G}} f'$, is the function from \mathfrak{G} into \mathbb{C} defined by

$$f \star_{\mathfrak{G}} f'(\mathfrak{g}) = \sum_{\mathfrak{h}\mathfrak{h}'=\mathfrak{g}} f(\mathfrak{h})f'(\mathfrak{h}').$$

Note that, when \mathfrak{G} is a cyclic group of order n , the group convolution reduces to the familiar

circular convolution [11]

$$f \star_{\mathfrak{C}_n} f'(i) = \sum_{j=0}^{n-1} f(j)f'(i-j).$$

Recall that one of the fundamental properties of the classical Fourier transform is that it turns convolution into a pointwise product. The \mathfrak{G} -Fourier representation has a similar function, but applied to \mathfrak{G} -convolution.

There are two important properties of the \mathfrak{G} -Fourier transform.

First, the \mathfrak{G} -Fourier transform is invertible:

$$f(\mathfrak{g}) = \frac{1}{|\mathfrak{G}|} \sum_{\rho \in \mathcal{R}} \text{trace} \left(\hat{f}(\rho) \rho \left(\mathfrak{g}^{-1} \right) \right).$$

Second, the \mathfrak{G} -Fourier transform turns convolution into element-wise product, where the elements multiplied are $d_\rho \times d_\rho$ matrices. Thus, to compute $f \star_{\mathfrak{G}} f'$, we compute the \mathfrak{G} -Fourier transforms \hat{f}, \hat{f}' , multiply corresponding pairs of matrices, and inverse \mathfrak{G} -Fourier transform the result. Assuming that multiplication of a $k \times k$ matrix takes time $O(k^3)$, the total time for the convolution is therefore $O\left(\sum_{\rho \in \mathcal{R}} d_\rho^3\right)$ plus the time to compute two \mathfrak{G} -FFTs and one inverse \mathfrak{G} -FFT. Note that the sum is bounded above by $|\mathfrak{G}|^{\frac{3}{2}}$, insofar as the sum of the squares of the degrees of the irreducible representations is $|\mathfrak{G}|$.

7.2.2 An algebra viewpoint

This subsection presents the \mathfrak{G} -Fourier transform in an alternative but equivalent framework.

The space of functions from \mathfrak{G} into \mathbb{C} forms an *algebra*, $\mathbb{C}[\mathfrak{G}]$, under element-wise addition and convolution product. An algebra is a vector space \mathfrak{A} over \mathbb{C} on which a binary, associative multiplication with unit is defined that distributes over $+$ and satisfies $(\alpha \mathfrak{a})\mathfrak{b} = \alpha(\mathfrak{a}\mathfrak{b})$ for $\alpha \in \mathbb{C}$ and $\mathfrak{a}, \mathfrak{b} \in \mathfrak{A}$ [253, p. 1]. For example, the set of $d \times d$ complex matrices forms a matrix algebra in which the product is simply the usual matrix product. The *dimension* of a matrix algebra is its dimension as a vector space over \mathbb{C} , and is thus the square of the number of rows or columns in its matrices.

The *direct sum* of two matrix algebras \mathfrak{A} and \mathfrak{A}' is the algebra whose underlying vector space is $\mathfrak{A} \oplus \mathfrak{A}'$ and for which multiplication is defined component-wise. It is a fundamental theorem of representation theory that the space of \mathfrak{G} -functions under convolution is algebra-isomorphic to a direct sum of matrix algebras [166, 557, 558].²¹ The dimensions of the components of the direct sum are $\{d_i^2\}_{i=0}^k$, where d_i ranges over the degrees of the irreducible representations of \mathfrak{G} and k is the number of distinct irreducible representations of \mathfrak{G} . The \mathfrak{G} -Fourier transform can also be viewed as the matrix of the isomorphism between the two vector spaces of dimension $|\mathfrak{G}|$. The complexity of computing the \mathfrak{G} -Fourier transform of a function $f \in \mathbb{C}[\mathfrak{G}]$ is called the *complexity*, $T_{\mathfrak{G}}$, of \mathfrak{G} . We should be more careful defining the model to which $T_{\mathfrak{G}}$ refers. The most common model used is the linear complexity, which is the minimal time-complexity of any straight-line arithmetic program over \mathbb{C} that computes the Fourier transform, although sometimes we also bound the size of the numbers that can be manipulated by the straight-line program. Since this is only an informal presentation,

²¹ One of the founders of the theory of algebras and representation theory was Theodor Molien, whose contributions are discussed in subsection 6.1.3.

we will be a bit careless, although we remark that obviously $T_{\mathfrak{G}} = O(|\mathfrak{G}|^2)$, insofar as this is an upper bound on the complexity of multiplication by a $|\mathfrak{G}| \times |\mathfrak{G}|$ matrix.

A Fourier transform and its inverse together give an algorithm for computing group convolution: given f, f' , compute their group Fourier transform, getting two elements of the associated $|\mathfrak{G}|$ dimensional matrix algebra; then multiply these vectors component-wise (where the components are block matrices of dimensions $d_i \times d_i$), and then take the inverse Fourier transform of the product. The complexity of the multiplication step is bounded above by $\sum_i d_i^3$, where, as usual, the d_i range over the degrees of a complete set of irreducible representations of \mathfrak{G} . It can be shown the complexity of the inverse Fourier transform is in fact within a small constant factor of the complexity of the Fourier transform, so that fast group Fourier transforms yield fast convolutions [189]. The exact time complexity of the convolution depends on the degrees of the irreducible representations of the group, although it is clearly bounded above by $O(|\mathfrak{G}|\sqrt{|\mathfrak{G}|})$ when \mathfrak{G} admits a fast Fourier transform, since the sum of the squares of the d_i is $|\mathfrak{G}|$.

When \mathfrak{G} is cyclic, then the \mathfrak{G} -Fourier transform is simply the standard Fourier transform matrix of order n . To see this, observe that a one-dimensional representation is obviously irreducible, and therefore the representation that sends $j \in \mathbb{Z}_n$ to the 1×1 matrix (ω^{kj}) is an irreducible one-dimensional representation of \mathbb{Z}_n for $k = 0, \dots, n - 1$; these form a complete set of one-dimensional representations.

7.3 Fast group Fourier transform algorithms: Background

The first person to consider non-Abelian group Fourier transforms seems to have been Karpovsky (1977) who considered the case of fast \mathfrak{G} -Fourier transforms when \mathfrak{G} was the direct product of subgroups each of which had a fast FFT [428]. Similar results were considered by Atkinson (1977) [65].

However, the first nontrivial \mathfrak{G} -Fourier transform algorithms are due to Beth, who gave $O(|\mathfrak{G}|^{3/2})$ time algorithms for the case when \mathfrak{G} is solvable [119, 120]. Beth also described several potential applications of these techniques [118].

When \mathfrak{G} is “close” to being Abelian, then, not surprisingly, fast \mathfrak{G} -Fourier transforms also exist. Clausen gave $O(|\mathfrak{G}|\text{polylog}|\mathfrak{G}|)$ time algorithms for the case when \mathfrak{G} is metabelian (a *metabelian* group is one that has Abelian normal subgroup $\mathfrak{H} \subset \mathfrak{G}$ such that $\mathfrak{G}/\mathfrak{H}$ is also Abelian; this class includes the group of symmetries of a k -gon, generalized quaternion groups, and groups for which there is a prime p such that $|\mathfrak{G}|$ is a power of p and smaller than p^6 , as are groups whose order is a product of distinct primes) [186]. This work was independently performed and also generalized by Rockmore [630], who showed that if \mathfrak{H} were normal in \mathfrak{G} , and $\mathfrak{G}/\mathfrak{H}$ was Abelian, then

$$T_{\mathfrak{G}} = O\left(\frac{|\mathfrak{G}|}{|\mathfrak{H}|} \cdot T_{\mathfrak{H}} + |\mathfrak{G}| \log\left(\frac{|\mathfrak{G}|}{|\mathfrak{H}|}\right)\right).$$

The most important specific class of \mathfrak{G} considered are the symmetric groups \mathfrak{S}_n . Fourier transforms for \mathfrak{S}_n have been shown by Persi Diaconis to have significance for the analysis of ranked data [239] [238, pp.141–160]; Diaconis’ work has been an important motivation for

the field, in fact. Any finite group \mathfrak{G} is a subgroup of some \mathfrak{S}_n , and some important groups, such as the alternating groups, are such big subgroups that a fast \mathfrak{S}_n -Fourier transform automatically yields a fast group transform for that group. Rockmore seems to have been the first to give a $T_{\mathfrak{S}_n} = O(|\mathfrak{S}_n| \text{polylog } |\mathfrak{S}_n|)$ algorithm for the problem [628], and refinements continue to be discovered [187, 190, 631]. In practice, \mathfrak{S}_{10} transforms require about 20 minutes on a Sparc 1 [187].

When \mathfrak{G} is a wreath product by a symmetric group, efficient transform algorithms have been given by Rockmore [632]. Recently, Maslen and Rockmore have applied the concept of adapted group diameter to the generation of efficient group Fourier transforms for a wide range of groups, including the groups above, in addition to general linear groups over a finite field [537].

Fast \mathfrak{G} -Fourier transform algorithms (\mathfrak{G} -FFTs) all use the same general structure. Suppose we want to compute the \mathfrak{G} -FFT of a function $f: \mathfrak{G} \rightarrow \mathbb{C}$. First, a subgroup $\mathfrak{H} \subset \mathfrak{G}$ is chosen. Next, several $|\mathfrak{G}|$ -FFTs of functions $f_1, \dots, f_k: \mathfrak{H} \rightarrow \mathbb{C}$ are computed, possibly recursively, for some functions f_1, \dots, f_k . Finally, the \mathfrak{G} -FFT of f is computed from the \mathfrak{H} -FFTs.

Mathematically, suppose that \mathfrak{H} has index k in \mathfrak{G} , and let X be a set of coset representatives for \mathfrak{H} in \mathfrak{G} . Let $\rho \downarrow \mathfrak{H}$ denote the restriction of ρ to \mathfrak{H} ; note that this is in turn a representation for \mathfrak{H} . Let $f_{\mathfrak{g}}: \mathfrak{H} \rightarrow \mathbb{C}$ be defined by $\mathfrak{h} \mapsto f(\mathfrak{g}\mathfrak{h})$. We have:

$$\hat{f}(\rho) = \sum_{\mathfrak{g} \in \mathfrak{G}} f(\mathfrak{g})\rho(\mathfrak{g}) \tag{7.5}$$

$$= \sum_{x \in X} \sum_{\mathfrak{h} \in \mathfrak{H}} f(x\mathfrak{h})\rho(x\mathfrak{h}) \tag{7.6}$$

$$= \sum_{\mathfrak{x} \in X} \rho(\mathfrak{x}) \sum_{\mathfrak{h} \in \mathfrak{H}} f(\mathfrak{x}\mathfrak{h})\rho(\mathfrak{h}) \quad (7.7)$$

$$= \sum_{\mathfrak{x} \in X} \rho(\mathfrak{x}) \sum_{\mathfrak{h} \in \mathfrak{H}} f_{\mathfrak{x}}(\mathfrak{h})\rho(\mathfrak{h}) \quad (7.8)$$

$$= \sum_{\mathfrak{x} \in X} \rho(\mathfrak{x}) \hat{f}_{\mathfrak{x}}(\rho \downarrow \mathfrak{H}). \quad (7.9)$$

It can be shown that the complete set of irreducible representations \mathcal{R} of \mathfrak{G} can always be chosen so that any representation $\rho \in \mathcal{R}$ is equal to a direct sum of irreducible representations of \mathfrak{H} , (not merely equivalent). Such a set of representations is called *adapted* to \mathfrak{H} .

Assuming \mathcal{R} is adapted, we have thereby reduced the computation of $\hat{f}(\rho)$ to the computation $\{\hat{f}(\eta)\}$, for a complete set of irreducible representations of \mathfrak{H} . Of course, this process can be iterated through a chain of subgroups.

Now, when $\mathfrak{G} = \mathfrak{C}_{2^k}$ then choosing the natural length- k chain of subgroups recovers the Cooley-Tukey FFT. If \mathfrak{G} is Abelian, then the \mathfrak{G} -Fourier transform can be computed from the structure theorem for Abelian groups. It can be shown that the $\mathfrak{G}_1 \times \mathfrak{G}_2$ -Fourier transform is the tensor product of the Fourier transforms for \mathfrak{G}_1 and \mathfrak{G}_2 . Since every Abelian group is the direct product of cyclic groups, whose Fourier transforms were computed above, the Fourier transform of an Abelian group is simply the tensor product of a number of copies of ordinary classical Fourier transforms, possibly with some permutations of the data. It is easy to see that the classical results now show that $T_{\mathfrak{G}} = O(|\mathfrak{G}| \log |\mathfrak{G}|)$ for Abelian \mathfrak{G} .

7.4 Parallel group Fourier transforms

7.4.1 Abelian case

The problem of parallelization of the classical Fourier transform (which is equivalent, we have seen, to the problem of parallelizing the \mathfrak{G} -Fourier transform for Abelian \mathfrak{G}) has been an active and important area of research since the seminal paper by Marshall Pease [598]. The Pease FFT might arguably be characterized as an early motivation for the construction of high-speed parallel computers, as his 1968 paper on the topic argued for the feasibility of highly-parallel FFT implementations, concluding “it would be possible to build a special purpose parallel computer to calculate the Fourier transforms of large sets of data at extremely high speed. . . [598].”

Unlike some classes of parallel algorithms, FFTs have been implemented on many physical machines and have been shown to attain high bandwidth in practice, and they are used in real applications. Indeed, the FFT is clearly parallelizable in a theoretical sense using $O(n)$ processors and $O(\log n)$ time, so most of the considerable body of literature on parallel FFT concentrates on reducing the constant factors in implementations. An excellent case-study of the CM-2 parallelization is by Johnsson, Krawitz, Frye, and Macdonald, who describe some of the aggressive coding tricks and data-reorganization permitting high-bandwidth FFT on the CM-2 [419]; see also [4, 421] for other CM-2 implementations. Swarztrauber provides a summary of some algorithms, and Johnsson, Jacquemin, and Ho describe a high-radix FFT [7, 417, 733]. Scalability issues are discussed by Gupta and Kumar [353].

Averbuch, Gabber, Gordisky, and Medan discuss the MIMD case, and Munthe-Kaas also discusses the problem [69,570]. The vector case has been treated by, for example, Korn and Lambiotte [459] and Schwarztrauber [732]. Mou and Wang (1993) provide a fairly recent analysis from a communication-theoretic point of view [567].

7.4.2 General case: Background

Much less attention has been devoted to the problem of parallelizing \mathfrak{G} -Fourier transforms for general finite \mathfrak{G} .

In the most general formulation, one could imagine an algorithm that, given an arbitrary finite group \mathfrak{G} , presented in terms of a generating set, for example, computes the \mathfrak{G} -Fourier transform matrix for \mathfrak{G} , as well as a fast algorithm for its application. This problem might arise in a situation in which the symmetry of the problem is not known a priori, but must be computed on-line. For most applications that have arisen to date, the structure of \mathfrak{G} is known off-line, and has not been considered even sequentially. In other cases, the irreducible representations of \mathfrak{G} would need to be computed [71,73].

Instead, we consider the case where \mathfrak{G} is fixed and known ahead of time, and thus we do not consider the time required to analyze the structure of \mathfrak{G} and to figure out a good \mathfrak{G} -Fourier transform. We know of only a few previous treatments of this problem.

Roziner, Karpovsky, and Trachtenberg considered the case in which \mathfrak{G} was the direct product

of subgroups [650]. This case is quite trivial, however, because, it is easy to see that

$$\mathbf{F}_{\mathfrak{G}_1 \times \mathfrak{G}_2} = \mathbf{F}_{\mathfrak{G}_1} \otimes \mathbf{F}_{\mathfrak{G}_2}, \quad (7.10)$$

therefore, the tensor-product parallelization techniques of section 4.3 apply. Note that equation 7.10, when iterated, recovers the fast algorithm for the Walsh transform, which is $\mathbf{F}_{\mathfrak{G}_2^k}$. Roziner et al. give numerical examples when one of the factors is the quaternion group, and confirm that very fast execution is possible; faster, in fact, than for FFTs of Abelian groups of the same size [650].

Clausen and Gollmann have considered the case of the VLSI implementations for the symmetric group, using ideas similar to the implementation we give below [191].

Diaconis and Rockmore sketch a parallel algorithm in a subsection of their paper [240, pp.326–328]. However, their algorithm has quadratic worst-case work complexity.

7.4.3 A parallel algorithm for general groups

Our presentation will closely follow the sequential presentation of Maslen and Rockmore (1995) [537].

Observe that multiplication by $\mathbf{F}_{\mathfrak{G}}$ can be performed in logarithmic time using $O(|\mathfrak{G}|^2)$ processors, as it consists of multiplication of a $|\mathfrak{G}| \times |\mathfrak{G}|$ -matrix by a vector of length $|\mathfrak{G}|$. In fact, however, all of the known fast Fourier transform algorithms are parallelizable using standard techniques from parallel processing because they can be viewed as factorizations of $\mathbf{F}_{\mathfrak{G}}$ in terms of simple sparse matrices.

In order to analyze parallelizability, we use the linear circuit model of computation. A linear circuit is a weighted directed acyclic graph with three types of nodes: input, output, and interior. Each input node has in-degree 0, each output node has out-degree 0. Each node v in a linear circuit is associated with a linear form $L(v)$ in its inputs as follows. The linear form associated to input node x is x . Let E be the set of edges, and $w(e) \in \mathbb{C}$ be the weight associated with edge $e \in E$. The linear form associated to a node v is

$$\sum_{\{u:(u,v) \in E\}} w(e)L(u).$$

Given a circuit G , its *size* $|G|$ is number of edges in G , and its *depth* $D(G)$ is the length of the longest path in G . Clearly, given a circuit G of size s and depth d and given scalars x_1, \dots, x_n , the value of the linear form corresponding to each output node of the circuit can be computed on a PRAM with s processors in time $O(d \log s)$.

Let $\mathfrak{H} < \mathfrak{G}$ and let X be a set of coset representatives for \mathfrak{H} in \mathfrak{G} . We let $M_{\mathfrak{G}}(X)$ be a circuit that computes $\sum_{\mathfrak{x} \in X} \rho(\mathfrak{x})F_{\mathfrak{x}}(\rho)$ for inputs being $d_{\rho} \times d_{\rho}$ matrices $F_{\mathfrak{g}}(\rho)$ for each $\rho \in \mathcal{R}$ and $\mathfrak{x} \in X$. Thus, this circuit has $|\mathfrak{G}||X|$ inputs and $|\mathfrak{G}|$ outputs.

Theorem 2 *Let $\mathfrak{H} < \mathfrak{G}$, let \mathcal{R} be a complete set of irreducible representations for \mathfrak{G} , let X be a set of coset representatives for \mathfrak{H} in \mathfrak{G} , and let $C_{\mathfrak{H}}$ compute $F_{\mathfrak{H}}$ for a complete set of irreducible representations for \mathfrak{H} to which \mathcal{R} is adapted. Then there is a circuit $C_{\mathfrak{G}}$ computing $F_{\mathfrak{G}}$ that satisfies*

$$|C_{\mathfrak{G}}| \leq 2|\mathfrak{G}| + \frac{|\mathfrak{G}|}{|\mathfrak{H}|}|C_{\mathfrak{H}}| + |M_{\mathfrak{G}}(X)|, \quad (7.11)$$

and

$$D(C_{\mathfrak{G}}) \leq 2 + D(C_{\mathfrak{H}}) + D(M_{\mathfrak{G}}(X)). \quad (7.12)$$

Proof: This follows easily from equation 7.9.

Iterating this construct, we see that, (cf. [537, Theorem 2.2])

Theorem 3 *Let $\mathfrak{G}_0 < \mathfrak{G}_1 < \dots < \mathfrak{G}_k = \mathfrak{G}$ be a subgroup chain in \mathfrak{G} , and let X_i , $i = 1, \dots, k$ be a set of coset representatives for \mathfrak{G}_{i-1} in \mathfrak{G}_i . Then there is a circuit $C_{\mathfrak{G}}$ satisfying*

$$|C_{\mathfrak{G}}| \leq |\mathfrak{G}| \left(2k + \frac{|C_{\mathfrak{G}_0}|}{|\mathfrak{G}_0|} + \sum_{i=1}^k \frac{|M_{\mathfrak{G}_i}(X_i)|}{|\mathfrak{G}_i|} \right)$$

and

$$D(C_{\mathfrak{G}}) \leq 2k + \left(D(C_{\mathfrak{G}_0}) + \sum_{i=1}^k D(M_{\mathfrak{G}_i}(X_i)) \right).$$

Proof: Unroll the recursion in Theorem 2.

The method we have illustrated here for translating statements about the sequential time complexity of \mathfrak{G} -FFTs into statements about their parallel complexity is easy to apply to most of the sequential algorithms. This is because these algorithms rely on a chain of subgroups, so that their depth of recursion is normally at most logarithmic.

We now outline the separation of variables technique [537] and show how it yields parallelizable sub-quadratic algorithms for many classes of groups.

The key to further speedups in a \mathfrak{G} -FFT is to speed up the computation $M_{\mathfrak{G}_i}(X_i)$, and this depends upon the matrices $\{\rho(\mathfrak{x}_i); \mathfrak{x}_i \in X_i\}$ having some special form. This special form is given by the following generalization of Schur's Lemma.

Theorem 4 *Schur's Lemma: Let $\mathfrak{h} < \mathfrak{G}$ and suppose that ρ is a representation for \mathfrak{G} such that $\rho \downarrow \mathfrak{h}$ is the direct sum*

$$\eta_1 \oplus \eta_1 \oplus \cdots \oplus \eta_r \oplus \cdots \oplus \eta_r,$$

where the multiplicity of η_i is m_i . Suppose that \mathbf{A} is a $d_\rho \times d_\rho$ matrix commuting with the matrices $\rho(\mathfrak{h})$, for each $\mathfrak{h} \in \mathfrak{h}$. Then there are $m_i \times m_i$ matrices \mathbf{B}_i such that

$$\mathbf{A} = \bigoplus_{i=1}^r \mathbf{B}_i \otimes \mathbf{I}_{d_{\eta_i}}.$$

Proof: This can be shown by modifying the usual proof of Schur's Lemma.

Schur's Lemma gives a parallel analogue of a result of Maslen and Rockmore [536, Corollary 4.4]:

Theorem 5 *Let $\mathfrak{K} \leq \mathfrak{h}$ be subgroups of \mathfrak{G} and let \mathcal{R} be a complete set of irreducible representations of \mathfrak{G} adapted to \mathfrak{h} and \mathfrak{K} . For each $\rho \in \mathcal{R}$ let $F(\rho)$ be a $d_\rho \times d_\rho$ matrix. Let $\mathfrak{h} \in \mathfrak{h}$ commute with each element of \mathfrak{K} . Then the set*

$$\{\rho(\mathfrak{h}) \cdot F(\rho) : \rho \in \mathcal{R}\}$$

can be computed by a linear circuit of size $|\mathfrak{G}| \mathcal{M}(\mathfrak{h}, \mathfrak{K})$ and depth $O(\log |\mathfrak{G}|)$, where $\mathcal{M}(\mathfrak{h}, \mathfrak{K})$ is the maximum multiplicity of an irreducible representation of \mathfrak{K} in the restriction to \mathfrak{K} of a representation in \mathcal{R} .

Proof: Each row of $\rho(\mathfrak{h})$ contains at most $\mathcal{M}(\mathfrak{h}, \mathfrak{K})$ nonzero entries by Schur's Lemma, since $\rho(\mathfrak{h})$ commutes with each $\rho(\mathfrak{k})$, for $\mathfrak{k} \in \mathfrak{K}$.

Now it follows from equation 7.9 that if we assume the existence of a fast FFT for a subgroup \mathfrak{H} of \mathfrak{G} then the main computational obstacle to a rapid evaluation of a \mathfrak{G} -FFT is multiplication by the matrices $\rho(\mathfrak{x})$, as \mathfrak{x} ranges over a complete set of coset representatives for \mathfrak{H} in \mathfrak{G} and ρ ranges over irreducible representations. The complexity of this key subcomputation is expressed by $M_{\mathfrak{G}}(X)$.

The remainder of this section describes ways in which the complexity of this computation may be bounded. It is certainly the case that if $\rho(\mathfrak{x})$ were of a particularly simple form—for example, diagonal, block-diagonal, or sparse—then of course the complexity of matrix multiplication by $\rho(\mathfrak{x})$ could be reduced from the brute-force. In general, however, the $\rho(\mathfrak{x})$ are not “nice” in this sense.

Our strategy, therefore, is to express each $\rho(\mathfrak{x})$ as a product of “nice” matrices. The form of each of these “nice” factors is given by Schur’s Lemma: it is a direct sum of tensor products. In other words, we have a collection of nice matrices (given by Schur’s Lemma), and we want to express our $\rho(\mathfrak{x})$ as products of these matrices. This problem is similar to the problem in permutation group theory in which a given group element is to be expressed as a product of generators, and indeed permutation group terminology will provide a useful notation for our results (once again we are following the sequential development of Maslen and Rockmore [537]).

A *strong generating set* with respect to a chain $\{\mathfrak{e}\} = \mathfrak{G}_0 < \cdots < \mathfrak{G}_n = \mathfrak{G}$ is a set S such that $S \cap \mathfrak{G}_i$ generates \mathfrak{G}_i for all i [72,684]. Our strategy will be to find a strong generating set for \mathfrak{G} whose representations have a nice form, and then to express the $\rho(\mathfrak{x})$ as a product

of these.

Let γ_i be the minimum length such that the set of products of $\leq \gamma_i$ elements from $S \cap \mathfrak{G}_i$ generates coset representatives for \mathfrak{G}_{i-1} in \mathfrak{G}_i . The *adapted diameter* $\gamma\{S, \{\mathfrak{G}_i\}_{i=0}^n\}$ of the chain $\{\mathfrak{G}_i\}$ relative to the strong generating set S is $\sum_{i=1}^n \gamma_i$. Let $\mathfrak{G}_{\mathfrak{g}}$ be the largest subgroup in the chain containing \mathfrak{g} , and $\mathfrak{G}'_{\mathfrak{g}}$ be the largest subgroup in the chain that commutes with \mathfrak{g} . Let

$$\mathcal{M}(S) = \max \left\{ \mathcal{M}(\mathfrak{G}_{\mathfrak{g}}, \mathfrak{G}'_{\mathfrak{g}}) : \mathfrak{g} \in S \right\}.$$

Intuitively, $\mathcal{M}(S)$ is a measure of the “badness” (from the point of view of Schur’s Lemma) of the representations of the generators in S , and $\gamma\{S, \{\mathfrak{G}_i\}_{i=0}^n\}$ is a measure of how many times this “badness” occurs in the expression for an arbitrary element of \mathfrak{G} . In order to get fast parallel \mathfrak{G} -FFTs, we will try to choose strong generating sets with “nice” generators for such that elements of \mathfrak{G} can be expressed in a reasonably small number of such generators.

Formally, the following parallel version of a sequential result now follows [537, Corollary 3.3]:

Theorem 6 *Let S be a strong generating set for \mathfrak{G} relative to the chain $\{\epsilon\} = \mathfrak{G}_0 < \dots < \mathfrak{G}_n = \mathfrak{G}$. Then there is a circuit computing $F_{\mathfrak{G}}$ in size bounded above by*

$$\kappa|\mathfrak{G}|\gamma(S, \{\mathfrak{G}_i\}_{i=0}^n)\mathcal{M}(S)$$

and depth bounded above by

$$O(\gamma(S, \{\mathfrak{G}_i\}_{i=0}^n) \text{polylog}|\mathfrak{G}|),$$

where κ is the maximum of the indices of \mathfrak{G}_{i-1} in \mathfrak{G}_i .

Proof: This follows from Theorem 5, the definition of adapted diameter, and Schur's Lemma.

We now describe strong generating sets for several classes of groups and obtain efficient parallel algorithms on application of Theorem 6 [537].

If \mathfrak{G} is Abelian, take $S = \mathfrak{G}$ and fix any chain of subgroup $\{\mathfrak{e}\} = \mathfrak{G}_0 < \dots < \mathfrak{G}_n = \mathfrak{G}$. Then $\gamma(S, \{\mathfrak{G}_i\}_{i=0}^n) \mathcal{M}(S) = n$. Since n may be bounded by $O(\log |\mathfrak{G}|)$ Theorem 5 recovers the parallelization of the Cooley-Tukey FFT.

When $\mathfrak{G} = \mathfrak{S}_n$, a symmetric group, then we can use the subgroup chain $\{\mathfrak{e}\} = \mathfrak{S}_1 < \mathfrak{S}_2 < \dots < \mathfrak{S}_n$, where \mathfrak{S}_k is identified with the stabilizer of $\{k+1, \dots, n\}$. By choosing the strong generating set S to be the transpositions $\{(i \ i+1)\}$, and using Young's seminormal form for the representations, it follows from the representation theory of the symmetric group that $\mathcal{M}(S) = 2$ and the adapted diameter is polylogarithmic; the existence of an $\text{polylog}|\mathfrak{G}|$ time \mathfrak{S}_n -FFT algorithm using $|\mathfrak{G}|$ processors for symmetric groups now follows [399, 659].²²

²² A very preliminary version of the forward \mathfrak{S}_n Fourier transform has been implemented in CM Fortran on a CM-5. The Young seminormal forms are computed using the serial C code of Baum and Clausen, which they graciously provided to the author [190]. Steven Skiena's Mathematica package `Combinatorica.m` was also used in some of the Young tableaux computations [690, Chapter 2]. This preliminary version requires 86 seconds to compute an \mathfrak{S}_{11} -FFT using a 512-node partition of a CM-5. This timing is about 14 times faster than the previously reported serial time for an \mathfrak{S}_{10} FFT, which is 11 times smaller. Our \mathfrak{S}_{10} code, however, is only about 80 times faster than the serial

When \mathfrak{G} is $\text{GL}_n(q)$, the group of nonsingular matrices over a finite field of order q , then it can be shown that there exists a strong generating set S with polylogarithmic adapted diameter and $\mathcal{M}(S) = |\mathfrak{G}|^{O(1)/n}$ [167, 537].

This technique extends to parallelizing other published \mathfrak{G} -FFT algorithms, most of which rely on applications of Clifford theory, which predicts the representations of \mathfrak{G} in terms of the representations of a normal subgroup \mathfrak{N} . We briefly consider the case of a solvable group \mathfrak{G} , which is a group for which there is a chain $\{\epsilon\} = \mathfrak{G}_0 < \mathfrak{G}_1 < \dots < \mathfrak{G}_n = \mathfrak{G}$ where each \mathfrak{G}_i is normal in \mathfrak{G}_{i+1} and the factor groups $\mathfrak{G}_{i+1}/\mathfrak{G}_i$ are cyclic of prime order. A factor group $\mathfrak{G}_{i+1}/\mathfrak{G}_i$ acts by conjugation on the representations of \mathfrak{G}_i and each orbit must have either a single element or $|\mathfrak{G}_{i+1}/\mathfrak{G}_i|$ elements. By Clifford theory, a \mathfrak{G}_{i+1} -irreducible representation is either an extension of a \mathfrak{G}_i -irreducible representation or is itself the induction of $|\mathfrak{G}_{i+1}/\mathfrak{G}_i|$ distinct irreducible representations of \mathfrak{G}_i . In either case the corresponding $M_{\mathfrak{G}_{i+1}}(X)$ can be computed in a circuit of size $O(|\mathfrak{G}_{i+1}|^{3/2})$ and depth $\text{polylog}|\mathfrak{G}_{i+1}|$ leading to a size $|\mathfrak{G}|^{3/2}$ and depth $\text{polylog}|\mathfrak{G}|$ circuit.

Finally, in the case of monomial groups—groups with representations all of whose matrices have only one nonzero entry in each row or column—a size $O(|\mathfrak{G}|\text{polylog}|\mathfrak{G}|)$ and depth $\text{polylog}|\mathfrak{G}|$ circuit exists. Because supersolvable groups (solvable groups whose \mathfrak{G}_i are normal in \mathfrak{G}) and metabelian groups are monomial, they have efficient parallel group FFTs.

Our discussion is summarized in the following theorem:

code. We expect the final version of our \mathfrak{S}_n -code to run much faster than the current version and to solve \mathfrak{S}_{12} -FFTs [746].

Theorem 7 *There exist circuits of size $|\mathfrak{G}|\text{polylog}|\mathfrak{G}|$ and depth $\text{polylog}|\mathfrak{G}|$ for the following classes of groups:*

- *Abelian groups*
- *Symmetric groups \mathfrak{S}_n [188, 190, 628].*
- *Metabelian groups [91, 186, 630].*
- *Supersolvable groups [189, pp.109–123].*

Futhermore, there exist circuits of size $|\mathfrak{G}|^{3/2}$ and depth $\text{polylog}|\mathfrak{G}|$ for solvable \mathfrak{G} , and size $|\mathfrak{G}||\mathfrak{G}|^{O(1)/n}$ for general $GL_n(q)$.

7.4.4 Group circulants

The definitions of \mathfrak{G} -circulant matrix and of \mathfrak{G} -invariant matrix were contained in section 3.2, and arose also in the context of the chess move generator matrix. It is not difficult to see that multiplication by a \mathfrak{G} -invariant matrix is equivalent to \mathfrak{G} -convolution; this is the connection between fast group Fourier transforms and the exploitation of symmetry.

Suppose that $\mathbf{M} = (\mathbf{M}_{\mathfrak{g}\mathfrak{g}'})_{\mathfrak{g},\mathfrak{g}' \in \mathfrak{G}}$ is an $n \times n$ matrix whose rows and columns are indexed by the elements of an n -element group \mathfrak{G} , and that $\mathbf{M}_{\mathfrak{g}\mathfrak{g}'} = \mathbf{M}_{\mathfrak{h}\mathfrak{g},\mathfrak{h}\mathfrak{g}'}$ for all $\mathfrak{g}, \mathfrak{g}', \mathfrak{h} \in \mathfrak{G}$.

Let \mathbf{v} be any \mathbb{C} -vector also indexed by the elements of \mathfrak{G} ; thus, \mathbf{v} can be thought of as an

element of $\mathbb{C}[\mathfrak{G}]$. The g th component of the product $\mathbf{M} \cdot \mathbf{v}$ is given by

$$(\mathbf{M} \cdot \mathbf{v})_g = \sum_{h \in \mathfrak{G}} M_{g,h} f(h) \quad (7.13)$$

$$= \sum_{h \in \mathfrak{G}} M_{h^{-1}g, \epsilon} f(h) \quad (7.14)$$

$$= (\mathbf{v} \star_{\mathfrak{G}} \mathbf{M}_{\epsilon})(g), \quad (7.15)$$

where \mathbf{M}_{ϵ} is the ϵ th column of \mathbf{M} . Hence $\mathbf{M} \cdot \mathbf{v} = \mathbf{v} \star \mathbf{M}_{\epsilon}$. Note that the equivalence of group-convolution and multiplication by a group circulant holds even when addition and multiplication are replaced by arbitrary operators.

It is now not difficult to see that the Fourier transform matrix block diagonalizes a \mathfrak{G} -circulant, where the sizes of the blocks are $d_i \times d_i$. This can be used to derive fast multiplication algorithms for \mathfrak{G} -circulant matrices when \mathfrak{G} admits a fast Fourier transform. Indeed, given a \mathfrak{G} -circulant matrix \mathbf{M} and a vector \mathbf{v} , we compute the \mathfrak{G} -Fourier transform $\hat{\mathbf{v}}$ of \mathbf{v} , which is a map assigning to each irreducible representation ρ of \mathfrak{G} of degree d_{ρ} a $d_{\rho} \times d_{\rho}$ matrix, compute the \mathfrak{G} -Fourier transform $\hat{\mathbf{M}}_{\epsilon}$, multiply corresponding matrices. The inverse \mathfrak{G} -Fourier transform of the result will be the desired product. The inverse \mathfrak{G} -Fourier transform is within a constant factor of the complexity of the \mathfrak{G} -Fourier transform, so that the complexity of convolution, which would be $|\mathfrak{G}|^2$ by brute force, becomes the sum $3T_{\mathfrak{G}} + \sum_{\rho} d_{\rho}^3$, where the sum is over all irreducible representations ρ . If we let $d_{\mathfrak{G}}^{\alpha}$ denote $\sum_{\rho \in \mathcal{R}} d_{\rho}^{\alpha}$, then group convolution requires time $O(d_{\mathfrak{G}}^3 + (T_{\mathfrak{G}}))$ assuming a cubic matrix multiply. Since $d_{\mathfrak{G}}^3 \leq |\mathfrak{G}|^{3/2}$ this gives an $|\mathfrak{G}|^{3/2}$ time complexity for group convolution for the classes of groups considered so far; since matrix multiplication is parallelizable these

can be computed in circuits of polylogarithmic depth.

This block-diagonalization property of \mathfrak{G} -Fourier transforms is the basis for a wide range of applications, which are discussed in the next section.

7.4.5 Applications

Several years after the publication of famous 1965 Cooley-Tukey paper on the fast Fourier transform [200] the *IEEE Transactions on Audio and Electroacoustics* published a “Special issue on the fast Fourier transform and its application to digital filtering and spectral analysis,” which contained the influential survey [193], the historical summary [199], and a number of other interesting articles, including a spectral analysis of the song of the killer whale. A brief article by Emanuel Parzen, “Informal comments on the uses of power spectrum analysis,” provided a clear motivation for applications of the classical fast Fourier transform:

I would like to offer a final idea that may be useful. People are very often interested in classifying patterns or records (for example, cardiograms). That is, one may want to decide whether a cardiogram is from a “good” patient. Various techniques are being considered for examining the record and performing some kind of analysis on it. It seems to me that one ought to consider taking a Fourier transform of these records, and work with that in the same role. That is, whenever someone thinks of a time domain approach to a problem, one should consider taking the Fourier transform of the time record of that sample and use that. Similarly, when one talks about pattern recognition in the plane, people are interested in recognizing the various letters of the alphabet. I have always wondered why they do not take a two-dimensional Fourier transform of the data; this might avoid some positioning problems. These are some of the ideas that have come to mind as I listened to talks on pattern-recognition problems. [590, p.76]

One way to understand Parzen's suggestions is to realize that analysis of a time-dependent function (time series) should be invariant to shifts in time; for instance, if analysis of a certain cardiogram suggests a prognosis, then the same prognosis is indicated if the cardiogram were taken a few minutes earlier or later. The engineering notion of the "time-domain" is, in our language, a particular instance of cyclic group invariance. Similarly, recognition of a character in the plane should be invariant under two-dimensional shifts of the character, hence Parzen's suggestion to take a two-dimensional Fourier transform of that data. The Fourier transform has the operational effect of making computation of invariant functions easier and of eliminating noise—irrelevant data—in the pattern.

This idea, of course, can be applied to more general groups using the machinery of group Fourier transforms; we thus have implicit parallelizations of such applications. Given any \mathfrak{G} -invariant function of data, if one *first* takes the \mathfrak{G} -Fourier transform of the data, then the function will be easier to compute. In fact, it will often depend only on the largest few Fourier coefficients.

Diaconis has given several applications in probability and statistics of these ideas [238,239]. For example, he considers the analysis of an election in which each voter ranks all the candidates in order [238, p. 142]. Any voter is essentially choosing a permutation in \mathfrak{S}_k , where there are k candidates. By taking the \mathfrak{S}_k -Fourier transform of the voters' choices, clusters of significant data are easier to see. Another example comes from the analysis of random walks on groups, as would arise, for example, in modeling the number of random transpositions required to shuffle a deck of cards well; the matrix of transition probabilities

is \mathfrak{S}_{52} -circulant, and the Fourier transform of this matrix lets us, for example, find the eigenvalues.

The notion that the \mathfrak{G} -Fourier transform selects key features is fundamentally the motivation behind the work on group-filters, which attempt to recover an input signal after it has been distorted by noise. Karpovsky and Trachtenberg have applied \mathfrak{G} -Fourier transforms to filtering and error detection [429,430,437,762]. Lenz [505,506] and Eberly and Wenzel [259] have used similar techniques, in the pattern-recognition in the plane, although Lenz' work is oriented more toward the continuous case, particularly of rotation groups, which we do not consider in this dissertation.

The theoretical motivation for some of the intuitive arguments we have presented may lie in the area of learning theory. Linial, Mansour, and Nisan have demonstrated connections between the learnability of a function and its Fourier coefficients [514]. Fourier transforms (still over Abelian groups, however) have been shown to be useful in a variety of learning problems, such as learning decision trees [443,472]. As explained by Clausen and Baum [189, Chapter 11], these ideas can be used as well when it is expected that the data would exhibit some non-Abelian invariance.

Rockmore and Lafferty have used fast Fourier transforms of linear groups over finite fields to explore the eigenvalues of their Cayley graphs; this can give information about graph-theoretical properties of these graphs, such as their diameter [185,476]. In fact, the adjacency matrix of the Cayley graph of \mathfrak{G} with respect to a set of generators S is the \mathfrak{G} -Fourier transform of the characteristic function of S at the regular representation for \mathfrak{G} .

A problem similar to fast manipulation of \mathfrak{G} -circulant matrices is the fast manipulation of \mathfrak{G} -equivariant matrices. Suppose that the rows and columns of an $n \times n$ matrix \mathbf{M} are indexed by an n -element set I on which \mathfrak{G} acts, and suppose that \mathbf{M} is \mathfrak{G} -equivariant. The block-diagonalization of \mathbf{M} can easily be reduced to the case where \mathbf{M} is \mathfrak{G} -circulant as follows. First suppose that the \mathfrak{G} action is fixed-point free. Let \mathcal{O} be a complete set of orbit representatives for the \mathfrak{G} action on I . Now, when $|\mathcal{O}| = |I/\mathfrak{G}| = 1$, then the \mathfrak{G} action is transitive and can be handled by previous techniques. When $|\mathcal{O}| > 1$, the only difficulty is notational. By reordering I , one decomposes \mathbf{M} into $|\mathcal{O}|^2$ \mathfrak{G} -circulant blocks each of size $|\mathfrak{G}| \times |\mathfrak{G}|$.

This idea has been extended recently to the case of an action that is not fixed-point free by Georg and Tausch [328]. \mathfrak{G} -Fourier transforms for \mathfrak{G} -equivariant matrices have been used in exploiting symmetry in the numerical solution of partial differential equations via finite-element, finite-difference, or boundary-element methods. These methods construct a matrix \mathbf{M} whose symmetry reflects the underlying symmetry in the problem: \mathbf{M} is normally \mathfrak{G} -equivariant. Often a linear system of the form $\mathbf{M}\mathbf{v} = \mathbf{b}$ must be solved, and \mathbf{M} must be eigensolved. Because an application of \mathfrak{G} -Fourier transform block-diagonalizes \mathbf{M} , the methods here will apply to this problem [19, 20, 22, 326–328, 738]. However, in most of this work the symmetry groups \mathfrak{G} to be considered are fairly small, and fast \mathfrak{G} -Fourier transform techniques are not necessary. The case of actions that may have fixed points has been considered by Georg and Tausch [328]. Healey and Treacy have considered the problem in the context of the mechanics of symmetrical structures [360]. An alternative approach,

	ϵ	τ	τ^2	τ^3	f	$f\tau$	$f\tau^2$	$f\tau^3$
W_0	1	1	1	1	1	1	1	1
W_1	1	1	1	1	-1	-1	-1	-1
W_2	1	-1	1	-1	1	-1	1	-1
W_3	1	-1	1	-1	-1	1	-1	1
T	$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$	$\begin{matrix} \omega & 0 \\ 0 & \omega^3 \end{matrix}$	$\begin{matrix} -1 & 0 \\ 0 & -1 \end{matrix}$	$\begin{matrix} \omega^3 & 0 \\ 0 & \omega \end{matrix}$	$\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix}$	$\begin{matrix} 0 & \omega^3 \\ \omega & 0 \end{matrix}$	$\begin{matrix} 0 & -1 \\ -1 & 0 \end{matrix}$	$\begin{matrix} 0 & \omega \\ \omega^3 & 0 \end{matrix}$

Table 7.1: The values of a complete set of irreducible inequivalent complex representations of \mathfrak{D}_4 at each element of \mathfrak{D}_4 , where $\omega = e^{2\pi i/4}$ is a primitive 4th root of unity. Each entry in this table is really a matrix: 1×1 matrices for the W_i and 2×2 matrices for T .

which is more like the orbit-decomposition algorithm of Chapter 6, was demonstrated in the case of finite-difference methods described by Hillis and Taylor [368].

7.5 Dihedral group transforms

We begin, by way of example, by considering \mathfrak{D}_4 , a group which was discussed in Section 3.1 and played a fundamental role in Chapter 6. It is generated by $\{\tau, f\}$ where $\tau^4 = f^2 = \epsilon$ and $\tau f = f\tau^3$. Fourier transforms for dihedral groups have been considered by Valenza [771], although he does not describe a fast algorithm. Vision applications are described in [259].

There are exactly five irreducible representations of \mathfrak{D}_4 . Four of them, W_0, W_1, W_2, W_3 , are of degree 1, and one of them, T , is of degree 2. The values of the representations on each element of \mathfrak{D}_4 are shown in Table 7.1.

The inequivalence of each of the representations in the $\mathcal{R} = \{W_0, W_1, W_2, W_3, T\}$ follows from the fact that their characters are inequivalent. That \mathcal{R} is a complete set of irreducible

representations follows from the fact that the sum of the squares of their degrees, $1^2 + 1^2 + 1^2 + 1^2 + 2^2 = 8 = |\mathfrak{D}_4|$.

Let $f: \mathfrak{D}_4 \rightarrow \mathbb{C}$ be an element of the group algebra $\mathbb{C}[\mathfrak{D}_4]$; we will think of f as being an element of \mathbb{C}^8 , that is, as a complex 8-vector under the ordering of \mathfrak{D}_4 given by the top row of Table 7.1.

The \mathfrak{D}_4 -Fourier transform \hat{f} of f is the function whose value at ρ is

$$\hat{f}(\rho) = \sum_{\mathfrak{g} \in \mathfrak{D}_4} f(\mathfrak{g})\rho(\mathfrak{g}).$$

The group \mathfrak{D}_4 has a normal cyclic subgroup

$$\mathfrak{N} = \{\epsilon, \tau, \tau^2, \tau^3\}.$$

Since \mathfrak{N} is cyclic, its representations are all one-dimensional and are given by Table 7.5.

Note that the entries of Table 7.5 form the matrix representation of the classical length-4 discrete Fourier transform:

$$F_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & -1 & \omega^3 \\ 1 & -1 & 1 & -1 \\ 1 & \omega^3 & -1 & \omega \end{pmatrix},$$

where ω is a primitive 4th root of unity.

Let $f_{\mathfrak{f}}(\mathfrak{d}) = f(\mathfrak{f}\mathfrak{d})$.

If h is any function (or representation) on \mathfrak{D}_4 , we write $h \downarrow \mathfrak{N}$ for the restriction to \mathfrak{N} of h .

We can represent the \mathfrak{D}_4 -Fourier transform of f at ρ in terms of the \mathfrak{N} -Fourier transforms

	ϵ	τ	τ^2	τ^3
C_0	1	1	1	1
C_1	1	ω	-1	ω^3
C_2	1	-1	1	-1
C_3	1	ω^3	-1	ω

Table 7.2: The values of a complete set of irreducible inequivalent complex representations of \mathfrak{N} at each element of \mathfrak{N} ($\omega = e^{2\pi i/4}$ is a primitive 4th root of unity.)

of the $f \downarrow \mathfrak{N}$ and $f_f \downarrow \mathfrak{N}$ as follows:

$$\widehat{f}(\rho) = \sum_{\mathfrak{d} \in \mathfrak{D}_4} f(\mathfrak{d})\rho(\mathfrak{d}) \quad (7.16)$$

$$= \sum_{\mathfrak{d} \in \mathfrak{N}} f(\mathfrak{d}) (\rho \downarrow \mathfrak{N})(\mathfrak{d}) + \rho(f) \sum_{\mathfrak{d} \in \mathfrak{N}} f_f(\mathfrak{d}) (\rho \downarrow \mathfrak{N})(\mathfrak{d}) \quad (7.17)$$

$$= \widehat{f \downarrow \mathfrak{N}}(\rho \downarrow \mathfrak{N}) + \rho(f) \widehat{f_f \downarrow \mathfrak{N}}(\rho \downarrow \mathfrak{N}). \quad (7.18)$$

The restrictions are given by the following equations:

$$W_0 \downarrow \mathfrak{N} = C_0 \quad (7.19)$$

$$W_1 \downarrow \mathfrak{N} = C_2 \quad (7.20)$$

$$W_2 \downarrow \mathfrak{N} = C_0 \quad (7.21)$$

$$W_3 \downarrow \mathfrak{N} = C_2 \quad (7.22)$$

$$T \downarrow \mathfrak{N} = C_1 \oplus C_3. \quad (7.23)$$

The matrix form of $F_{\mathfrak{D}_4}$ is given by

$$F_{\mathfrak{D}_4} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & \omega & -1 & \omega^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \omega^3 & -1 & \omega \\ 0 & 0 & 0 & 0 & 1 & \omega & -1 & \omega^3 \\ 1 & \omega^3 & -1 & \omega & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Equations 7.16 and 7.19 yield a factorization

$$F_{\mathfrak{D}_4} = ((I_2 \otimes F_2) \oplus I_4) \cdot P \cdot (I_2 \otimes F_4), \quad (7.24)$$

where P is an 8×8 permutation matrix. Explicitly,

$$P = (P_2^4 \oplus \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}) \cdot P_2^8.$$

In the language of section 4.3, equation 7.24 denotes the program that, given an input vector of length 8, first performs two parallel FFTs of length 4, and then two parallel FFTs of length 2; the permutations can be thought of as a readdressing.

A similar analysis shows that

$$F_{\mathfrak{D}_{2k}} = ((I_2 \otimes F_2) \oplus I_{4k-4}) \cdot Q \cdot (I_2 \otimes F_{2k}), \quad (7.25)$$

where Q is a permutation matrix and \mathfrak{D}_{2k} is a dihedral group of order $4k$. Indeed, \mathfrak{D}_{2k} has a normal cyclic subgroup of index 2, and it has 4 one-dimensional representations and $k-1$ 2-dimensional representations with restriction properties similar to that for \mathfrak{D}_4 [214, pp.333-340].

Equation 7.25 lends itself immediately to a parallel implementation using the techniques of section 4.3. The implementation was coded by running two separate copies of the CM-5 CM Fortran CMSSL FFT [409,418] [748, Chapter 10].²³ Since there is no canonical order for the rows of the \mathfrak{D}_n -FFT, the permutation Q was chosen to be simply a tensor product of l_2 with a bit-reversal permutation, thereby allowing the library FFT to leave its arguments in bit-reversed order. The program currently performs a $\mathfrak{D}_{2^{27}}$ -FFT over \mathbb{C} in about 2.9 seconds on a 512 node CM-5. Thus, the input and output vectors each comprises 2^{28} complex numbers with 32-bit real and imaginary parts (i.e., the input vector is about 2×10^9 bytes).

7.6 String matching

This section describes the application of \mathfrak{G} -Fourier transforms to generalized string matching. Subsection 7.6.1 presents some background on the problem, and the next subsection describes its formulation and solution in our model.

7.6.1 Background

This subsection briefly describes a few aspects of the area of string matching; of course, an enormous amount of work has been omitted from this description.

²³ The author thanks Roger Frye for his advice on attaining peak performance from the library FFT routines.

The earliest systematic approaches to the problem of text searching seems to have been the construction of concordances for Biblical texts, which date from the 13th century. Most early computer applications were, of course, numerical in nature; but one of the first articles on nonnumerical computing to appear in the *Journal of the Association for Computing Machinery* also contained the earliest use known to the author of the word “string” to mean a sequence of symbols manipulated by computer:

Areas are set aside for shuttling strings of control fields back and forth until a completely sorted sequence is obtained. Optimum results are realized (both with respect to time and memory space) under two string merging since the extra program steps required for greater than two string merging are more costly than the savings in passes over the data. [307, p. 147]

The modern concept of “string matching,” however, seems to date from about the mid-1960s, when the function was used in text editors.

One of the earliest algorithms that did better than brute-force was due to Ken Thompson, who gave an algorithm for matching regular expressions in 1968 [750]. Thompson’s elegant algorithm converted the pattern into an automaton, and thence into a sequence of instructions in IBM 7094 machine language that, together with run-time calls, searched for the original pattern.²⁴

A crucial breakthrough was provided by Peter Weiner’s famous “Linear pattern matching algorithm” [797]. Weiner gave a linear-time algorithm using “bi-trees,” an early form of

²⁴ “It is assumed that the reader is familiar with regular expressions [444] and the machine language of the IBM 7094 computer [207]” wrote Thompson in the article, which appeared in the *Communications of the ACM* [750].

suffix trees, for the classical string-matching problem. Suffix trees were streamlined in work by McCreight (1976) [542].

Knuth, Morris, and Pratt's famous paper [448] gave an automata-based linear pattern matching algorithm, which, although it appeared in 1977, was based on much earlier work. This paper not only laid the foundation for a wide class of automata-based algorithms, but also contained some interesting historical information (pp.338–340). For example, they describe the following amusing incident:

One of the authors (J. H. Morris) was implementing a text editor for the CDC 6400 computer during the summer of 1969, and since the necessary buffering was rather complicated he sought a method that would avoid backing up the text file. Using concepts of finite automata theory as a model, he devised an algorithm equivalent to the method presented above [i.e., the KMP algorithm], although his original form of presentation made it unclear that the running time was $O(m + n)$. Indeed, it turned out that Morris' routine was too complicated for other implementors of the system to understand, and he discovered several months later that gratuitous “fixes” had turned his routine into a shambles. [448, p.338]

Knuth, Morris, and Pratt trace the development of algorithms like theirs, which avoid backing up after a mismatch, to work by E.N. Gilbert from 1960 on comma-free codes [334]; Gilbert had needed algorithms for quickly recognizing strings that signaled the beginning of a block of codewords. They use a generalization of a lemma on periodicity properties of strings which originally appeared in a 1962 article by Lyndon and Schützenberg [527]; also see [470].

These algorithms are fundamentally automata-based [209]. They have considerable flexibility for solving a wide range of combinatorial questions about strings, for example, matching

multiple patterns [9], repetition-detection [60,528], also see [10]. A detailed complexity analysis is contained in the pair of articles [315,316]. By contrast, the algorithms we present will be based on generalizations of convolution-based algorithms, such those of [2,295]. The Aho-Corasick technique has been applied to array-matching, in which each column is thought of as a single character in a very large alphabet [46,47,79,126]. For multi-dimensional approximate algorithms see [48,462]. Parallelization is discussed in [59,484,785]. This is also related to parallel language recognizers, see [460] for grid methods. Fast heuristic methods are given in [139]; similar is [211]. For randomized algorithms see [425]; Harrison (1971) did early work on hashing as well [356].

Intensive work continues in the areas of computing combinatorial properties of strings, motivated especially by genome applications; the Combinatorial Pattern Matching conference proceedings contain additional references [57,58,210].

7.6.2 Mathematical formulation

In order to apply the group formulation it is necessary to formalize the notion of string matching.

A *string* over a finite *alphabet* Σ is a map $s: \{0, \dots, m-1\} \rightarrow \Sigma$. We call m the *length* of the string and write $s_i = s(i)$. We will always deal with two strings, p and t , the *pattern* and the *text*, of lengths m and n respectively. These may be over different alphabets, Σ_P and Σ_T , respectively.

The classical string-matching problem is to find all occurrences of $\Sigma_{\mathbf{P}}$ in $\Sigma_{\mathbf{T}}$. This formulation can be generalized in several ways:

1. The definition of a “match” between a pattern character and a text character can be generalized to predicates other than simple equality.
2. The requirement that each character in the pattern match each character in the text can be generalized to allow partial matches.
3. The allowable operations on the pattern to align it with the text can be modified from the default one of “rigid sliding.”

These modifications are formalized as follows.

We are given two operators, a *character comparison* operator $\boxtimes: \Sigma_{\mathbf{P}} \times \Sigma_{\mathbf{T}} \rightarrow R$, and a *combining* operator $\boxplus: R \times R \rightarrow R$. We assume that \boxplus is associative, but no restrictions are placed on the operator \boxtimes . The character comparison operator formalizes generalized comparisons between the pattern and the text, and the combining operator formalizes the notion of allowing partial matches.

There are several ways to formalize the notion of generalized alignments. We make the simplifying assumption that $m = n$, which can always be done by adding an identity for \boxplus to R and padding the pattern with don't cares. Let $X = \{0, \dots, n - 1\}$; thus, $p, t: X \rightarrow \Sigma$. We call p and t *strings on X* . The *support* of a string on X is the set of elements in X that are not don't cares. Now we define an *alignment set* A to be a set of permutations of X . Each alignment $a \in A$, $a: X \rightarrow X$ induces a mapping of the set of strings on X by

$$(a(s))_j = s_{a(j)}.$$

We define the generalized product of two strings on X as follows:²⁵

$$s \begin{array}{|c|} \hline \boxplus \\ \hline \boxtimes \\ \hline \end{array} s' = \boxplus_{x \in X} (s_x \boxtimes s'_x).$$

The *generalized string matching problem* for alignment set A and comparison operation $\begin{array}{|c|} \hline \boxplus \\ \hline \boxtimes \\ \hline \end{array}$ is to compute $a(p) \begin{array}{|c|} \hline \boxplus \\ \hline \boxtimes \\ \hline \end{array} t$ for each $a \in A$.

In the *classical exact string-matching problem* the character comparison operator is $\equiv \Sigma_{\mathbf{P}} \times \Sigma_{\mathbf{T}} \rightarrow \{0, 1\}$, where $\Sigma_{\mathbf{P}} = \Sigma_{\mathbf{T}} \cup \{\phi\}$ and ϕ is a “don’t care” character that matches any element of $\Sigma_{\mathbf{T}}$. Thus $(\sigma \equiv \sigma') = 1$ iff $\sigma = \phi$ or $\sigma = \sigma'$. In this case, the combining operator is simply boolean AND (\wedge), $X = \mathbb{Z}_n$ is a finite cyclic group, $A = \mathbb{Z}_n$ is identified with the set of translations of X ($a(x) = x + a \bmod n$), and the pattern is restricted to containing no ϕ s inside an interval of length m containing its support.

The formulation we gave is built upon the theoretical framework of Muthukrishnan, Palem and Ramesh in the area of non-standard stringology [574, Chapter 3] [575,576], who consider general classes of character comparison operations and combining functions. They consider arbitrary match relations \boxtimes . They thereby obtain a graph that they call the *matching graph* of the problem. This is a bipartite graph whose nodes are the disjoint union of $\Sigma_{\mathbf{P}}$ and $\Sigma_{\mathbf{T}}$ and for which there is an edge from pattern element x to text element y whenever x matches y . They relate the complexity of the matching problem for a given match relation to graph-theoretical properties of the match-graph. For example, they show that if the

²⁵ This notation is taken from [295].

matching-graph can be covered by k disjoint cliques, then a pattern of size m can be matched against a text of size m using $\log(k)$ linear boolean convolutions of size n and m , again using Fischer-Paterson techniques.

More general character comparison operations, combinators, and alignments have been considered by a number of authors. We review these generalizations now:

First: General character comparison operations were considered by Fischer and Paterson in their seminal 1973 paper [295]. They consider string matching when $\Sigma_{\mathbf{P}}$ contains a special “don’t care” character ϕ that matches any character in the alphabet [295]. For example, the pattern string $\mathbf{ab}\phi\mathbf{d}$ would match the text strings \mathbf{abcd} and \mathbf{abgd} and so on. Fischer and Paterson reduce the problem of string matching with don’t cares to integer multiplication. This gives an $O(\log(|\Sigma|)n \log m \log \log m)$ bit-complexity algorithm. The doubly-logarithmic factor arises because they are using the Schönhage-Strassen integer multiplication algorithm [670], which uses the FFT over a finite ring and recursively uses FFT’s to perform the arithmetic required in the larger FFT’s.

Abrahamson pursued the idea of generalized string matching further [2]. He considered the case where a pattern character can denote subsets of elements from $\Sigma_{\mathbf{T}}$. For example, the pattern element $\langle \mathbf{a|b|c} \rangle$ matches a text character of either \mathbf{a} , \mathbf{b} , or \mathbf{c} . The pattern string $\mathbf{de}\langle \mathbf{a|b|c} \rangle\mathbf{f}$ would match texts \mathbf{decf} and \mathbf{deaf} but not \mathbf{dedf} . Abrahamson also permitted complements of subsets of characters in $\Sigma_{\mathbf{T}}$ in the pattern, using the notation $[\mathbf{a|b|c}]$ to denote all the characters in Σ except for \mathbf{a} , \mathbf{b} , and \mathbf{c} . Thus, the pattern element $[\]$ is equivalent to Fischer-Paterson’s don’t care symbol.

Abrahamson showed that such “generalized” patterns of size m could be matched against a text of size n in time $O(n\sqrt{m}\text{polylog}(m))$ time. His method was similar to Fischer-Paterson’s: he reduced the problem to $O(\sqrt{m})$ convolutions (as compared to the $O(\log(m))$ reduction used by Fischer-Paterson). Perhaps the major difficulty in discussing string matching with subsets is notational: there are several slightly different metrics for the “length” of the pattern that one can use. These are the total number of symbols (including brackets), the length of a string against which it can be matched, and the number of actual alphabet symbols. Like most other authors, in order to simplify the notation we will just assume in the sequel that these measures are all linearly related; many of the results do go through for other patterns though.

Second: General *combining operations* are typically used in forms of *approximate matching*. Approximation algorithms, in which only a certain number of mismatches are allowed between the pattern and the text, have been considered in [312,313,485]; an application to nucleotide sequence matching is in [491]; see [662] for other applications in biology. However, our formulation does not encompass models in which characters in the pattern may match 0 characters in the text, or in which characters may be inserted in the text. This is the *k-differences* problem, and is often used in biological applications [488,491,662]. A survey is in [314]; see [196,206,208] for parallelizations.

In *counting matching* the combining operator is replaced by addition, so that the goal is to count the number of mismatches at each alignment; this problem was also considered by Fischer and Paterson and others. In threshold matching, the combining operator is

normally max or min, and the character comparison operator takes values in the positive reals.

Third: The class of allowed *alignments* can be broadened. In classical string matching, the pattern and the text are both linear strings. The pattern rigidly slides along the text until a match is attained. Equivalently, one can view the text as being arranged in a circle, and the pattern is slid along the circle, which as we have seen, is equivalent to group matching over a cyclic group.

The case of string matching over an Abelian group is equivalent, by the structure theorem on Abelian groups, to array matching, in which the pattern and the text are rank k arrays, where the group is the direct product of k cyclic groups. Array matching algorithms were initiated by Bird and Baker [79, 126], who use automata techniques in the style of [9]; also see [46, 47, 390, 427]. Of course, the modifications can be combined; for example, for multi-dimensional *approximate* algorithms see [48, 462]; array matching is typically studied for its application to image processing [490]; also see [735].

Some general classes of alignments do not form groups. The most important of these is tree-matching [256, 375, 461], for which Kosaraju used a combination of convolution and suffix-tree based techniques in the ordered labeled case [461]; the motivation for tree-matching was originally from logic programming [375]. Other forms of tree matching are considered in [18, 424, 529, 783].

Very general group alignments have recently been considered in the context of the mapping of molecular structures with specific 3-dimensional structures; this set forms a group, more

precisely, the legal alignments of the molecules forms a group, but the group may not be known a priori, so our techniques may not be applicable [294].

For additional references see [574] and the recent Conferences on Combinatorial Pattern Matching.

An important case of the general alignment problem addressed in this section is the case when the alignments form a permutation group, that is, there is a permutation group \mathfrak{G} acting on $\{0, \dots, n-1\}$. For specificity, suppose that initially the pattern p is aligned with the text elements $t_0 \dots t_{m-1}$. Each group element $g \in \mathfrak{G}$ induces an alignment of the pattern p in the text sending p_i to p_{gi} .

7.6.3 Reducing generalized matrix multiplication to matrix multiplication over \mathbb{C}

In the previous subsection, we formulated the concept of a generalized string-matching problem over an arbitrary alignment set. Such a problem has a natural expression as a generalized matrix multiplication problem in which the symmetry in the associated matrix reflects the symmetry in the alignment set A . For example, if the set of alignments forms a group \mathfrak{G} , then the matrix M will be \mathfrak{G} -invariant. We would like to use the \mathfrak{G} -FFT algorithms of section 7.4. However, these algorithms only work when the matrix entries are from a field, for example \mathbb{C} .

Therefore, in order to apply the parallel group FFT algorithms to the problem of group

string matching, we first reduce the problem of generalized matrix multiplication to matrix multiplication over \mathbb{C} .

We consider several types of generalized matrix multiplication problems arising from exact matching with don't cares, counting matching with don't cares, exact matching with general match relations, counting matching with general match relations, and Abrahamson-style-matching.

We write the generalized matrix product to be computed as $\mathbf{M} \begin{matrix} \boxplus \\ \boxtimes \end{matrix} \mathbf{v}$, which we sometimes abbreviate to $\mathbf{M}\mathbf{v}$, where \mathbf{M} is an $n \times n$ matrix with entries in $\Sigma_{\mathbf{P}}$, \mathbf{v} is an n -vector with entries in $\Sigma_{\mathbf{T}}$, and $+$ and \times are replaced by \boxplus and \boxtimes respectively:

$$\left(\mathbf{M} \begin{matrix} \boxplus \\ \boxtimes \end{matrix} \mathbf{v} \right)_i = (\mathbf{M}_{i,0} \boxtimes \mathbf{v}_0) \boxplus \cdots \boxplus (\mathbf{M}_{i,n-1} \boxtimes \mathbf{v}_{n-1}).$$

As explained above, in exact-matching with don't cares, $\Sigma_{\mathbf{P}} = \Sigma_{\mathbf{T}} \cup \{\phi\}$, $R = \{0, 1\}$. For this case, generalized matrix multiplication is no harder than complex matrix multiplication:

Theorem 8 *The generalized matrix multiplication $\mathbf{M} \begin{matrix} \boxplus \\ \boxtimes \end{matrix} \mathbf{v}$ can be performed in one complex matrix multiplication.*

Proof: Let ω be a primitive $\max(|\Sigma_{\mathbf{T}}|, 3)$ th root of unity. Construct a complex matrix $\tilde{\mathbf{M}}$ by replacing every occurrence in \mathbf{M} of the j th symbol in $\Sigma_{\mathbf{T}}$ by ω^j , and every occurrence of ϕ by 0. Similarly, construct $\tilde{\mathbf{v}}$ by replacing each occurrence of the j th symbol in $\Sigma_{\mathbf{T}}$ by ω^{-j} . Let c_i be the number of ϕ s in the i th row of \mathbf{M} . The product of two roots of unity equals 1 if and only if the roots are conjugate. Therefore, the i th component of the product $\tilde{\mathbf{M}}\tilde{\mathbf{v}}$ equals $n - c_i$ if and only if the i th row of \mathbf{M} matches \mathbf{v} .

Remark: Because we are performing exact complex arithmetic, as is standard in the analysis of \mathfrak{G} -FFTs, it is natural to ask whether our algorithms in fact require too much precision to be practical in a fixed-precision implementation (this point was raised by Kosaraju in personal communication). In the cyclic case, this question has been addressed by Knuth [447, pp.290–295], and the underlying stability of the classical FFT algorithms should make precision requirements reasonable in fixed-point implementations. We have not studied the precision issue in the case of general \mathfrak{G} -FFTs, although we doubt it would be a problem (also see, e.g. [566,633,634]). If, in fact, precision were a problem than the matter would be easily remedied by performing boolean \mathfrak{G} -convolution using \mathfrak{G} -FFTs over a suitable finite field \mathfrak{F} , as the group-algebra decomposition goes through for appropriate finite fields as well, given some minor conditions on the fields.

We now consider counting-matching, in which the number of mismatches must be computed. In this problem, we use the same character comparison operator \equiv , but we interpret the domain $R = \mathbb{N}$, the natural numbers. We let the combining function \boxplus be addition. We now have:

Theorem 9 $M_{\equiv}^{\boxplus} \mathbf{v}$ can be performed in $|\Sigma_{\mathbb{T}}| + 1$ complex matrix multiplications.

Proof: For each symbol $\sigma \in \Sigma_{\mathbb{T}}$, construct \widetilde{M}_{σ} by replacing each occurrence of σ in M by a 1 and every other symbol by a 0; similarly for $\widetilde{\mathbf{v}}_{\sigma}$. It is then easy to see that

$$\left(M_{\equiv}^{\boxplus} \mathbf{v} \right)_i = \sum_{\sigma \in \Sigma_{\mathbb{T}}} \left(\widetilde{M}_{\sigma} \widetilde{\mathbf{v}}_{\sigma} \right)_i + c_i,$$

where the c_i are as in Theorem 8.

Finally, we consider matching in the style of Abrahamson. In this problem, $\Sigma_{\mathbf{P}}$ comprises two disjoint copies, $\Sigma_{\mathbf{P}_1}$ and $\Sigma_{\mathbf{P}_2}$ of the set of subsets of $\Sigma_{\mathbf{T}}$. The character comparison operator \sim is defined as follows. If $p \in \Sigma_{\mathbf{P}_1}$ then

$$p \sim \sigma = \begin{cases} 1 & \text{if } \sigma \in p \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, if $p \in \Sigma_{\mathbf{P}_2}$ then

$$p \sim \sigma = \begin{cases} 0 & \text{if } \sigma \in p \\ 1 & \text{otherwise.} \end{cases}$$

Abrahamson-style matching is somewhat more complex than classical exact matching:

Theorem 10 *Computation of $\mathbf{M} \begin{bmatrix} + \\ \sim \end{bmatrix} \mathbf{v}$ requires n^2/s matrix multiplications and additional overhead of ns for positive integer s .*

Proof: We follow the method of Abrahamson [2]. First, write $\mathbf{M} = \mathbf{M}^+ + \mathbf{M}^-$, where \mathbf{M}^+ comprises only positive pattern elements and a 0 symbol matching no element of $\Sigma_{\mathbf{T}}$, and \mathbf{M}^- comprises only negative pattern elements and 0's. Then $\mathbf{M}\mathbf{v} = \mathbf{M}^+\mathbf{v} + \mathbf{M}^-\mathbf{v}$, so that it suffices to consider left multiplication by \mathbf{M}^+ and \mathbf{M}^- separately.

Let $\mathbf{M}_{\leq s}^+$ be \mathbf{M}^+ restricted to symbols that occur at most s times in \mathbf{M}^+ (for some arbitrary integer s , to be specified later) and let $\mathbf{M}_{> s}^+$ be \mathbf{M}^+ restricted to symbols that occur more than s times in \mathbf{M}^+ . That is, to construct $\mathbf{M}_{\leq s}^+$ we delete from any pattern element each symbol that occurs more than s times in \mathbf{M}^+ , leaving 0's unaltered.

First we compute $\mathbf{M}_{\leq s}^+ \begin{bmatrix} + \\ \sim \end{bmatrix} \mathbf{v}$. Each entry in \mathbf{v} will point to a list of all the (at most s) symbols that match it. There are thus at most ns match pairs between entries in \mathbf{M} and entries in \mathbf{v} ,

each of which is associated with at most one entry in the product $\mathbf{M}\mathbf{v}$. Therefore, the total time to compute $\mathbf{M}_{\leq s}^+ \mathbf{v}$ is, up to possibly logarithmic factors, $O(ns)$. We remark that if \mathbf{M} is \mathfrak{G} -circulant then this time is reduced by a factor of $|\mathfrak{G}|$, assuming that \mathfrak{G} -multiplication and \mathfrak{G} -inversion is fast.

In order to compute $\mathbf{M}_{> s}^+ \mathbf{v}$, we observe that there are at most $\frac{n^2}{s}$ distinct symbols in $\mathbf{M}_{> s}^+$. We can loop over these and thereby compute $\mathbf{M}_{> s}^+ \mathbf{v}$ in $\frac{n^2}{s}$ matrix multiplications of $n \times n$ complex matrices by complex n -vectors.

Putting these two parts together, we see that computation of $\mathbf{M}^+ \mathbf{v}$ can be done in $O\left(\frac{n^2}{s}\right)$ matrix multiplications and additional overhead of $O(ns)$, where the overhead decreases linearly with the size of the invariance group.

A similar method works for computing \mathbf{M}^- . This proves the theorem.

7.6.4 The application of group FFTs to parallel string matching

Theorems 8 and 10 have shown that when the set of alignments forms a group \mathfrak{G} , the related matching problems can be reduced to a generalized multiplication by a \mathfrak{G} -circulant matrix, which then reduces to a series of multiplications by complex \mathfrak{G} -circulants. Informally, one says that the reductions of Fischer-Paterson and Abrahamson preserve symmetry.

This now easily yields algorithms for string matching on groups, because multiplication by the resulting \mathfrak{G} -circulants can be performed in via fast and parallelizable \mathfrak{G} -FFTs.

We let $P_{\mathfrak{G}}$ be the number of processors required to perform \mathfrak{G} -convolution in polylogarithmic

time. We have seen that $P_{\mathfrak{G}}$ is of the order of $d^3(\mathfrak{G})$ for symmetric, alternating, metabelian, supersolvable, and monomial groups and that $P_{\mathfrak{G}}$ is of the order of $\max(|\mathfrak{G}|^{3/2}, d^3(\mathfrak{G}))$ for solvable groups, where, of course, we may replace $d^3(\mathfrak{G})$ by $d^\alpha(\mathfrak{G})$, for α the exponent of matrix multiplication, if we are willing to use the asymptotically fast matrix multiplication algorithms.

Theorem 11 *Exact string matching with don't cares over a group \mathfrak{G} can be performed using $P_{\mathfrak{G}}$ processors in polylogarithmic time. Abrahamson-style string matching can be performed using $\max(P_{\mathfrak{G}}, |\mathfrak{G}|^{3/2})$ processors in polylogarithmic time.*

Proof: Choose $s = n^{3/2}$ in Theorem 10.

The Abrahamson and Fischer-Paterson reductions respect tensor products; when $l_k \otimes M$ is defined in the obvious manner, then the reductions given by the theorems reduce $(l_k \otimes M) \begin{bmatrix} \oplus \\ \otimes \end{bmatrix} \mathbf{v}$ to a generalized multiplication of complex matrices of the form $(l_k \otimes \tilde{M}) \mathbf{w}$. This latter product can be performed via fast matrix-multiplication, such as by Coppersmith-Winograd. For example, n length n patterns can be matched against n length n strings, even in the presence of don't cares, in sub-cubic time; this idea is similar to Valiant's well-known reduction of context-free language recognition to a very general form of matrix multiplication [772].

7.7 Future work

The preliminary \mathfrak{S}_n -FFT implementation needs to be polished. We expect to improve our preliminary timings by a factor of at least three, and to compute \mathfrak{S}_{12} -FFTs.

The exact complexity of group Fourier transforms for arbitrary finite groups continues to be open. A promising line of attack along these lines might be to use the Classification theorem for finite simple groups. Good algorithms for many of the infinite classes of finite groups are known [537], and, from a complexity-theoretic point of view, the complexity of FFTs for the sporadic simple groups can be folded into the constant factor. Given a group \mathfrak{G} and a maximal normal subgroup \mathfrak{H} , then the quotient $\mathfrak{G}/\mathfrak{H}$ must be simple. It is tempting to apply Clifford theory to try to derive good \mathfrak{G} -FFTs from the inductively good FFTs of the factors $\mathfrak{G}/\mathfrak{H}$ and \mathfrak{H} . (In fact, it is an easy consequence of the classification theorem that \mathfrak{G} has a subgroup of size at least $\sqrt{|\mathfrak{G}|}$ [293]. This result combines with the usual arguments from Frobenius reciprocity [189], to give an $O(|\mathfrak{G}|^{1.75})$, even without using fast matrix-multiplication algorithms.)²⁶

In string matching, the main area for further research would be to apply convolution techniques to more general classes of pattern-matching problems. It would also be of interest to describe a unified mathematical theory that specializes to suffix-trees, in some sense. We have seen that multiplication by a group-equivariant matrix can be performed by transform-

²⁶ Michael Clausen, in recent personal communication with the author, has announced that the large subgroups predicted by the Classification Theorem do in fact yield an $O(|\mathfrak{G}|^{1.44})$ time algorithm for general \mathfrak{G} -FFTs when Coppersmith-Winograd matrix multiplication bounds are used.

ing the operands into spectral space and performing a pointwise product there. Similarly, a suffix-tree based approach first transforms the text into a suffix tree, and applies a very simple matching algorithm on the suffix tree.

Finally, it would be interesting to obtain empirical results on the efficacy of group Fourier-transform based learning algorithms.

Chapter 8

Equivariant factorization of Abelian groups

Each of the problems considered in this chapter will be phrased as an appropriate \mathcal{G} -invariant mapping over spaces \mathfrak{V} and \mathfrak{W} where \mathcal{G} is Abelian. The associated matrix M will be factored into primitives that correspond to the basic computational primitives supported by the machine.

The applications we consider here are:

1. Describing Fortran 90 communication intrinsics. There are two parts to this application: first, we write down explicit matrix representations of common Fortran 90 intrinsics and observe that they correspond to \mathcal{G} -equivariant matrices for appropriate \mathcal{G} . The goal of this formulation is simply to try to elucidate the fundamental primitives in Fortran 90. Second, we make the simple observation that, on some compilers, it can be faster to use external linear algebra routines than to use the Fortran 90

reduction and broadcast operator. Despite its triviality this amounts to a technique for speeding up reduction and broadcast on various architectures. Some timings for Cray, CM-5 and CM-200 will be given to support this.

2. The next implementation is of n -body simulation. A direct method is used, as, even in the context of multipole or other tree methods, direct solvers are useful at the leaves. By applying the new Fortran 90 communication intrinsics, we implemented and sped up an direct n -body solver for the analysis of flux dynamics in superconducting thin-films. This implementation used a semiring version of a tuned tensor-product routine.
3. Finally, we describe parallelization of a convolution arising in the analysis of protein simulation in water. This involved an interesting application of nested scans, for which we provide a \mathcal{G} -invariant matrix formulation. The specific convolution we solved was counting the number of contiguous subsequences of length k , $k = 1, \dots, n$ in a binary string of length n .

Although these applications are fairly straightforward, the symmetry is not immediately apparent, and the speedup that was obtained was comparatively high.

8.1 Fortran 90

Fortran has undergone significant changes since its introduction but remains the most widely used and important language for the design of high-performance scientific computing applications. There are a number of proposed extensions to the hoary Fortran 77 standard,

including High-performance Fortran, Fortran D, Fortran 90 and many vendors' proprietary extensions to Fortran, such as Cray Fortran and CM-Fortran. By the same token, the efficiency of many standard scientific programs are tied to their efficiency as Fortran code.

This section analyzes several representative Fortran 90 communication intrinsics. They will be presented as matrices and factored.

The Fortran 90 definition of `CSHIFT` is as follows. Given a (Fortran) rank k array \mathbf{A} , a positive integer D , and a distance to shift S , `CSHIFT(A, DIM = d, SHIFT = s)` is the rank k array obtained by circularly shifting the k th dimension of \mathbf{A} to the right a distance s .

For example, when \mathbf{v} is a 1-dimensional array, which is to say, a vector, then we can write

$$\mathbf{A} = \begin{pmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{n-1} \end{pmatrix}$$

$$\text{CSHIFT}(\mathbf{v}, \text{DIM} = 1, \text{SHIFT} = 1) = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_{n-1} \\ \mathbf{v}_0 \end{pmatrix} \quad (8.1)$$

Let us write `CSHIFT(v)` or just \mathbf{Cv} for `CSHIFT(v, DIM = 1, SHIFT = 1)`.

Then \mathbf{C} may be written as a matrix:

$$\mathbf{C} = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \quad (8.2)$$

Note that \mathbf{C} is invariant under the cyclic group \mathfrak{C}_n .

Now, let \mathbf{A} be a rank k vector with dimensions (d_1, \dots, d_k) . Then the corresponding matrix to compute the generalized $\text{CSHIFT}(\mathbf{A}, i, s)$ is simply

$$\bigotimes_{j=1}^{i-1} I_{d_j} \otimes C_{d_i} \otimes \bigotimes_{j=i+1}^k I_{d_j}$$

Implementation of CSHIFT on a hypercube is equivalent to embedding a grid in a hypercube, a problem which, in its simplest form, can be easily solved by Gray coding the coordinates, as observed by Gilbert in 1958 [333], also see section 2.1. When the grid size is not a power of two and the dilation needs to be minimized then good embeddings are still known [171, 371]. Therefore, any matrix that can be factored in terms of CSHIFT s is a priori efficiently computable.

Collecting terms in the tensor product, we get the matrix of

$$I_{\prod_{j < i} d_j} \otimes C_{d_i} \otimes I_{\prod_{j > i} d_j}$$

It is possible to use this factorization to derive parallel algorithms for implementing CSHIFT on a hypercube and other networks, by factoring the permutation matrices implemented by the hypercube, for example, as has been explicitly carried out by Kaushik, et al. [434, 435].

Similar equations obtain for EOSHIFT , which is defined just like CSHIFT except that there is no wraparound, instead, 0s are filled in.

Given a rank k array \mathbf{A} , $\text{SUM}(\mathbf{A}, \text{dim} = r)$ is the rank $k - 1$ array formed by summing along the r th dimension of \mathbf{A} .

Let \mathbf{K}_n be a column vector of 1s of length n . When \mathbf{A} is a rank 1 array, then $\text{SUM}(\mathbf{A}, \text{DIM} = 1) = \mathbf{K}_n^T \cdot \mathbf{A}$.

Suppose \mathbf{A} is an $n \times m$ matrix. Then by our convention above, \mathbf{A} is stored as an nm vector. $\text{SUM}(\mathbf{A}, \text{DIM} = 1)$ is the m -vector that is the sum of the rows of \mathbf{A} , and therefore can be computed by taking the sum of each column independently. Therefore, $\text{SUM}(\mathbf{A}, \text{DIM} = 1) = \mathbf{1}_m \otimes \mathbf{K}_n^T \cdot \mathbf{A}$.

This is also $\mathbf{K}^T \cdot \mathbf{A}$, where \mathbf{A} is considered as a matrix; similarly $\text{SUM}(\mathbf{A}, \text{DIM} = 2)$ can be written as $\mathbf{A} \cdot \mathbf{K}$ in matrix form. This formulation has the additional advantage that it seems more canonical than the somewhat esoteric syntax of the Fortran 90 `SUM`.

The `PROD` intrinsic behaves just like `SUM`, but taking products rather than sums. This can be handled easily in our framework, using an appropriate semiring, for example, $(\mathbb{R}, \max, \times)$.

The `SUM` and `PROD` intrinsics are instances of the reduction primitive used in data-parallel programming, and our formulation thus applies to general reductions as well.

We next consider the FORTRAN 90 version of the broadcast primitive, the `SPREAD`. Given a 1-dimensional array \mathbf{A} of length n ,

$$\text{SPREAD}(\mathbf{A}, \text{DIM} = 1, \text{NCOPIES} = m)$$

is the $n \times m$ matrix formed from the copies of \mathbf{A} . It can be seen to equal the tensor product $\mathbf{K} \otimes \mathbf{A}$. Similarly, $\text{SPREAD}(\mathbf{A}, \text{DIM} = 2, \text{NCOPIES} = m)$ is the tensor product $\mathbf{A} \otimes \mathbf{K}$.

Interestingly, these formulations also result in faster running times for `SUM` and `SPREAD` on 3 classes of machines that we tried. These were a massively-parallel hypercube, the CM-200,

n	1024	2048	4096	8192
SUM	19.8	70.9	275	—
MATMUL	4.21	7.39	11.6	28.4
SPREAD	13.1	43.7	159	622
\otimes	5.59	11.0	28.7	100.0

Table 8.1: Timings in milliseconds for certain FORTRAN 90 intrinsics versus equivalent algebraic formulation for varying n . Timings were obtained on one sequencer (512 nodes) of a CM-200 running slicewise CM FORTRAN 1.1. SUM failed due to insufficient memory on the $n = 8K$ problem.

n	128	256	512	1024
SUM	0.892	3.012	11.844	44.096
MATMUL	0.216	0.840	3.396	13.148

Table 8.2: Timings in milliseconds for SUM vs. matrix vector multiplication. Timings were performed on a Cray X-MP/24 by L. L. Daemen of Los Alamos National Laboratory. The CALMATH (Cray Assembly Language Mathematical Library) was used.

a message-passing fat-tree, the CM-5, and a vector machine, the Cray Y-MP. These results are summarized in the accompanying tables

One interpretation of this curious set of timings is that the fundamental operations of inner-product and outer-product were optimized more carefully than reduction and broadcast.

n	256	512	1024	2048
SUM	15.1	72.1	348	1392
MATMUL	4.1	69.6	21.7	78.8

Table 8.3: Timings in milliseconds for SUM and MATMUL for varying n . Timings were obtained on a 32 node partition of a 1K node CM-5, OS version 7.1.5, CMF version CM5 SPARC 1.2. Timings were provided by Thinking Machines Corporation engineer W. Weseloh. (Much faster hardware units should soon be available for the CM-5.)

8.2 n -body simulation

(Note: Some of the material in this section originally appeared in a joint paper with L. Daemen and J. Gubernatis which appeared in *Journal of Computational Physics*, **115**(2) December, 1994 [717].)

Simulating the properties of n particles, mutually interacting through a pairwise force, is one of the oldest problems in computational physics, and have been implemented on electronic computers since 1957, when the UNIVAC simulated 32 particles at 300 interactions per hour.

These simulations can yield information inaccessible by other means and lead to insight and predictive behavior for a wide range of problems and properties. This section reports on several simple procedures that reduced by an order of magnitude the computation time of the implementation of such a simulation on the massively parallel CM-200. The design of these procedures illustrates the theory outlined in previous sections, and no use of symmetry characteristics is needed.

In n -body simulations, the positions of the particles X_i are evolved in time by numerically integrating the equations of motion. The problem typically reduces to solving a system of first-order differential equations of the form:

$$\frac{dP_i}{dt} = G_i + H_i$$

where G_i is the net force on particle i due to its interaction with all other particles and P_i is the momentum of particle i . This force is defined by $G_i = \sum_{j \neq i} F_{ij}$ with F_{ij} being the pairwise force between particles i and j and H_i being the net force on particle i due to all other interactions. Often, F_{ij} depends only on the distance between the particles, i.e., $F_{ij} = F(|X_i - X_j|)$. This force might be, for example, Coulomb's law or the Lennard-Jones interaction. The H_i may be, for example, external fields or random forces simulating contact with a heat bath. The equations of motion can be integrated by a variety of means; the computation bottleneck, however, is the computation of the G_i .

Two frequently used classes of techniques for computing G_i are tree methods and direct methods. The tree methods recursively decompose the system of particles into subsystems and express the interaction between the subsystems by a multipole expansion [348,350]. These methods are particularly suitable for large systems because their asymptotic complexity is proportional to $O(n \log n)$ for adaptive methods or proportional to $O(n)$ for non-adaptive methods. The direct method, on the other hand, simply sums the forces between all pairs of particles so its complexity is proportional to $O(n^2)$. However, the proportionality constant for the direct method is smaller than that for the multipole methods.

One of the first parallel n -body codes was described in an intriguing 1985 paper by Aplegate, Douglas, Gürsel, Hunter, Seitz and Sussman where they describe the construction of a "digital orrery." The digital orrery comprised $n = 10$ "planet" computers, each of which stored the force on a single planet. They were interconnected in a ring, which was circulated to compute the all-pairs interaction. The intended domain of study of the digital

orrery was the study of certain open problems in orbital mechanics [338,577,803,804]. Each planet computer was housed on its own board, and the ten boards were stored in a box of a cubic foot, drawing 150 watts of power. The attained rate of speed was comparable to high-performance parallel computers of the time, such as that of the ILIAC IV, on this problem: 10 MFlops [137].

Since that time, of course, parallel n -body simulation has become an active and important area of research. Much of it has focused on the harder problem of simulating tree codes. The seminal paper on Connection Machine simulations is Hillis and Barnes (1987) [366]. They give three algorithms: a massively parallel variant of the digital orrery, computing all the forces using $O(n^2)$ processors, and a hierarchical algorithm based on a tree embedding of the tree used in the Barnes-Hut tree algorithm [84], although an implementation is only sketched. Zhao and Johnsson have implemented the parallel multipole method [349,823] on the CM-2, using some aggressive coding tricks [824]. Sometimes the structure of the problem permits long-range forces to be ignored or otherwise simplified, and this can permit efficient implementation [95,519], also see [94,795].

The direct method is often used instead of the asymptotically more efficient multipole method because

1. The multipole expansion may be unknown.
2. The system is sufficiently small that the direct method is faster than the multipole method.

Other methods use domain-decomposition approaches and Monte-Carlo approaches, which are outside the scope of this section. However, note that some of the domain-decomposition methods neglect the higher-order force in the Lennard-Jones interaction, which is not possible for general forces.

In addition, at some sufficiently fine granularity in the decomposition in the multipole method, the direct method becomes faster because of the lower proportionality factor and, thus, becomes preferred. This crossover generally is true at the leaves in a multipole methods. Thus efficient implementation of the direct method is an essential aspect of any direct net force computation.

The parallel calculation of the net force has a computation part and a communication part. The computation part is mainly concerned with the calculation of F_{ij} . Its optimization, which normally involves the computation of some function of the distance between X_i and X_j is essentially architecture independent. On a parallel machine, once the distance between the particles is known, this force can often be computed with no inter-processor communication.

The optimization of the communication part of the direct solver is the main result of this section. It depends on the the structure of the direct solver. Two common methods are used: one is the “all-to-all broadcast method”;the other is what we will call the “Fortran 90” method.

The all-to-all broadcast method presumes that the locations X of the particles are distributed throughout the nodes of the architecture. A local copy Y of X is made; then for

each Y_i a Hamiltonian path through the processor network is computed. The location of each particle is then successively routed along its path which ensures that each particle location will eventually be transmitted through each processor. The force on a particle is computed simply by summing the pairwise interactions of all the particles that pass through the processor in which it is stored. For systems with a small number of particles, this method performs poorly on the CM-200 because of processor under-utilization.

The Fortran 90 method involves the use of standard Fortran 90 intrinsics, such as `SUM` and `SPREAD`. If A is a matrix, then `SUM(A, DIM = 2)` is the vector whose i th element is the sum of the i th row of A . If X is an n -dimensional vector, then `SPREAD(X, DIM = 2, NCOPIES = n)` is the $n \times n$ matrix A whose ij th element is X_i . Similarly, `SPREAD(X, DIM = 1, NCOPIES = n)` is the matrix whose ij th component is X_j . The Fortran 90 method for computing the net force on each particle in a one-dimensional system is shown in the following pseudo-code:

```
S1=SPREAD(X,DIM=1,NCOPIES=n)
S2=SPREAD(X,DIM=2,NCOPIES=n)
R=ABS(S2-S1)
F=FUNC(R)
G=SUM(F,DIM=2)
```

The ij th element of the matrix R is the distance between particles at positions X_i and X_j . This matrix is then used to obtain the forces F_{ij} from the user-defined function `FUNC`. Finally, `SUM` is used to sum the forces.

Although the Fortran 90 method has the advantage of portability to any platform, the communication functions `SUM` and `SPREAD` are fairly slow. Furthermore, the `SPREAD` and `SUM` syntax seems to lack a certain naturalness. The first optimization trick that we tried

was to replace the communication operations of the Fortran 90 method with inner and outer products. Since these products are basic vector (matrix) computational tools, we thought it was reasonable to assume that they would be well optimized on the CM-200.

The FORTRAN 90 formulation of the previous code is then:

```

K=1
S1=OPROD(K,X)
S2=OPROD(X,K)
R=ABS(S2-S1)
F=FUNC(R)
G=MATMUL(F,K)

```

The final optimization involved special purpose microcode but is based on the simple observation that the displacement $X_i - X_j$ between two particles can be interpreted as the outer sum of the vectors X and $-X$, which could be computed by trivially replacing the multiplication call in the outer-product routine with an addition call. Using this routine, we can replace the two outer-product calls in the code by a single outer-sum call, and our pseudo-code becomes

```

K=1
D=OSUM(X,-X)
R=ABS(D)
F=FUNC(R)
G=MATMUL(F,K)

```

Compared to the original pseudo-code, which was essentially the original Fortran 90 coding, the new code reduced the computation time by a factor of 10.

In conclusion, the direct net force computation in the n -body problem was formulated in

terms of fast primitives that are well-suited to the target architecture. The communication overhead of the direct n -body solver was reduced by one order of magnitude. The method consisted in replacing Fortran 90 intrinsics by inner- and outer-product functions. In one case, a routine which was a small modification to the library outer-product routine was made to convert it to an outer-sum routine. The technique was implemented and tested on a molecular dynamics problem geared toward the study of flux line dynamics in superconducting thin films.

8.3 Parallel prefix and an application from computational biology

This section discusses the parallelization of an application that arose from computational biology. After some preprocessing, it is possible to see that the problem is equivalent to computing a certain statistic on strings, namely, finding the number of substrings of contiguous 1s of length l , for $l = 1, \dots, n$. The associated matrix has the symmetry of a cyclic group, and can be factored into the product of two parallel prefix matrices.

The dynamics of water molecules around biomolecules have been studied a number of authors.²⁷

Techniques such as high resolution neutron diffraction, X-ray crystallography, and multidi-

²⁷ A more detailed presentation, particularly of the physical motivation, is contained in the joint work with Angel García [319, 718]; also compare [318, 739].

mensional NMR techniques have been used to study the characteristics of water-molecules around a protein [319].

Insight into the structures that are formed is often obtained by studying the amount of time an individual water molecule is near a particular protein site, or within the “hydration shell” of the protein site (atom). The hydration shell is simply a sphere of some fixed radius r .

The time-dependent behavior of water molecules is studied by means of the following function of time [675,676]. We let $P_{\alpha,j}(t) = 1$ if the j th water molecule is within the hydration shell of the protein site α at time t . We let $P'_{\alpha,j}(t, t') = 1$ if the j th water molecule was never outside the hydration shell of the protein site at α .

In the molecular dynamics simulation, the time variable is discretized and the positions of the water molecules are computed at times $t_1, t_2, t_3, \dots, t_n$, where n is the number of states in the simulation. The function $P_{\alpha,j}$ is thereby discretized into a binary sequence of length n .

It is necessary to compute for each i the number of binary subsequences of length i in the resulting sequence.

This is performed as follows. First, the vector \mathbf{v} of all elements of the sequence of length precisely i are computed, that is, that are not a member of any subsequence.

The parallel prefix, or SCAN, of a vector \mathbf{v} is defined to be the sequence of partial sums of \mathbf{v} [475]:

$$\begin{aligned}
(\mathbb{T}\mathbf{v})_0 &\equiv \mathbf{v}_0 + \mathbf{v}_1 + \cdots + \mathbf{v}_{n-2} + \mathbf{v}_{n-1} \\
(\mathbb{T}\mathbf{v})_1 &\equiv \mathbf{v}_1 + \cdots + \mathbf{v}_{n-2} + \mathbf{v}_{n-1} \\
(\mathbb{T}\mathbf{v})_i &\equiv \mathbf{v}_i + \mathbf{v}_{i+1} + \mathbf{v}_{i+2} + \cdots + \mathbf{v}_{n-1}, \quad (0 \leq i \leq n-1).
\end{aligned}$$

However,

$$\begin{aligned}
(\mathbb{T}\mathbf{v})_i &= (\mathbf{E}^0\mathbf{v})_i + (\mathbf{E}^1\mathbf{v})_i + \cdots + (\mathbf{E}^{n-i}\mathbf{v})_i \\
&= \left((\mathbf{E}^0 + \mathbf{E}^1 + \cdots + \mathbf{E}^{n-i}) \mathbf{v} \right)_i \\
&= \left(\sum_{j=0}^{n-1} \mathbf{E}^j \mathbf{v} \right)_i.
\end{aligned}$$

Hence,

$$\mathbb{T} = \sum_{j=0}^{n-1} \mathbf{E}^j.$$

Scans are often defined over general associative operators, and have been proposed as a fundamental primitive for parallel architectures [131, 132, 464]. Their matrix expression is implicit in [450] but this work is the first known to the author to make the matrix form explicit; parallel recurrence solvers were also considered in=citemeyer:parallel.

Note that although the scan matrix is not quite circulant, it can be padded by a factor of two to make it \mathfrak{C}_{2n} -circulant. This padding, which essentially is a way of introducing periodic boundary conditions, is similar to the trick used to make the string matrix circulant.

In order to derive an efficient algorithm for multiplication by \mathbb{T} it is necessary to factor \mathbb{T} , which is equivalent to factoring its above polynomial representation. Since $\mathbf{E}^n = \mathbf{0}$ we are

factoring in the ring $\frac{\mathbb{Z}[x]}{x^n}$ [492]. Assume $n = 2^k$.

$$\begin{aligned} 1 + \mathbf{E} &= 1 + \mathbf{E} \\ (1 + \mathbf{E})(1 + \mathbf{E}^2) &= 1 + \mathbf{E} + \mathbf{E}^2 + \mathbf{E}^3 \\ (1 + \mathbf{E})(1 + \mathbf{E}^2)(1 + \mathbf{E}^4) &= 1 + \mathbf{E} + \cdots + \mathbf{E}^6 + \mathbf{E}^7 \\ \prod_{j=0}^{k-1} (1 + \mathbf{E}^{2^j}) &= \sum_{j=0}^{2^k-1} \mathbf{E}^j. \end{aligned}$$

Therefore,

$$\begin{aligned} \prod_{j=0}^{\log_2 n-1} (I_n + \mathbf{E}^{2^j}) &= \sum_{j=0}^{n-1} \mathbf{E}^j \\ &= \mathbf{T}. \end{aligned}$$

On an n -node hypercube, left multiplication of a vector by \mathbf{E}^{2^j} is constant time, as is left multiplication by $I_n + \mathbf{E}^{2^j}$ [333,371]. Each term in the product is thus constant time, and there is logarithmic number of terms, so parallel prefix is logarithmic time. Of course, an $O(n \log n)$ algorithm is easy to derive directly [475] or by using, for instance, \mathfrak{C}_n -fast Fourier transforms [758].

The advantage of the factorization approach is that it applies in the case when scans are implemented as a primitive operation on the machine [131,365,742], as exemplified below.

By using certain simple parallel prefix operations the problem can be reduced to the following: given a vector \mathbf{v} , compute the vector $\mathbf{M}\mathbf{v}$, where $\mathbf{M}\mathbf{v}$ is defined so that

$$(\mathbf{M}\mathbf{v})_i = \mathbf{v}_i + 2\mathbf{v}_{i+1} + 3\mathbf{v}_{i+2} + \cdots + (n-i+1)\mathbf{v}_n, 1 \leq i \leq n$$

Let \mathbf{E} and \mathbf{T} be the previously defined shift and parallel prefix operators. Then \mathbf{M} is a linear transformation which satisfies

$$\mathbf{M} = \mathbf{E}^0 + 2\mathbf{E}^1 + 3\mathbf{E}^2 + \cdots + (n-2)\mathbf{E}^{n-1}.$$

This polynomial can be factored:

$$\left(\sum_{j=0}^n \mathbf{E}^j \right)^2 = \left(\sum_{j=0}^{n-1} (j+1)\mathbf{E}^j \right) + \sum_{k=n} a_k \mathbf{E}^k$$

The right hand term vanishes because $\mathbf{E}^n = 0$. Therefore,

$$\begin{aligned} \mathbf{M} &= \sum_{j=0}^{n-1} (j+1)\mathbf{E}^j \\ &= \left(\sum_{j=0}^{n-1} \mathbf{E}^j \right)^2 \\ &= \mathbf{T}^2. \end{aligned}$$

Hence, $\mathbf{M} = \mathbf{T}^2$ and can be computed by two iterated applications of parallel prefix.

By using the methodology of this paper, we were able to parallelize the application and perform in approximately 6 hours an analysis computation that would have taken approximately 800 hours of time on a vector minisupercomputer using the previous techniques. This allowed analyses to be performed on many more protein sites than had been possible. Detailed results are included in [718].

Chapter 9

Conclusion and future work

This thesis described techniques for the design of parallel programs that solve well-structured problems with inherent symmetry.

Part I demonstrated the reduction of such problems to generalized matrix multiplication by a group-equivariant matrix. Fast techniques for this multiplication were described, including factorization, orbit decomposition, and Fourier transforms over finite groups. Our algorithms entailed interaction between two symmetry groups: one arising at the software level from the problem's symmetry and the other arising at the hardware level from the processors' communication network.

Part II illustrated the applicability of our symmetry-exploitation techniques by presenting a series of case studies of the design and implementation of parallel programs.

First, a parallel program that solved chess endgames by factorization of an associated dihedral group-equivariant matrix was described. This code ran faster than previous serial

programs and discovered a number of results in its domain.

Second, parallel algorithms for Fourier transforms for finite groups were developed and preliminary parallel implementations for group transforms of dihedral and of symmetric groups were described. Applications in learning, vision, pattern recognition and statistics were proposed.

Third, parallel implementations solving several computational science problems were described, including the direct N-body problem, convolutions arising from molecular biology, and some communication primitives such as broadcast and reduce. Some of our implementations ran orders of magnitude faster than previous techniques, and were used in the investigation of various physical phenomena.

The next logical stage in the development of our paradigm is the implementation of software tools to support its application.

One way to approach an implementation of the ideas in this paper is by analogy with the work of Soicher on GRAPE [700]. GRAPE is a graph-manipulation package, in which each graph is associated with a subgroup of its automorphism group. We propose to extend this idea to general arrays; we propose to build a general BLAS-like package for the manipulation of arrays in which each array also comes with an associated group of invariances.

A complementary approach would be the implementation of our algorithms as part of the loop transformation phase of an optimizing parallelizing compiler [81,82,511]. Once a compiler detects a group invariance, it could call, for example, a group convolution algorithm.

Automatic implementation of group FFTs would present little difficulty, but finding good heuristics to factorize the matrices would probably not be feasible, so that user compiler-directives would be required.

It would also be interesting to explore generalizations of the tensor-product formulation to nonlinear operators. Suppose that \mathbf{A} and \mathbf{B} are arbitrary maps from m -tuples to m -tuples. It is natural to define $l \otimes \mathbf{A}$ to be the map from ml -tuples to ml -tuples obtained by running \mathbf{A} simultaneously on l contiguous length m segments. We can *define* $\mathbf{A} \otimes l$ by the Commutation Theorem to be

$$\mathbf{A} \otimes l \equiv \mathbf{P}_m^n (l \otimes \mathbf{A}) \mathbf{P}_l^n.$$

We can then define the tensor product of arbitrary operators on tuples by analogy with equation 4.4:

$$\mathbf{A} \otimes \mathbf{B} \equiv (\mathbf{A} \otimes l) (l_m \otimes \mathbf{B}).$$

Of course, it is unclear whether anything is gained by the added generality.

Finally, and more speculatively, the ubiquity and centrality of symmetry considerations in a number of disparate applications might argue for the utility of a symmetry-theoretic classification of computational problems. This line of speculation would soon lead into category-theoretic considerations, insofar as one views a functor as a kind of generalization of a group action (since a group action is simply a functor from a one-object category in which each morphism is invertible).

Appendix A

List of symbols

General

\mathbb{R}^k	k -dimensional Euclidean space
\mathbb{C}	The complex numbers
A, B, M	Matrices
F_n	Discrete Fourier transform of degree n
I_n	$n \times n$ identity matrix
P_n^m	Stride permutation
K_k	Vector of k ones
$\mathcal{G}, \mathfrak{H}$	Finite groups
e	Identity element of a group
$\mathfrak{H} < \mathcal{G}$	\mathfrak{H} is a subgroup of \mathcal{G}
$ \mathcal{G} $	Number of elements in \mathcal{G}
X, Y	Sets
\otimes, \bigotimes^k	Tensor product, k th tensor power
\oplus	Direct sum of vector spaces or matrices
A^T	Transpose of A
$\text{diag}(\mathbf{v})$	Diagonal matrix with \mathbf{v} 's elements on diagonal
Sym^j	Symmetric power
\mathcal{D}_4	The dihedral group of order 8
r	90° rotation element of \mathcal{D}_4
f	Flipping operation, element of \mathcal{D}_4
\mathcal{C}_n	The cyclic group of order n
\mathcal{S}_k	The symmetric group of order k
\mathcal{GF}_2	The finite field of order 2
X/\mathcal{G}	Orbit space for \mathcal{G} -action on X
$\mathcal{V}_n, \mathcal{W}_m$	Vector space of dimensions n, m
\mathfrak{F}	A field

\mathbb{Z}_k	The integers modulo k
\mathfrak{M}_m^n	The set of $m \times n$ matrices
$\mathbb{I}_{\mathbf{P},i}$	Unmove operator for i th piece according to the rules of p
i, j, k, m, n, l	Positive integers.
$\mathfrak{g}, \mathfrak{h}$	Group elements
\mathbf{v}, \mathbf{w}	Vectors
$f = O(\text{polylog}(g))$	f is at most polylogarithmic in g

Chapter 6

$\mathbb{I}^{\text{White}}, \mathbb{I}^{\text{Black}}$	Unmove predecessor functions
p	Generic chess piece type
$\mathbb{I}_{\mathbf{P},s}$	Unmove operator of p on the s th coordinate
\mathfrak{C}	$\mathfrak{V}_8 \otimes \mathfrak{V}_8$
\mathfrak{B}	$\otimes^k \mathfrak{C}$
$\int_{\mathfrak{G}} x$	$\sum_{\mathfrak{g} \in \mathfrak{G}} \mathfrak{g}^x$
s	Element of \mathfrak{S}_k

Chapter 7

$F_{\mathfrak{G}}$	Fourier matrix of \mathfrak{G}
ρ, η	Group representations of degrees d_ρ, d_η
$\rho \downarrow \mathfrak{H}$	Restriction of ρ to \mathfrak{H}
γ	Adapted diameter of a group
S	Strong generating set
G	An acyclic arithmetic circuit
$D(G)$	Depth of G
$ G $	Size of G
$\mathcal{M}(\mathfrak{G}, \mathfrak{H})$	Maximum multiplicity of irreducible rep. of \mathfrak{H} in restriction of one in \mathfrak{G}
\mathcal{R}	Complete set of adapted irreducible representations of \mathfrak{G}
p	Pattern string
t	Text string
Σ	Finite alphabet
$\Sigma_{\text{T}}, \Sigma_{\text{P}}$	Text and pattern alphabets
σ	Character
s	String
ϕ	Don't care character
\equiv	Exact-match (with don't cares)
\sim	Subset matching (Abrahamson)
\boxplus	Generalized combining operator



Generalized character-comparison

Matrix multiplication with generalized $+$, \times

Chapter 8

EOSHIFT
 CSHIFT
 SUM
 OSUM
 SPREAD
 SCAN
 MATMUL

End-off shift function (E)
 Circular shift function
 $+$ -reduction function
 Outer sum function
 Broadcast function
 Parallel prefix function (T)
 Matrix multiplication function

Bibliography

- [1] SYED KAMAL ABDALI AND BENJAMIN DAVID SAUNDERS. Transitive closure and related semiring properties via elimination. *Theoretical Computer Science*, **40**(2–3):257–274, 1985.
- [2] KARL RAYMOND ABRAHAMSON. Generalized string matching. *SIAM Journal on Computing*, **16**(6):1039–1051, December 1987.
- [3] D. ADAMS. Cray T3D system architecture overview. Technical report, Cray Research Inc., September 1993. Revision 1.C.
- [4] GEORGE B. ADAMS, E.C. BRONSON, THOMAS L. CASAVANT, L.H. JAMIESON, AND RAY A. KAMIN. Experiments with parallel fast Fourier transforms. In Magdy A. Bayoumi, editor, *Parallel Algorithms and Architectures for DSP applications*, volume 149 of *The Kluwer International Series in Engineering and Computer Science. VLSI, Computer Architecture, and Digital Signal Processing*, pages 49–75. Kluwer Academic, Dordrecht, Netherlands, 1991.
- [5] JEANNE C. ADAMS, WALTER S. BRAINERD, JEANNE T. MARTIN, BRIAN T. SMITH, AND JERROLD L. WAGENER. *Fortran 90 Handbook: Complete ANSI/ISO Reference*. McGraw-Hill Book Co., New York, 1992.
- [6] [أَلْعَدْلِي]. كِتَاب الشَطْرَنَج, n.d. (al-‘Adlī, Book of Chess. Photographic copy of Arabic manuscript).
- [7] RAMESH C. AGARWAL, FRAN GOERTZEL GUSTAVSON, AND MOHAMMED ZUBAIR. An efficient parallel algorithm for the 3-D FFT NAS parallel benchmark. In *Proceedings of the Scalable High-Performance Computing Conference*, pages 129–133, Knoxville, TN, 23–25 May 1994. IEEE Computer Society Press, Los Alamitos, CA, 1994.
- [8] DHARMA P. AGRAWAL. Graph theoretical analysis and design of multistage interconnection networks. *IEEE Transactions on Computers*, **C-32**:637–648, July 1983.
- [9] ALFRED VAINO AHO AND MARGARET JOHN CORASICK. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, **18**(6):333–340, June 1975.
- [10] ALFRED VAINO AHO, DANIEL S. HIRSCHBERG, AND JEFFREY DAVID ULLMAN. Bounds on the complexity of the longest common subsequence problem. *Journal of the Association for Computing Machinery*, **23**(1):1–12, January 1976.
- [11] ALFRED VAINO AHO, JOHN EDWARD HOPCROFT, AND JEFFREY DAVID ULLMAN. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley Publishing Company, Reading, MA, 1974.

- [12] W. AHRENS. Mathematische Schachfragen [mathematical chess questions]. 1902 throughout.
- [13] MIKLOS AJTAI, J. KOMLÓS, AND E. SZEMERÉDI. An $O(n \log n)$ sorting network. In *Proceedings of the Fifteenth Annual Symposium on Theory of Computing*, pages 1–9, Boston, MA, 25–27 April 1983. Association for Computing Machinery, New York, 1983.
- [14] SHELDON B. AKERS AND BALAKRISHNAN KRISHNAMURTHY. Group graphs as interconnection networks. In *The Fourteenth International Conference on Fault-Tolerant Computing: Digest of Papers: FTCS-14*, pages 424–427, Kissimmee, FL, 20–22 June 1984. IEEE Computer Society Press, Silver Spring, MD, 1984.
- [15] SHELDON B. AKERS AND BALAKRISHNAN KRISHNAMURTHY. On group graphs and their fault tolerance. *IEEE Transactions on Computers*, **C-36**(7):885–888, July 1987.
- [16] SELIM GEORGE AKL, DAVID T. BARNARD, AND RALPH J. DORAN. Simulation and analysis in deriving time and storage requirements for a parallel alpha-beta pruning algorithm. In *Proceedings of the Ninth IEEE International Conference on Parallel Processing*, pages 231–234, Columbus, OH, 26–29 August 1980. IEEE Computer Society Press, Los Alamitos, CA, 1980.
- [17] SELIM GEORGE AKL, DAVID T. BARNARD, AND RALPH J. DORAN. Design, analysis and implementation of a parallel tree search algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-4**(2):192–203, March 1982.
- [18] TATSUYA AKUTSU. A linear time pattern matching algorithm between a string and a tree. In Alberto Apostolico, Maxime Crochemore, Zvi Galil, and Udi Manber, editors, *Combinatorial Pattern Matching: 4th Annual Symposium. Proceedings*, volume 684 of *Lecture Notes in Computer Science*, pages 1–10, Padova, Italy, 2–4 June 1993. Springer-Verlag, Berlin, 1993.
- [19] EUGENE LEO ALLGOWER, KLAUS BÖHMER, KURT GEORG, AND RICK MIRANDA. Exploiting symmetry in boundary element methods. *SIAM Journal on Numerical Analysis*, **29**(2):534–552, April 1992.
- [20] EUGENE LEO ALLGOWER, KLAUS BÖHMER, KURT GEORG, AND RICK MIRANDA. Exploiting symmetry in boundary element methods. *SIAM Journal on Numerical Analysis*, **29**:534–552, 1992.
- [21] Eugene Leo Allgower, Klaus Böhmer, and Martin Golubitsky, editors. *Bifurcation and Symmetry: Cross Influence Between Mathematics and Applications*, volume 104 of *International Series of Numerical Mathematics*. Birkhäuser Verlag, Basel, Germany, 1992.
- [22] EUGENE LEO ALLGOWER, KURT GEORG, AND RICK MIRANDA. Exploiting permutation symmetries with fixed points in linear equations. In *Proceedings of the 22nd AMS-SIAM Summer Seminar on Applied Mathematics: Exploiting Symmetry in Applied and Numerical Analysis*, volume 29 of *Lectures in Applied Mathematics*, pages 23–36, Colorado State University, 26 July–1 August 1992. American Mathematical Society, Providence, Rhode Island, 1992.
- [23] INGO ALTHÖFER. A parallel game tree search algorithm with linear speedup. *Journal of Algorithms*, **15**(2):175–198, September 1993.
- [24] INGO ALTHÖFER AND BERNHARD WALTER. Weak zugzwang: statistics on some chess endgames. *International Computer Chess Association Journal*, **17**(2):75–77, June 1994.

- [25] SAMAN P. AMARASINGHE, JENNIFER M. ANDERSON, MONICA S. LAM, AND AMY W. LIM. An overview of a compiler for scalable parallel machines. In Utpal Banerjee, David Gelernter, Alexandru Nicolau, and David A. Padua, editors, *Languages and Compilers for Parallel Computing: Sixth International Workshop: Proceedings*, volume 768 of *Lecture Notes in Computer Science*, pages 253–272, Portland, OR, 12–14 August 1993. Springer-Verlag, Berlin/Heidelberg, 1994.
- [26] GENE MYRON AMDAHL. Limits of expectation. *International Journal of Supercomputer Applications*, **2**(1):88–94, Spring 1988.
- [27] [FRIEDRICH LUDWIG AMELUNG?]. Baltische Schachpartien aus den Jahren 1858 bis 1892, Partie 114 [Baltic chess games from the years 1892 to 1858, Game 114]. *Baltische Schachblätter*, **4**:266–267, 1893. Score of consultation Theodor Molien and A. Hasselblatt v. Friedrich Amelung.
- [28] FRIEDRICH LUDWIG AMELUNG. Das Endspiel von zwei Offizieren gegen einen Springer [The endgame of two pieces against a rook]. *Baltische Schachblätter*, **4**:290–297, 1893.
- [29] FRIEDRICH LUDWIG AMELUNG. Auszüge aus den Briefen von A. Ascharin an F. Amelung [Excerpts from the letters of A. Ascharin to F. Amelung]. *Baltische Schachblätter*, **5**:12–38, 1898.
- [30] FRIEDRICH LUDWIG AMELUNG. Das Endspiel von Thurm gegen Springer (Fortsetzung) [The endgame of rook against knight (continuation)]. *Deutsche Schachzeitung*, **55**(2):37–41, February 1900.
- [31] FRIEDRICH LUDWIG AMELUNG. Das Endspiel von Thurm gegen Springer (Fortsetzung) [The endgame of rook against knight (continuation)]. *Deutsche Schachzeitung*, **55**(4):101–105, April 1900.
- [32] FRIEDRICH LUDWIG AMELUNG. Das Endspiel von Thurm gegen Springer (Fortsetzung) [The endgame of rook against knight (continuation)]. *Deutsche Schachzeitung*, **55**(5):134–138, May 1900.
- [33] FRIEDRICH LUDWIG AMELUNG. Das Endspiel von Thurm gegen Springer (Fortsetzung) [The endgame of rook against knight (continuation)]. *Deutsche Schachzeitung*, **55**(7):198–202, July 1900.
- [34] FRIEDRICH LUDWIG AMELUNG. Das Endspiel von Thurm gegen Springer (Schluss) [The endgame of rook against knight (conclusion)]. *Deutsche Schachzeitung*, **55**(9):261–266, September 1900.
- [35] FRIEDRICH LUDWIG AMELUNG. Das Endspiel von Thurm gegen Springer [The endgame of rook against knight]. *Deutsche Schachzeitung*, **55**(1), January 1900.
- [36] FRIEDRICH LUDWIG AMELUNG. Das Endspiel von Thurm und Springer gegen die Dame [The endgame of rook and knight against queen]. *Deutsche Schachzeitung*, **56**(7):193–197, July 1901.
- [37] FRIEDRICH LUDWIG AMELUNG. Das Endspiel von Thurm und Springer gegen die Dame [The endgame of rook and knight against queen]. *Deutsche Schachzeitung*, **56**(8):225–229, August 1901.

- [38] FRIEDRICH LUDWIG AMELUNG. Die Endspiele mit Qualitätsvorteil, insbesondere das Endspiel von Thurm und Läufer gegen Läufer und Springer [The endgames with exchange advantage, especially the endgame of rook and bishop against bishop and knight]. *Deutsche Schachzeitung*, **57**(9):265–268, September 1902.
- [39] FRIEDRICH LUDWIG AMELUNG. Die Endspiele mit Qualitätsvorteil, insbesondere das Endspiel von Thurm und Läufer gegen Läufer und Springer (Fortsetzung) [The endgames with exchange advantage, especially the endgame of rook and bishop against bishop and knight (continuation)]. *Deutsche Schachzeitung*, **57**(10):297–300, October 1902.
- [40] FRIEDRICH LUDWIG AMELUNG. Die Endspiele mit Qualitätsvorteil, insbesondere das Endspiel von Thurm und Läufer gegen Läufer und Springer (Schluss) [The endgames with exchange advantage, especially the endgame of rook and bishop against bishop and knight (conclusion)]. *Deutsche Schachzeitung*, **57**(11):330–332, November 1902.
- [41] FRIEDRICH LUDWIG AMELUNG. Das Endspiel von Turm und Läufer gegen zwei Springer [The endgame of rook and bishop against two knights]. *Düna-Zeitung (Feuilleton-Beilage: Für Haus und Familie)*, **40**:52–53, 16–29 February 1908.
- [42] FRIEDRICH LUDWIG AMELUNG. Endspiel 1028. *Deutsches Wochenschach*, **24**(14):130, 5 April 1908.
- [43] FRIEDRICH LUDWIG AMELUNG. Lösungspreis-endspiel Nr. 178 [Solution prize-endgame No. 178]. *Düna-Zeitung (Feuilleton-Beilage: Für Haus und Familie)*, **63**:87, 15–28 March 1908.
- [44] FRIEDRICH LUDWIG AMELUNG. Lösung des Preisendspiels Nr. 178 [Solution of prize endgame No. 178]. *Düna-Zeitung (Feuilleton-Beilage: Für Haus und Familie)*, **13**:20–21, 17–30 January 1909.
- [45] [DÜNA-ZEITUNG]. Friedrich Ludwg Amelung. *Düna-Zeitung (Feuilleton-Beilage: Für Haus und Familie)*, **70**:76–78, 28 March–10 April 1909.
- [46] AMIHOOD AMIR AND GARY BENSON. Two-dimensional periodicity and its applications. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 440–452, Orlando, FL, 27–29 January 1992. ACM Press, New York, 1992.
- [47] AMIHOOD AMIR, GARY BENSON, AND MARTIN FARACH. Alphabet independent two dimensional matching. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 59–67, Victoria, B.C., Canada, 4–6 May 1992. ACM Press, New York, 1992.
- [48] AMIHOOD AMIR AND GAD M. LANDAU. Fast parallel and serial multidimensional approximate array matching. *Theoretical Computer Science*, **81**(1):97–115, 22 April 1991.
- [49] MYOUNG AN, JAMES WILLIAM COOLEY, AND RICHARD TOLIMIERI. Factorization method for crystallographic Fourier transforms. *Advances in Applied Mathematics*, **11**(3):358–371, September 1990.
- [50] MYOUNG AN, CHAO LU, E. PRINCE, AND RICHARD TOLIMIERI. Fast Fourier transforms for space groups containing rotation axes of order three and higher. *Acta Crystallographica*, **A48**(Part 3):346–349, May 1992.
- [51] THOMAS ANANTHARAMAN, MURRAY S. CAMPBELL, AND FENG-HSIUNG HSU. Singular extensions: adding selectivity to brute force searching. In *Proceedings: 1988 Spring Symposium Series: Computer Game Playing*, pages 8–13, Stanford University, Stanford, CA, 22–24 March 1988. American Association for Artificial Intelligence.

- [52] JAMES P. ANDERSON, SAMUEL A. HOFFMAN, JOSEPH SHIFMAN, AND ROBERT J. WILLIAMS. D825: A multiple-computer system for command and control. In *Proceedings of the 1962 Fall Joint Computer Conference*, volume 22 of *AFIPS Conference Proceedings*, pages 86–96, 1962.
- [53] H.C. ANDREWS AND J. KANE. Kronecker matrices, computer implementations, and generalized spectra. *Journal of the Association for Computing Machinery*, **17**(2):260–268, April 1970.
- [54] FRED S. ANNEXSTEIN, MARC BAUMSLAG, AND ARNOLD LEONARD ROSENBERG. Group action graphs and parallel architectures. Technical Report COINS Technical Report 87-133, University of Massachusetts, 1987.
- [55] FRED S. ANNEXSTEIN AND MARC BAUMSLAG. Hamiltonian circuits in Cayley digraphs. Technical Report COINS 88-40, University of Massachusetts Computer and Information Science Department, Amherst, MA, 20 April 1988.
- [56] FRED S. ANNEXSTEIN, MARC BAUMSLAG, AND ARNOLD LEONARD ROSENBERG. Group action graphs and parallel architectures. *SIAM Journal on Computing*, **19**(3):544–569, June 1990.
- [57] Alberto Apostolico, Maxime Crochemore, and Zvi Galil, editors. *Combinatorial Pattern Matching: 4th Annual Symposium*, volume 684 of *Lecture Notes in Computer Science*, Padova, Italy, 2–4 June 1993. Springer-Verlag, Berlin/New York, 1993.
- [58] Alberto Apostolico, Maxime Crochemore, Zvi Galil, and Udi Mandber, editors. *Combinatorial Pattern Matching: Third Annual Symposium*, volume 644 of *Lecture Notes in Computer Science*, Tucson, AZ, 29 April–1 May 1992. Springer-Verlag, Berlin/New York, 1992.
- [59] ALBERTO APOSTOLICO, C. ILIOPOULOS, GAD M. LANDAU, BARUCH SCHIEBER, AND UZI VISHKIN. Parallel construction of a suffix tree with applications. *Algorithmica*, **3**(3):347–365, 1988.
- [60] ALBERTO APOSTOLICO AND FRANCO P. PREPARATA. Optimal off-line detection of repetitions in a string. *Theoretical Computer Science*, **22**(2–3):297–315, 1983.
- [61] ANDREW W. APPEL. An efficient program for many-body simulation. *SIAM Journal on Scientific and Statistical Computing*, **6**(1):85–103, January 1985.
- [62] JAMES H. APPEGATE, MICHAEL R. DOUGLAS, YEKTA GÜRSEL, PETER HUNTER, CHARLES LEWIS SEITZ, AND GERALD JAY SUSSMAN. A digital orrery. *IEEE Transactions on Computers*, **C-34**(9):822–831, September 1985.
- [63] V. L. ARLAZAROV AND A. L. FUTER. Computer analysis of a rook endgame. In J. E. Hayes, Donald Michie, L. J. Mikulich, and Ellis Horwood, editors, *Machine Intelligence 9*. Ellis Horwood Ltd., Chichester, England, 1979.
- [64] MICHAEL ASCHBACHER. *Finite Group Theory*, volume 10 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge/New York, 1986.
- [65] MICHAEL D. ATKINSON. The complexity of group algebra computations. *Theoretical Computer Science*, **5**(2):205–209, 1977.
- [66] LOUIS AUSLANDER AND RICHARD TOLIMIERI. Ring structure and the Fourier transform. *Mathematical Intelligencer*, **7**(3):49–52, 54, 1985.

- [67] LOUIS AUSLANDER AND RICHARD TOLIMIERI. Ambiguity functions and group representations. In *Group Representations, Ergodic Theory, Operator Algebras, and Mathematical Physics: Proceedings of a Conference in Honor of George W. Mackey (21–23 May 1984; Berkeley, CA)*, volume 6 of *Mathematical Sciences Research Institute Publications*, pages 1–10. Springer-Verlag, New York, 1987.
- [68] Юрий Львович Авербах [YURI LVOVICH AVERBAKH]. Шахматные Окончания; Ферзевые [Chess endings: queen]. Физкультура и спорт, Москва, 2nd edition, 1982.
- [69] AMIR AVERBUCH, ERAN GABBER, BOAZ GORDISSKY, AND YOAV MEDAN. A parallel FFT on a MIMD machine. *Parallel Computing*, **15**(1–3):61–74, September 1990.
- [70] LÁSZLÓ BABAI, D.YU. GRIGORIEV, AND D.M. MOUNT. Isomorphism of graphs with bounded eigenvalue multiplicity. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 36–41, San Francisco, CA, 5–7 May 1982. ACM Press, New York, 1982.
- [71] LÁSZLÓ BABAI, EUGENE MICHAEL LUKS, AND ÁKOS SERESS. Permutation groups in NC. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 409–420, New York, 25–27 May 1987. ACM Press, New York, 1987.
- [72] LÁSZLÓ BABAI, EUGENE MICHAEL LUKS, AND ÁKOS SERESS. Fast management of permutation groups. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 272–282, White Plains, NY, 24–26 October 1988. IEEE, Washington, DC, 1988.
- [73] LÁSZLÓ BABAI AND LAJOS RÓNYAI. Computing irreducible representations of finite groups. *Mathematics of Computation*, **55**(192):705–722, October 1990.
- [74] CHARLES BABBAGE. Games of skill. In *Passages From the Life of a Philosopher*, pages 465–471. Longman, Green, Longman, Roberts, and Green, London, 1864.
- [75] ROLAND C. BACKHOUSE AND B. A. CARRÉ. Regular algebra applied to path-finding problems. *Journal of the Institute of Mathematics and its Applications*, **15**(2):161–186, April 1975.
- [76] ROLAND C. BACKHOUSE, J.P.H.W. VAN DEN EIJNDE, AND A.J.M. VAN GASTEREN. Calculating path algorithms. *Science of Computer Programming*, **22**(1–2):3–19, April 1994.
- [77] JOHN W. BACKUS. Can programming be liberated from the von Neumann style? A functional style and its algebra of programs (1977 ACM Turing Award Lecture). *Communications of the ACM*, **21**(8):613–641, August 1978.
- [78] DAVID H. BAILEY, K. LEE, AND H.D. SIMON. Using Strassen’s algorithm to accelerate the solution of linear systems. *Journal of Supercomputing*, **4**(4):357–371, January 1991.
- [79] THEODORE PAUL BAKER. A technique for extending rapid exact-match string matching to arrays of more than one dimension. *SIAM Journal on Computing*, **7**(4):533–541, November 1978.
- [80] HENRI E. BAL AND ROBERT VAN RENESSE. A summary of parallel alpha-beta search results. *International Computer Chess Association Journal*, **9**(3):146–151, September 1986.
- [81] UTPAL BANERJEE. *Loop Transformations for Restructuring Compilers: the Foundations*. Kluwer Academic, Boston, MA, 1993.
- [82] UTPAL BANERJEE. *Loop Parallelization*, volume 2 of *Loop Transformations for Restructuring Compilers*. Kluwer Academic, Boston, MA, 1994.

- [83] C.R. BANGER. *Construction of Multidimensional Arrays as Categorical Data Types*. PhD thesis, Queen's University, Kingston, Canada, 1994.
- [84] JOSH EDWARD BARNES AND PIET HUT. A hierarchical $O(N \log N)$ force calculation algorithm. *Nature*, **324**:446–449, 4 December 1986.
- [85] GERHARD BARTH. An alternative for the implementation of the Knuth-Morris-Pratt algorithm. *Information Processing Letters*, **13**(4,5), End 1981.
- [86] J.G. BASHMAKOVA. *Molin, Fedor Eduardovich*, volume 3, pages 1739–1740. Charles Scribner's Sons, New York, 1991.
- [87] KENNETH EDWARD BATCHER. Design of a massively parallel processor. *IEEE Transactions on Computers*, **C-29**(9):836–840, September 1980.
- [88] F. BAUDE AND DAVID BENSON SKILLICORN. Vers de la programmation parallèle structurée fondée sur la théorie des catégories. *Techniques et Sciences de l'Informatique*, **13**(4):525–537, 1994.
- [89] ULRICH BAUM. Existence and efficient constructions of fast fourier transforms on supersolvable groups. *Computational Complexity*, **1**(3):235–256, 1991.
- [90] ULRICH BAUM AND MICHAEL CLAUSEN. Some lower and upper complexity bounds for generalized Fourier transforms and their inverses. *SIAM Journal on Computing*, **20**(3):451–459, June 1991.
- [91] ULRICH BAUM, MICHAEL CLAUSEN, AND B. TIETZ. Improved upper complexity bounds for the discrete Fourier transform. *Applicable Algebra in Engineering, Communication and Computing*, **2**(1):35–43, 1991.
- [92] GILBERT BAUMSLAG. *Topics in Combinatorial Group Theory*. Lectures in Mathematics. Birkhäuser, Basel, Boston, Berlin, 1993.
- [93] MARC BAUMSLAG AND ARNOLD LEONARD ROSENBERG. Processor-time tradeoffs for Cayley graph interconnection networks. In *The Sixth Distributed Memory Computing Conference Proceedings*, pages 630–636, Portland, OR, 28 April–1 May 1991. IEEE Computer Society Press, Los Alamitos, CA, 1991.
- [94] DAVID M. BEAZLEY AND PETER S. LOMDAHL. Message-passing multi-cell molecular dynamics on the Connection Machine 5. *Parallel Computing*, **20**(2):173–195, February 1994.
- [95] DAVID M. BEAZLEY, PETER S. LOMDAHL, P. TAMAYO, AND N. GRØNBECH-JENSEN. 50 gigaflops molecular dynamics on the connection machine. In Howard Jay Siegel, editor, *Proceedings of the Eighth International Parallel Processing Symposium*, Cancun, Mexico, 26–29 April 1994. IEEE Computer Society Press, Los Alamitos, CA, 1994.
- [96] [CARL BEHTING AND PAUL KERKOVIVUS?]. Das zweite Baltische Schachturnier [The second Baltic chess tournament]. *Baltische Schachblätter*, **9**:1–24, 1902.
- [97] ALEX G. BELL. Torres y Quevedo. In *The Machine Plays Chess?*, Pergamon Chess Series, chapter 2, pages 8–11. Pergamon Press, Oxford, 1978.
- [98] RICHARD ERNEST BELLMAN. On a new iterative algorithm for finding the solutions of games and linear programming problems. Technical Report P-473, The RAND Corporation, U.S. Air Force Project RAND, Santa Monica, CA, 1 June 1954.

- [99] RICHARD ERNEST BELLMAN. The theory of games. Technical Report P-1062, The RAND Corporation, Santa Monica, CA, 15 April 1957.
- [100] RICHARD ERNEST BELLMAN. On the reduction of dimensionality for classes of dynamic programming processes. Technical report, The RAND Corporation, Santa Monica, CA, 7 March 1961.
- [101] RICHARD ERNEST BELLMAN. On the application of dynamic programming to the determination of optimal play in chess and checkers. *Proceedings of the National Academy of Sciences of the United States of America*, **53**(2):244–246, February 1965.
- [102] VACLAV EDVARD BENEŠ. Optimal rearrangeable multistage connecting networks. *Bell System Technical Journal*, **43**(4, Part 2):1641–1656, July 1964.
- [103] VACLAV EDVARD BENEŠ. Permutation groups, complexes, and rearrangeable connecting networks. *Bell System Technical Journal*, **43**(4, Part 2):1619–1640, July 1964.
- [104] VACLAV EDVARD BENEŠ. *Mathematical theory of connecting networks and telephone traffic*, volume 17 of *Mathematics in Science and Engineering*. Academic Press, New York, 1965.
- [105] A.VAN BERGEN. An ultimate look at the KPK data base. *International Computer Chess Association Journal*, **8**(4):216–218, December 1985.
- [106] JOHANN BERGER. Der Turm und ein leichter Offizier gegen zwei leichte Offiziere [Rook and minor piece against two minor pieces]. In *Theorie und Praxis der Endspiele: Ein Handbuch für Schachfreunde*, pages 167–169. Veit and comp., Leipzig, 1890.
- [107] JOHANN BERGER. *Theorie und Praxis der Endspiele: Ein Handbuch für Schachfreunde* [Theory and practice of the endgame: a handbook for chessplayers]. Vereinigung Wissenschaftlicher Verleger Walter de Gruyter & Co., Berlin, Leipzig, 2nd edition, 1922.
- [108] JOHANN BERGER. *Theorie und Praxis der Endspiele: ein Handbuch für Schachfreunde* [Theory and practice of the endgame: a handbook for chessplayers]. Olms, Zurich, 1981.
- [109] KLAUS BERKLING. Arrays and the lambda calculus. In Lenore M. Restifo Mullin et. al., editor, *Arrays, Functional Languages and Parallel Systems*, pages 1–17. Kluwer Academic Publishers, 1991.
- [110] ELWYN RALPH BERLEKAMP, JOHN HORTON CONWAY, AND RICHARD K. GUY. *Winning Ways For Your Mathematical Plays*. Volume 1: *Games in General*. Academic Press, London and New York, 1982.
- [111] HANS JACK BERLINER. Backgammon computer program beats world champion. *Artificial Intelligence*, **14**(2):205–220, September 1980.
- [112] HANS JACK BERLINER AND MURRAY S. CAMPBELL. Using chunking to solve chess pawn endgames. *Artificial Intelligence*, **23**(1):97–120, May 1984.
- [113] HANS JACK BERLINER AND WILLIAM HENRY CARL EBELING. The SUPREM architecture: a new intelligent paradigm. *Artificial Intelligence*, **28**(1):3–8, February 1986.
- [114] ROBERT BERNECKY. Compiling APL. In Lenore M. Restifo Mullin et. al., editor, *Arrays, functional languages and parallel systems*, pages 19–33. Kluwer Academic Publishers, Boston, MA, 1991.
- [115] J. BERNTSEN. Communication efficient matrix multiplication on hypercubes. *Parallel Computing*, **12**(3):335–342, December 1989.

- [116] DMITRI P. BERTSEKAS. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Englewood Cliffs, NJ, 1987.
- [117] DMITRI P. BERTSEKAS AND JOHN N. TSISIKLIS. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall International, London, 1989.
- [118] THOMAS BETH. *Verfahren der schnellen Fourier-Transformation: die allgemeine diskrete Fourier-Transformation: ihre algebraische Beschreibung, Komplexität und Implementierung* [Methods of fast Fourier transforms: the general discrete Fourier transform: its algebraic description, complexity and implementation], volume 61 of *Leitfäden der angewandten Mathematik und Mechanik, Teubner Studienbücher: Informatik*. B.G. Teubner, Stuttgart, 1984.
- [119] THOMAS BETH. On the computational complexity of the general discrete Fourier transform. *Theoretical Computer Science*, **51**(3):331–339, 1987.
- [120] THOMAS BETH. Generalized Fourier transforms. In Rainer Janßen, editor, *Trends in Computer Algebra: International Symposium Bad Neuenahr, May 19–21, 1987: Proceedings*, volume 296 of *Lecture Notes in Computer Science*, pages 92–118. Springer-Verlag, Berlin, 1988.
- [121] SAID BETTAYEB, ZEVI MILLER, AND HAL SUDBOROUGH. Embedding k -D meshes into optimum hypercubes with dilation $2k - 1$ (extended abstract). In Michel Cosnard, Afonso Ferreira, and Joseph Peters, editors, *Proceedings of the First Canada–France Conference on Parallel and Distributed Computing: Theory and Practice*, volume 805 of *Lecture Notes in Computer Science*, pages 73–80, Montréal, Canada, 19–21 May 1994. Springer-Verlag, Berlin, 1994.
- [122] GYAN BHANOT AND SRIKANTH SASTRY. Solving the Ising model on a $5 \times 5 \times 4$ lattice using the Connection Machine. Technical Report TMC-43, Thinking Machines Corporation, Cambridge, MA, November 1989.
- [123] SANDEEP NAUTAM
BHATT, F. CHUNG, FRANK THOMSON LEIGHTON, AND ARNOLD LEONARD ROSENBERG. Optimal simulation of tree machines. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 274–282, Toronto, Ontario, 27–29 October 1986. IEEE Computer Society Press, Washington, DC / Los Angeles, CA, 1986.
- [124] FREDERIC BIEN. Constructions of telephone networks by group representations. *Notices of the American Mathematical Association*, **36**(1):5–22, January 1989.
- [125] NORMAN BIGGS. *Algebraic Graph Theory*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, England, 2nd edition, 1993.
- [126] RICHARD S. BIRD. Two dimensional pattern matching. *Information Processing Letters*, **6**(5):168–170, October 1977.
- [127] RICHARD S. BIRD. An introduction to the theory of lists. In Manfred Broy, editor, *Proceedings of the NATO Advanced Study Institute on Logic of Programming and Calculi of Discrete Design*, volume 36 of *NATO Advanced Science Institutes Series, Series F: Computer and Systems Sciences*. Springer-Verlag, Berlin, 1987, Marktoberdorf, Germany, 29 July–10 August 1986.
- [128] RICHARD S. BIRD. Algebraic identities for program calculation. *Computer Journal*, **32**(2):122–126, April 1989.

- [129] RICHARD S. BIRD AND OEGE DE MOOR. From dynamic programming to greedy algorithms. In Bernhard Möller, Helut A. Partsch, and Steve Schuman, editors, *Formal Program Development: IFIP TC2/WG 2.1 State-of-the-Art report*, volume 755 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin/New York, 1993.
- [130] RICHARD S. BIRD, JEREMY GIBBONS, AND GERAINT JONES. Formal derivation of a pattern matching algorithm. *Science of Computer Programming*, **12**(2):93–104, July 1989.
- [131] GUY E. BLELLOCH. Scans as primitive parallel operations. In Sartaj Sahni, editor, *Proceedings of the Sixteenth International Conference on Parallel Processing*, pages 355–362, Pennsylvania State University, 17–21 August 1987. Pennsylvania State University Press, University Park, PA, 1987.
- [132] GUY E. BLELLOCH. *Vector Models for Data-Parallel Computing*. Artificial Intelligence. MIT Press, Cambridge, MA, 1990.
- [133] GUY E. BLELLOCH. NESL: a nested data parallel language. Technical Report CMU-CS-92-193, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January 1992.
- [134] GUY E. BLELLOCH. Prefix sums and their applications. In John H. Reif, editor, *Synthesis of Parallel Algorithms*, pages 35–60. Morgan Kaufmann Publishers, San Mateo, CA, 1993.
- [135] C. DE BOOR. Efficient computer manipulation of tensor products. *ACM Transactions on Mathematical Software*, **5**(2):173–182, 1979.
- [136] SHEKHAR BORKAR, ROBERT COHN, GEORGE COX, SHA GLEASON, THOMAS GROSS, HSING-TSUNG KUNG, MONICA S. LAM, BRIAN MOORE, CRAIG PETERSON, JOHN PIEPER, LINDA RANKIN, PING SHENG TSENG, JIM SUTTON, JOHN URBANSKI, AND JON WEBB. iWarp: an integrated solution to high-speed parallel computing. In *Proceedings: Supercomputing '88*, pages 330–339, Orlando, FL, 14–18 November 1988. IEEE Computer Society Press, Washington, DC, 1988.
- [137] W.J. BOUKNIGHT, STEWART A. DENENBERG, DAVID E. MCINTYRE, J.M. RANDALL, AMED H. SAMEH, AND DANIEL L. SLOTNICK. The ILIAC IV system. *Proceedings of the IEEE*, **60**(4):369–388, 1972.
- [138] CHARLES LEONARD BOUTON. Nim, a game with a complete mathematical theory. *Annals of Mathematics*, **3**(2):35–39, 1902.
- [139] ROBERT STEPHEN BOYER AND J STROTHER MOORE. A fast string searching algorithm. *Communications of the ACM*, **20**(10):762–772, October 1977.
- [140] RICHARD PEIRCE BRENT. The parallel evaluation of general arithmetic expressions. *Journal of the Association for Computing Machinery*, **21**(2):201–206, April 1974.
- [141] CLAY P. BRESHEARS AND MICHAEL ALLEN LANGSTON. MIMD versus SIMD computation: experience with non-numeric parallel algorithms. In Trevor N. Mudge, Veljko M. Milutinovic, and Lawrence Hunter, editors, *Proceedings of the Twenty-Sixth Hawaii International Conference on Systems Sciences*, volume 2, pages 298–307, Wailea, HI, 5–8 January 1993. IEEE Computer Society, Los Alamitos, CA, 1993.
- [142] JONATHAN BRIGHT, SIMON KASIF, AND LEWIS BENJAMIN STILLER. Exploiting algebraic structure in parallel state space search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 2, pages 1341–1346, Seattle, Washington, 31 July–4 August 1994. AAAI Press / The MIT Press, Menlo Park/Cambridge, 1994.

- [143] CYNTHIA A. BROWN, LARRY FINKELSTEIN, AND PAUL WALTON PURDOM, JR. Backtrack searching in the presence of symmetry. In Teo Mora, editor, *Proceedings of the 6th International Conference on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 357 of *Lecture Notes in Computer Science*, pages 99–110, Rome, Italy, 4–8 July 1988. Springer-Verlag, Berlin, 1989.
- [144] CYNTHIA A. BROWN, LARRY FINKELSTEIN, AND PAUL WALTON PURDOM, JR. A new base change algorithm for permutation groups. *SIAM Journal on Computing*, **18**(5):1037–1047, October 1989.
- [145] RICHARD ANTHONY BRUALDI AND HERBERT JOHN RYSER. *Combinatorial Matrix Theory*, volume 39 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, England, 1991.
- [146] JEAN-PHILIPPE BRUNET AND S. LENNART JOHNSON. All-to-all broadcast with applications on the Connection Machine. *International Journal of Supercomputer Applications*, **6**(3):241–256, Fall 1992.
- [147] DUNCAN A. BUELL. Broadcast and total exchange in supertoroidal networks. Technical Report SRC-TR-91-048, Supercomputing Research Center, Bowie, MD, 6 November 1991.
- [148] DUNCAN A. BUELL. Supertoroids, FFT butterflies, and cube-connected-cycles. Technical Report SRC-TR-91-045, Supercomputing Research Center, 17100 Science Drive, Bowie, Maryland, 20 August 1991.
- [149] JERRY R. BURCH, EDMUND MELSON CLARKE, KENNETH L. MCMILLAN, DAVID L. DILL, AND L.J. HWANG. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, **98**(2):142–170, June 1992.
- [150] ALICE R. BURKS AND ARTHUR WALTER BURKS. The ENIAC: first general-purpose electronic computer. *Ann. Hist. Comput.*, **3**(4):310–399, 1981.
- [151] MICHAEL BUSSIECK, HANNES HASSLER, GERHARD J. WOEGINGER, AND UWE T. ZIMMERMANN. Fast algorithms for the maximum convolution problem. *Operations Research Letters*, **15**(3):133–141, April 1994.
- [152] GREGORY BUTLER. *Fundamental Algorithms For Permutation Groups*, volume 559 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin/New York, 1992.
- [153] GREGORY BUTLER AND CLEMENT WING HONG LAM. Isomorphism testing of combinatorial objects. *Journal of Symbolic Computation*, **1**(4):363–381, December 1985.
- [154] JIN-YI CAI. Parallel computation over hyperbolic groups. In *Proceedings of the 24th Annual Symposium on Theory of Computing*, pages 106–115, Victoria, B.C, Canada, 4–6 May 1992. Association for Computing Machinery, New York, 1992.
- [155] LARRY J. CAMPBELL, LUKE L. DAEMEN, JAMES E. GUBERNATIS, AND LEWIS BENJAMIN STILLER. Computer simulations of flux dynamics in superconducting thin-films, August 1992.
- [156] LOWELL CAMPBELL, GUNNAR ERIK CARLSSON, MICHAEL J. DINNEEN, VANCE FABER, MICHAEL RALPH FELLOWS, MICHAEL ALLEN LANGSTON, JAMES W. MOORE, ANDREW P. MULLHAUPT, AND HARLAN B. SEXTON. Small diameter symmetric networks from linear groups. *IEEE Transactions on Computers*, **41**(2):218–220, February 1992.
- [157] MURRAY S. CAMPBELL. Chunking as an abstraction mechanism. Technical Report CMU-CS-88-116, Carnegie-Mellon University, 22 February 1988.

- [158] MURRAY S. CAMPBELL AND THOMAS ANTHONY MARSLAND. A comparison of minimax tree search algorithms. *Artificial Intelligence*, **20**:347–367, 1983.
- [159] LYNN ELLIOT CANNON. *A Cellular Computer to Implement the Kalman Filter Algorithm*. PhD thesis, Montana State University, 1969.
- [160] GUNNAR ERIK CARLSSON, J.E. CRUTHIRDS, HARLAN B. SEXTON, AND CHRISTOPHER G. WRIGHT. Interconnection networks based on a generalization of cube-connected cycles. *IEEE Transactions on Computers*, **C-34**(8):769–772, August 1985.
- [161] GUNNAR ERIK CARLSSON, MICHAEL RALPH FELLOWS, HARLAN B. SEXTON, AND CHRISTOPHER G. WRIGHT. Group theory as an organizing principle in parallel processing. Unpublished manuscript, December 1985.
- [162] GUNNAR ERIK CARLSSON, HARLAN B. SEXTON, M.J. SHENSA, AND CHRISTOPHER G. WRIGHT. Algebraic techniques in systolic array design. Technical Report NOSC TR 942, Naval Ocean Systems Center, San Diego, CA 92152–5000, February 1984.
- [163] GUNNAR ERIK CARLSSON, HARLAN B. SEXTON, AND CHRISTOPHER G. WRIGHT. Node-transitive networks. Technical Report TN 1380, Naval Ocean Systems Center, San Diego, CA 92152–5000, March 1985.
- [164] PIETRO CARRERA. *Del gioco de gli scacchi* [The game of chess], volume 3. [Gionanni de Rossi?], [Trento?], 1617.
- [165] J. CARRIER, LESLIE GREENGARD, AND VLADIMIR ROKHLIN. A fast adaptive multipole algorithm for particle simulations. *SIAM Journal on Scientific and Statistical Computing*, **9**(4):669–686, July 1988.
- [166] ÉLIE CARTAN. Sur les groupes bilinéaires et les systèmes de nombres complexes [On bilinear groups and systems of complex numbers]. *Ann. Fac. Sc. Toulouse*, **12**, 1898.
- [167] ROGER WILLIAM CARTER. *Simple Groups of Lie Type*, volume 28 of *Pure and Applied Mathematics: A Series of Texts and Monographs*. John Wiley & Sons, London/New York, 1972.
- [168] ARTHUR CAYLEY. On the theory of groups, as depending on the symbolic equation $\theta^n = 1$. *Philosophical Magazine*, **7**:123–130, 1854.
- [169] ARTHUR CAYLEY. On the theory of groups, as depending on the symbolic equation $\theta^n = 1$. *Philosophical Magazine*, **7**:131–132, 1854.
- [170] R.M. CHAMBERLAIN. Gray codes, fast Fourier transforms, and hypercubes. *Parallel Computing*, **6**(2):225–233, February 1988.
- [171] MEE YEE CHAN. Embedding of grids into optimal hypercubes. *SIAM Journal on Computing*, **20**(5):834–864, October 1991.
- [172] BRUCE CHANDLER AND WILHELM MAGNUS. *The History of Combinatorial Group Theory: A Case Study in the History of Ideas*, volume 9 of *Studies in the History of Mathematics and Physical Sciences*. Springer-Verlag, New York, 1982.
- [173] ASHOK KUMAR CHANDRA, DEXTER CAMPBELL KOZEN, AND LARRY JOSEPH STOCKMEYER. Alternation. *Journal of the Association for Computing Machinery*, **28**(1):114–133, January 1981.

- [174] HENRY KER-CHANG CHANG AND JONATHAN JEN-RONG CHENG. A parallel algorithm for the knapsack problem with memory/processor tradeoff $M^2P = O(2^{n/2})$. *International J. of High Speed Computing*, **4**(2):109–120, June 1992.
- [175] CHAPPAIS. *Essais analytiques sur les échecs, avec figures* [analytical essay on chess, with illustrations], [ca. 1780].
- [176] BARBARA CHAPMAN, PIYUSH MEHROTRA, AND HANS P. ZIMA. Programming in Vienna Fortran. *Scientific Programming*, **1**(1):31–50, Fall 1992.
- [177] GEN-HUEY CHEN AND JIN-HWANG JANG. An improved parallel algorithm for 0/1 knapsack problem. *Parallel Computing*, **18**(7):811–821, 1992.
- [178] ANDRÉ CHÉRON. *Nouveau traité complet d'échecs: La fin de partie* [New complete treatise of chess]. Yves Demailly, Lille, France, 1952.
- [179] ANDRÉ CHÉRON. *Lehr- und Handbuch der Endspiele* [Textbook and handbook of the endgame]. S. Engelhardt, Berlin, 2 edition, 1960, 1970. Enlarged and revised edition of Nouveau traite complet d'échecs, la fin de partie.
- [180] WAI-MEE CHING. Automatic parallelization of APL-style programs. *APL90 Conference proceedings, APL Quote Quad*, **20**(4):76–80, July 1990.
- [181] JAEYOUNG CHOI, JACK J. DONGARRA, ROLDAN POZO, AND DAVID W. WALKER. ScaLAPACK: a scalable linear algebra library for distributed memory concurrent computers. In *Proceedings of the Fourth Symposium on the Frontiers of Massively Parallel Computation: Frontiers '92*, McLean, VA, 1992. IEEE Computer Society Press, Los Alamitos, CA, 1992.
- [182] JAEYOUNG CHOI, JACK J. DONGARRA, AND DAVID W. WALKER. The design of a parallel, dense linear algebra software library: reduction to Hessenberg, tridiagonal and bidiagonal form. In Jack J. Dongarra and Bernard Tourancheau, editors, *Proceedings of the Second Workshop on Environments and Tools for Parallel Scientific Computing*, pages 98–111, Townsend, Tennessee, 25–27 May 1994. Society for Industrial and Applied Mathematics, Philadelphia, 1994.
- [183] DANIEL LE MÉTAYER CHRIS HANKIN AND DAVID SANDS. A parallel programming style and its algebra of programs. In Arndt Bode, Mike Reeve, and Gottfried Wolf, editors, *PARLE '93: Parallel Architectures and Languages Europe: 5th International PARLE Conference*, volume 694 of *Lecture Notes in Computer Science*, pages 367–390, Munich, Germany, 14–17 June 1993. Springer-Verlag, Berlin, 1993.
- [184] DAVID V. CHUDNOVSKY, GREGORY V. CHUDNOVSKY, AND MONTY MONTAGUE DENNEAU. Regular graphs with small diameter as models for interconnection networks. In Svetlana Kartashev and Steven I. Kartashev, editors, *Proceedings: Third International Conference on Supercomputing*, volume 3, pages 232–239, Boston, MA, 15–20 May 1988. International Supercomputing Institute, St. Petersburg, FL, 1988.
- [185] FAN RONG KING CHUNG. Diameters and eigenvalues. *Journal of the American Mathematical Society*, **2**(2):187–196, April 1989.
- [186] MICHAEL CLAUSEN. Fast Fourier transforms for metabelian groups. *SIAM Journal on Computing*, **18**(3):594–593, June 1989.
- [187] MICHAEL CLAUSEN. Fast generalized Fourier transforms. *Theoretical Computer Science*, **67**(1):55–63, 5 September 1989.

- [188] MICHAEL CLAUSEN AND ULRICH BAUM. Fast Fourier transforms for symmetric groups. In Larry Finkelstein and William Kantor, editors, *Groups and Computation: Workshop on Groups and Computation*, volume 11 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 27–40, Rutgers, NJ, 7–10 October 1991. American Mathematical Society, Providence, RI, 1993.
- [189] MICHAEL CLAUSEN AND ULRICH BAUM. *Fast Fourier Transforms*. Bibliographisches Institut Wissenschaftsverlag, Mannheim, 1993.
- [190] MICHAEL CLAUSEN AND ULRICH BAUM. Fast Fourier transforms for symmetric groups: theory and implementation. *Mathematics of Computation*, **61**(204):833–847, October 1993.
- [191] MICHAEL CLAUSEN AND DIETER GOLLMANN. Spectral transforms for symmetric groups—fast algorithms and VLSI architectures. In Claudio Moraga, editor, *Proceedings of the Third International Workshop on Spectral Techniques*, pages 67–85, University of Dortmund, F.R.G., 4–6 October 1988.
- [192] P. CLOTE AND E. KRANAKIS. Boolean functions, invariance groups, and parallel complexity. *SIAM Journal on Computing*, **20**(3):553–590, June 1991.
- [193] WILLIAM T. COCHRAN, JAMES WILLIAM COOLEY, DAVID L. FAVIN, HOWARD D. HELMS, REGINALD A. KAENEL, WILLIAM W. LANG, GEORGE C. MALING, JR., DAVID E. NELSON, CHARLES M. RADER, AND PETER D. WELCH (G-AE SUBCOMMITTEE ON MEASUREMENT CONCEPTS). What is the fast Fourier transform? *IEEE Transactions on Audio and Electroacoustics*, **AU-15**(2):45–55, June 1967.
- [194] RICHARD COLE AND UZI VISHKIN. Approximate and exact parallel scheduling with applications to list, tree, and graph problems. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 478–491, Toronto, Ontario, 27–29 October 1986. IEEE Computer Society Press, Washington, DC / Los Angeles, CA, 1986.
- [195] ALBERT JOHN COLEMAN. *Induced Representations With Applications to S_n and $GL(n)$* , volume 4 of *Queen's Papers in Pure and Applied Mathematics*. Queen's University, Kingston, Ontario, 1966.
- [196] J. F. COLLINS AND A. F. W. COULSON. Applications of parallel processing algorithms for DNA sequence analysis. *Nucleic Acids Research*, **12**(1 Part 1):181–192, 11 January 1984.
- [197] MICHAEL JOHN COLLINS. *Representations and Characters of Finite Groups*, volume 22 of *Cambridge Studies in Applied Mathematics*. Cambridge University Press, Cambridge, England/New York, 1990.
- [198] J. H. CONDON AND KENNETH LANE THOMPSON. Belle chesse hardware. In Peter W. Frey, editor, *Chess Skill in Man and Machine*, pages 201–210. Springer-Verlag, 1977.
- [199] JAMES WILLIAM COOLEY, PETER A. W. LEWIS, AND PETER D. WELCH. Historical notes on the fast Fourier transform. *IEEE Transactions on Audio and Electroacoustics*, **AU-15**(2):76–79, June 1967.
- [200] JAMES WILLIAM COOLEY AND JOHN WILDER TUKEY. An algorithm for the machine calculation of complex Fourier series,. *Mathematics of Computation*, **19**(90):297–301, April 1965.
- [201] GENE COOPERMAN AND LARRY FINKELSTEIN. Combinatorial tools for computational group theory. In Larry Finkelstein and William Kantor, editors, *Groups and Computation: Workshop on Groups and Computation*, volume 11 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 53–86, Rutgers, NJ, 7–10 October 1991. American Mathematical Society, Providence, RI, 1993.

- [202] GENE COOPERMAN, LARRY FINKELSTEIN, AND NAMITA SARAWAGI. Applications of Cayley graphs. In *Applied Algebra, Algebraic Algorithms, and Error-correcting Codes : 8th International Conference, AAEECC-8: Proceedings*, volume 508 of *Lecture Notes in Computer Science*, pages 367–378, Tokyo, Japan, 20–24 August 1990. Berlin, Springer-Verlag, 1991.
- [203] GENE COOPERMAN, LARRY FINKELSTEIN, AND NAMITA SARAWAGI. A random base change algorithm for permutation groups. In *ISSAC '90: Proceedings of the International Symposium on Symbolic and Algebraic Computation*, Tokyo, Japan, August 20–24 1990. ACM press, New York, 1990.
- [204] GENE COOPERMAN, LARRY FINKELSTEIN, AND BRYANT WHITTIER YORK. A parallel implementation of group membership and the method of random subproducts. In *Proceedings of the First Summer Institute on Issues and Ostacles in the Practical Implementation of Parallel Algorithms and the Use of Parallel Machines*, Hanover, NH, 23–27 June 1992. Dartmouth Institute for Advanced Graduate Studies in Parallel Computation, Dartmouth College, 1992.
- [205] DON COPPERSMITH AND SHMUEL WINOGRAD. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, **9**(3):251–280, March 1990.
- [206] NOLAN G. CORE, ELIZABETH W. EDMISTON, JOEL HASKIN SALTZ, AND ROGER M. SMITH. Supercomputers and biological sequence comparison algorithms. *Computers and Biomedical Research*, **22**(6):497–515, December 1989.
- [207] INTERNATIONAL BUSINESS MACHINES CORPORATION. *IBM 7094 Principles of Operation*. IBM Systems Reference Library. IBM, Poughkeepsie, NY, 1963.
- [208] A. F. W. COULSON, J. F. COLLINS, AND A. LYALL. Protein and nucleic acid sequence database searching: a suitable case for parallel processing. *The Computer Journal*, **30**(5):420–424, October 1987.
- [209] MAXIME CROCHEMORE. Optimal factor transducers. In Alberto Apostolico and Zvi Galil, editors, *Proceedings of the NATO Advanced Research Workshop on Combinatorial Algorithms on Words*, volume 12 of *NATO Advanced Science Institutes Series F: Computer and Systems Sciences*, pages 31–43, Maratea, Italy, 18–22 June 1984. Springer-Verlag, Berlin/New York, 1985.
- [210] Maxime Crochemore and Dan Gusfield, editors. *Combinatorial Pattern Matching: 5th Annual Symposium*, volume 807 of *Lecture Notes in Computer Science*, Asilomar, CA, 5–8 June 1994. Springer-Verlag, Berlin/New York 1993.
- [211] MAXIME CROCHEMORE AND DOMINIQUE PERRIN. Two-way string-matching. *Journal of the Association for Computing Machinery*, **38**(3):651–675, July 1991.
- [212] [ALFRED CROSSKILL]. The rook and bishop against rook. *The Chess-Player's Magazine*, **2**:305–311, 1864.
- [213] VAN-DAT CUNG AND BERTRAND LE CUN. An efficient implementation of parallel A*. In Michel Cosnard, Afonso Ferreira, and Joseph Peters, editors, *Proceedings of the First Canada-France Conference on Parallel and Distributed Computing: Theory and Practice*, volume 805 of *Lecture Notes in Computer Science*, Montréal, Canada, 19–21 May 1994. Springer-Verlag, Berlin, 1994.
- [214] CHARLES W. CURTIS AND IRVING REINER. *Representation Theory of Finite Groups and Associative Algebras*, volume 11 of *Pure and Applied Mathematics: A Series of Texts and Monographs*. John Wiley & Sons, 1962.

- [215] ROBERT CYPHER AND JORGE L. C. SANZ. *The SIMD Model of Parallel Computation*. Springer-Verlag, New York, 1994.
- [216] LUKE L. DAEMEN, JAMES EDWARD GUBERNATIS, AND LARRY J. CAMPBELL. Flux lattice melting in superconducting thin films. Unpublished, December 1991.
- [217] E. DENNING DAHL. Mapping and compiled communication on the Connection Machine system. In *Proceedings of the Fifth Distributed Memory Computing Conference*, pages 756–766, Charleston, South Carolina, 8–12 April 1990. IEEE Computer Society, Los Alamitos, CA, 1990.
- [218] GORDON CHARLES DANIELSON AND CORNELIUS LANCZOS. Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids. *Journal of the Franklin Institute*, **233**(4):365–380, April 1942.
- [219] GORDON CHARLES DANIELSON AND CORNELIUS LANCZOS. Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids (continued from april issue). *Journal of the Franklin Institute*, **233**(5):435–452, May 1942.
- [220] ALAIN DARTE, TANGUY RISSET, AND YVES ROBERT. Loop nest scheduling and transformations. In Jack J. Dongarra and Bernard Tourancheau, editors, *Environments and Tools for Parallel Scientific Computing*, volume 6 of *Advances in Parallel Computing*, pages 309–332. North-Holland, Amsterdam, 1993.
- [221] JACK WINFRED DAVIDSON AND CHRISTOPHER WARWICK FRASER. Automatic generation of peephole optimizations. In *Proceedings of the SIGPLAN '84 Symposium on Compiler Construction*, pages 111–116, Montréal, Canada, 17–22 June 1984. Association for Computing Machinery, New York, 1984.
- [222] MARC DAVIO. Kronecker products and shuffle algebra. *IEEE Transactions on Computers*, **C-30**(2):116–125, February 1981.
- [223] PHILIP J. DAVIS. *Circulant Matrices*. John Wiley & Sons, New York, 1979.
- [224] THOMAS RAYNER DAWSON AND W. HUNSDORFER. *Retrograde Analysis*. Whitehead and Miller, Leeds, 1915.
- [225] PILAR DE LA TORRE AND CLYDE P. KRUSKAL. Towards a single model of efficient computation in real parallel machines. In Emile H.L. Aarts, Jan van Leeuwen, and M. Rem, editors, *PARLE '91: Parallel Architectures and Languages Europe*, volume 505–506 of *Lecture Notes in Computer Science*, pages 6–24, Eindhoven, Netherlands, 1991. Springer-Verlag, Berlin/New York, 1991.
- [226] LUIS RAMIREZ DE LUCENA. *Repetición de Amores y Arte de Ajedrez* [treatise on love and the game of chess]. Salamanca, ca. 1497. Facsimile reprint Colección Joyas Bibliográficas:Madrid, Spain. 1953.
- [227] RUY LOPEZ DE SIGURA. *Libro de la invencion liberal y arte del juego del axedrez* [The book of liberal invention and the art of the game of chess]. Alcala, 1561.
- [228] [NICHOLAS DE ST NICHOLAI?]. *Bonus Socius* [good companion], 13th century.
- [229] ERIC F. VAN DE VELDE. Data redistribution and concurrency. *Parallel Computing*, **16**(2–3):125–138, December 1990.
- [230] NICOLAAS GOVERT DEBRUIJN. A combinatorial problem. *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen (A)*, **49**(Part 2):758–764, 1946.

- [231] RINA DECHTER AND JUDEA PEARL. Generalized best-first search strategies and the optimality of A*. *Journal of the Association for Computing Machinery*, **32**(3):505–536, July 1985.
- [232] WOLFGANG DECKER. *Sports and Games of Ancient Egypt*. Yale University Press, New Haven, 1992.
- [233] ELIEZER DEKEL, DAVID NASSIMI, AND SARTAJ SAHNI. Parallel matrix and graph algorithms. *SIAM Journal on Computing*, **10**(4):657–673, November 1981.
- [234] SITO T. DEKKER, H. JAAP VAN DEN HERIK, AND I.S. HERSCHBERG. Complexity starts at five. *International Computer Chess Association Journal*, **10**(3):125–138, September 1987.
- [235] SITO T. DEKKER, H. JAAP VAN DEN HERIK, AND I.S. HERSCHBERG. Perfect knowledge revisited. *Artificial Intelligence*, **43**(1):111–123, April 1990.
- [236] ARTHUR L. DELCHER AND SIMON KASIF. Efficient parallel term matching and anti-unification. *Journal of Automated Reasoning*, **3**(3):391–406, 1992.
- [237] JAMES W. DEMMEL, MICHAEL T. HEATH, AND HENK A. VAN DER VORST. Parallel linear algebra. In *Acta Numerica 1993*, pages 111–197. Cambridge University Press, Cambridge, England, 1993.
- [238] PERSI DIACONIS. *Group Representations in Probability and Statistics*, volume 11 of *Lecture Notes—Monograph Series*. Institute of Mathematical Statistics, Hayward, CA, 1988.
- [239] PERSI DIACONIS. A generalization of spectral analysis with application to ranked data. *Annals of Statistics*, **17**(3):949–979, September 1989.
- [240] PERSI DIACONIS AND DANIEL NAHUM ROCKMORE. Efficient computation of the Fourier transform on finite groups. *Journal of the American Mathematical Society*, **3**(2):297–332, April 1990.
- [241] JOHN D. DIXON. Constructing representations of finite groups. In Larry Finkelstein and William Kantor, editors, *Groups and Computation: Workshop on Groups and Computation*, volume 11 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 113–126, Rutgers, NJ, 7–10 October 1991. American Mathematical Society, Providence, RI, 1993.
- [242] MICHAEL DIXON AND JOHANN DE KLEER. Massively parallel assumption based truth maintenance. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, volume 1, pages 199–204, Saint Paul, MN, 21–26 August 1988. Morgan Kaufmann, Palo Alto, CA, 1988.
- [243] JACK J. DONGARRA, JEREMY DU CROZ, SVEN HAMMARLING, AND IAIN DUFF. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, **16**(1):1–17, March 1990.
- [244] JACK J. DONGARRA, FRAN GOERTZEL GUSTAVSON, AND A. KARP. Implementing linear algebra algorithms for dense matrices on a vector pipeline machine. *SIAM Review*, **26**(1):91–112, January 1984.
- [245] RICHARD NOEL DRAPER. A fast distributed routing algorithm for supertoroidal networks. Technical Report SRC-TR-91-032, Supercomputing Research Center Institute for Defense Analyses, Bowie, MD, 24 July 1990.
- [246] RICHARD NOEL DRAPER. A fast distributed routing algorithm for supertoroidal networks. Technical report, Supercomputing Research Center, July 1990.

- [247] RICHARD NOEL DRAPER. Lecture on chess endgames and integral transforms and the paper by Lewis Stiller “Group graphs and computational symmetry on massively parallel architecture”. Unpublished notes, 14 March 1991.
- [248] RICHARD NOEL DRAPER. An overview of supertoroidal networks. Technical Report SRC-TR-91-035, Supercomputing Research Center, Bowie, Maryland, 17 January 1991.
- [249] RICHARD NOEL DRAPER. An overview of supertoroidal networks. In *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 95–102, Hilton Head, SC, 21–24 July 1991. ACM Press, Baltimore, MD, 1991.
- [250] RICHARD NOEL DRAPER AND VANCE FABER. The diameter and mean diameter of supertoroidal networks. Technical Report SRC-TR-90-004, Supercomputing Research Center, Bowie, MD, 1990.
- [251] JAMES R. DRISCOLL AND MERRICK LEE FURST. Computing short generator sequences. *Information and computation*, **72**:117–132, 1987.
- [252] JAMES R. DRISCOLL AND DENNIS M. HEALY JR. Computing fourier transforms and convolutions on the 2-sphere. Manuscript, 24 August 1992.
- [253] YURIJ A. DROZD AND VLADIMIR V. KIRICHENKO. *Finite Dimensional Algebras*. Springer-Verlag, Berlin, 1994.
- [254] MEIR DRUBIN. Kronecker product factorization of the FFT matrix. *IEEE Transactions on Computers*, **C-20**(5):590–593, May 1971.
- [255] JIANZHONG DU AND JOSEPH Y.-T. LEUNG. Complexity of scheduling parallel task systems. *SIAM Journal on Discrete Mathematics*, **2**(4):473–487, November 1989.
- [256] MOSHE DUBINER, ZVI GALIL, AND EDITH MAGEN. Faster tree pattern matching. *Journal of the Association for Computing Machinery*, **41**(2):205–213, March 1994.
- [257] NANCY EATON. The historical development of matrices and determinants. Master’s thesis, San Jose State College, Department of Mathematics, June 1969.
- [258] WILLIAM HENRY CARL EBELING. *All the Right Moves: A VLSI Architecture for Chess*. ACM Distinguished Dissertations. MIT Press, Cambridge, MA, 1987.
- [259] DAVID EBERLY AND DENNIS WENZEL. Adaptation of group algebras to signal and image processing. *CVGIP: Graphical Models and Image Processing*, **53**(4):340–348, July 1991.
- [260] THE EDITORS. Thompson: quintets with variations. *International Computer Chess Association Journal*, **16**(2):86–90, June 1993.
- [261] K. EFE. The crossed cube architecture for parallel computing. *IEEE Transactions on Parallel and Distributed Systems*, **3**(5):512–524, September 1992.
- [262] JACK A. EIDSWICK. Cubelike puzzles—what are they and how do you solve them? *American Mathematical Monthly*, **93**(3):157–176, March 1986.
- [263] JAN-OLOF EKLUNDH. A fast computer method for matrix transposing. *IEEE Transactions on Computers*, **C-21**(7):801–803, July 1972.
- [264] NOAM DAVID ELKIES. Chess art in the computer age. *American Chess Journal*, **1**(2):48–52, September 1993.

- [265] NOAM DAVID ELKIES. On numbers and endgames. In Richard Nowakowski, editor, *Proceedings of the 1994 Workshop on Combinatorial Games*, Berkeley, CA, 11–22 July 1994. To appear.
- [266] THOMAS ELLMAN. Abstraction via approximate symmetry. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, volume 2, pages 916–921, Chambéry, France, 28 August–3 September 1993. Morgan Kaufmann Publishers, San Mateo, CA, 1993.
- [267] ABDOL-HOSSEIN ESFAHANIAN, LIONAL M. NI, AND BRUCE ELI SAGAN. The twisted N-cube with application to multiprocessing. *IEEE Transactions on Computers*, **40**(1):88–93, January 1991.
- [268] P.M. FLANDERS ET AL. Efficient high speed computing with the distributed array processor. In D.J. Kuch, Duncan Lawrie, and Ahmed Sameh, editors, *Proceedings of the Symposium on High Speed Computer and Algorithm Organization*, pages 113–127, University of Illinois, 13–15 April 1977. Academic Press, New York, 1977.
- [269] MACHGIELIS EUWE. *Het Eindspel* [The endgame]. G.B. van Goor Zonen’s Uitgeversmaatschappij, ’s-Gravenhage;Batavia, 1940.
- [270] MACHGIELIS EUWE. *Stukken Tegen Stukken II* [Pieces against pieces II], volume 5 of *Het Eindspel*. G.B. van Goor Zonen’s Uitgeversmaatschappij, ’s-Gravenhage;Batavia, 1940.
- [271] MACHGIELIS EUWE. *Das Endspiel* [The endgame]. Das Schach-Archiv, F.L. Rattman, Hamburg, 1957.
- [272] MATTHEW EVETT, JAMES HENDLER, AMBUJASHKA MAHANTI, AND DANA NAU. PRA*: A memory-limited heuristic search procedure for the Connection Machine. In *Proceedings of the Third Symposium on the Frontiers of Massively Parallel Computations*, pages 145–149, College Park, MD, 8–10 October 1990. IEEE Computer Society Press, Los Alamitos, CA, 1990.
- [273] LYNN FOREST TEN EYCK. Crystallographic fast Fourier transforms. *Acta Crystallographica, Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, **A29**(Part 2):183–190, 1 March 1973.
- [274] VANCE FABER, JAMES W. MOORE, AND WILLIAM Y.C CHEN. Cycle prefix digraphs for symmetric interconnection networks. *Networks*, **23**(7):641–649, October 1993.
- [275] ALBERT FÄSSLER AND EDUARD L. STIEFEL. *Gruppentheoretische Methoden und ihre Anwendung* [Group-theoretical methods and their application]. B. G. Teubner, Stuttgart, 1979.
- [276] ALBERT FÄSSLER AND EDUARD L. STIEFEL. *Group theoretical methods and their applications*. Birkhäuser, Boston, 1992. translated revised edition.
- [277] RAINER FELDMANN. *Spielbaumsuche mit Massiv Parallelen Systemen* [Game-tree search with massively parallel systems]. PhD thesis, University of Paderhorn, 1993.
- [278] RAINER FELDMANN, BURKHARD MONIEN, PETER MYSLIWIEZ, AND O. VORNBERGER. Distributed game-tree search. *International Computer Chess Association Journal*, **12**(2):65–73, June 1989.
- [279] RAINER FELDMANN, BURKHARD MONIEN, PETER MYSLIWIEZ, AND O. VORNBERGER. Distributed game tree search. In Vipin Kumar, Laveen N. Kanal, and P.S. Gopalakrishnan, editors, *Parallel Algorithms for Machine Intelligence and Vision*, pages 66–101. Springer-Verlag, 1990.

- [280] RAINER FELDMANN, BURKHARD MONIEN, PETER MYSLIWIEZT, AND O. VORNBERGER. A fully distributed chess program. In Don F. Beal, editor, *Advances in Computer Chess 6*, Ellis Horwood Series in Artificial Intelligence, pages 1–27. Ellis Horwood, New York/London, 1991.
- [281] RAINER FELDMANN, PETER MYSLIWIEZT, AND BURKHARD MONIEN. Studying overheads in massively parallel min/max-tree evaluation (extended abstract). In *6th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 94–103, Cape May, NJ, 27–29 June 1994. ACM Press, New York, 1994.
- [282] MICHAEL RALPH FELLOWS. *Encoding graphs in graphs*. PhD thesis, University of California at San Diego, Department of Computer Science, San Diego, CA, 1985.
- [283] EDWARD W. FELTEN AND STEVE WILLIAM OTTO. Chess on a hypercube. In *Third Conference on Hypercube Concurrent Computers and Applications*, volume 2, pages 1329–1341, Pasadena, CA, 19–20 January 1988. ACM Press, New York, 1988.
- [284] CHRIS FERGUSON AND RICHARD EARL KORF. Distributed tree search and its applications to alpha-beta pruning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, volume 2, pages 128–132, Saint Paul, MN, 21–26 August 1988. American Association for Artificial Intelligence, Los Altos, CA, 1988.
- [285] AFONSO G. FERREIRA. Efficient parallel algorithms for the knapsack problem. In Michel Cosnard, Michael H. Barton, and Marco Vanneschi, editors, *Parallel Processing*, pages 169–180. North Holland, Pisa, Italy, April 1988.
- [286] AFONSO G. FERREIRA. The knapsack problem on parallel architectures. In Michel Cosnard et al, editor, *Proceedings of the International Workshop on Parallel and Distributed Algorithms 1988*, Chateau de Bonas, Gers, Fr, oct 3–4 1989. North Holland.
- [287] AFONSO G. FERREIRA. A parallel time/hardware tradeoff $T \cdot H = O(2^{n/2})$ for the knapsack problem. *IEEE Transactions on Computers*, **40**(2):221–225, February 1991.
- [288] C. COTTI FERRERO AND GIOVANNI FERRERO. On certain extensions of quasirings. *Rivista di Matematica della Universita di Parma Serie 5*, **1**:57–63, 1993.
- [289] AMOS FIAT, SHAHAR MOSES, ADI SHAMIR, ILAN SHIMSHONI, AND GABOR TARDOS. Planning and learning in permutation groups. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 274–279, Research Triangle Park, NC, 30 October–1 November 1989. IEEE Computer Society Press, Los Alamitos, CA, 1989.
- [290] NATHAN J. FINE AND HERBERT S. WILF. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, **16**:109–114, 1965.
- [291] REUBEN FINE. *Basic Chess Endings*. David McKay Company, New York, 1941.
- [292] RAPHAEL ARI FINKEL AND JOHN PHILIP FISHBURN. Parallelism in alpha-beta search. *Artificial Intelligence*, **19**(1):89–106, September 1982.
- [293] LARRY FINKELSTEIN, DANIEL KLEITMAN, AND FRANK THOMSON LEIGHTON. Applying the Classification Theorem for finite simple groups to minimize pin count in uniform permutation architectures. In John H. Reif, editor, *VLSI Algorithms and Architectures: 3rd Aegean Workshop on Computing: Proceedings*, volume 319, pages 247–256, Corfu, Greece, 28 June–1 July 1988. Springer-Verlag, Berlin, 1988.

- [294] DANIEL FISCHER, RACQUEL NOREL, RUTH NUSSINOV, AND HAIM J. WOLFSON. 3-D docking of protein molecules. In Alberto Apostolico, Maxime Crochemore, Zvi Galil, and Udi Manber, editors, *Combinatorial Pattern Matching: 4th Annual Symposium. Proceedings*, volume 684 of *Lecture Notes in Computer Science*, pages 20–34, Padova, Italy, 2–4 June 1993. Springer-Verlag, Berlin, 1993.
- [295] MICHAEL JOHN FISCHER AND MICHAEL STEWART PATERSON. String-matching and other products. In Richard Manning Karp, editor, *Complexity of Computation: Proceedings of a Symposium in Applied Mathematics of the American Mathematical Society and the Society for Industrial and Applied Mathematics*, volume 7 of *SIAM-AMS Proceedings*, pages 113–125, New York, NY, 18–19 April 1973. American Mathematical Society, Providence, RI, 1974.
- [296] DANIEL S. FISHER. Flux-lattice melting in thin-film superconductors. *Physical Review B*, **22**(3):1190–1199, August 1980.
- [297] ROBERT W. FLOYD. Algorithm 97: Shortest path. *Communications of the ACM*, **5**(6):345, June 1962.
- [298] MICHAEL JOHN FLYNN. Very high-speed computing systems. *Proceedings of the IEEE*, **54**(12):1901–1909, December 1966.
- [299] DAVID FORTHOFFER, LARS RASMUSSEN, AND SITO DEKKER. A correction to some KRKB-database results. *International Computer Chess Association Journal*, **12**(1):25–27, March 1989.
- [300] HIGH PERFORMANCE FORTRAN FORUM. High performance fortran language specification version 1.0, May 1993.
- [301] GEOFFREY C. FOX, SEEMA HIRANANDANI, KEN KENNEDY, CHARLES H. KOELBEL, ULRICH KREMER, CHAU-WEN TSENG, AND M. WU. Fortran D language specification. Technical Report TR90-141, Department of Computer Science, Rice University, Houston, TX, December 1990.
- [302] GEOFFREY C. FOX, SEEMA HIRANANDANI, KENNETH KENNEDY, CHARLES KOELBEL, U. KREMER, CHAU-WEN TSENG, AND M. WU. Fortran D language specification. Technical Report TR90-141, Department of Computer Science, Rice University, December 1990.
- [303] GEOFFREY C. FOX, MARK A. JOHNSON, GREGORY A. LYZENGA, STEVE WILLIAM OTTO, JOHN K. SALMON, AND WOJTEK FURMANSKI. *Solving Problems on Concurrent Processors*. Printice-Hall, 1988.
- [304] GEOFFREY C. FOX, STEVE WILLIAM OTTO, AND A.J.G. HEY. Matrix algorithms on a hypercube. I. matrix multiplication. *Parallel Computing*, **4**(1):17–31, February 1987.
- [305] AVIEZRI S. FRAENKEL AND DAVID ISAAC JOSEPH LICHTENSTEIN. Computing a perfect strategy for $n \times n$ chess requires time exponential in n . *Journal of Combinatorial Theory, Series A*, **31**(2):199–214, September 1981.
- [306] PETER FRANKL. Cops and robbers in graphs with large girth and Cayley graphs. *Discrete Applied Mathematics*, **17**:301–305, 1987.
- [307] EDWARD HARRY FRIEND. Sorting on electronic computer systems. *Journal of the Association for Computing Machinery*, **3**(3):134–168, 1956.
- [308] FERDINAND GEORG FROBENIUS. Über die Primfactoren der Gruppendedeterminante [On prime factors of group determinants]. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin*, pages 1343–1382, 1896.

- [309] ROGER FRYE AND JACEK MYCZKOWSKI. Exhaustive search of unstructured trees on the connection machine, October 1990.
- [310] WILLIAM FULTON AND JOE HARRIS. *Representation Theory: A First Course*, volume 129 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1991.
- [311] JOAQUIM GABARRO. The design of a parallel algorithm to solve the word problem for free partially commutative groups. In Luc Bouge, Michel Cosnard, Yves Robert, and Denis Trystram, editors, *Parallel Processing: CONPAR 92: Second Joint International Conference on Vector and Parallel Processing*, volume 634, pages 805–806, Lyon, France, 1–4 September 1992. Springer-Verlag, Berlin, 1992.
- [312] ZVI GALIL AND RAFFAELE GIANCARLO. Improved string matching with k mismatches. *SIGACT News*, **17**(4):52–54, Spring 1986.
- [313] ZVI GALIL AND RAFFAELE GIANCARLO. Parallel string matching with k mismatches. *Theoretical Computer Science*, **51**:341–348, 1987.
- [314] ZVI GALIL AND RAFFAELE GIANCARLO. Data structures and algorithms for approximate string matching. *Journal of Complexity*, **4**(1):33–72, 1988.
- [315] ZVI GALIL AND RAFFAELE GIANCARLO. On the exact complexity of string matching: lower bounds. *SIAM Journal on Computing*, **20**(6):1008–1020, December 1991.
- [316] ZVI GALIL AND RAFFAELE GIANCARLO. On the exact complexity of string matching: upper bounds. *SIAM Journal on Computing*, **21**(3):407–437, June 1992.
- [317] ZVI GALIL, CHRISTOPH MARTIN HOFFMANN, EUGENE M. LUKS, CLAUS P. SCHNORR, AND ANDREAS WEBER. An $O(n^3 \log n)$ deterministic and $O(n^3)$ probabilistic isomorphism test for trivalent graphs. In *Proceedings 23d IEEE Symposium on Foundations of Computer Science*, pages 118–125, Chicago, IL, 3–5 November 1982. IEEE Computer Society Press, Los Alamitos, CA, 1982.
- [318] ANGEL E. GARCÍA. Large-amplitude nonlinear motions in proteins. *Physical Review Letters*, **68**(17):2696–2699, 27 April 1992.
- [319] ANGEL E. GARCÍA AND LEWIS BENJAMIN STILLER. Computation of the mean residence time of water in the hydration shells of biomolecules. *Journal of Computational Chemistry*, **14**(11):1396–1406, November 1993.
- [320] RALPH GASSER. Applying retrograde analysis to Nine Men’s Morris. In David N. L. Levy and Don F. Beal, editors, *Heuristic Programming in Artificial Intelligence: the Second Computer Olympiad*, Ellis Horwood Series in Artificial Intelligence, pages 161–173. Ellis Horwood Limited, Chichester, England, 1991.
- [321] RALPH GASSER. Solving Nine Men’s Morris. Manuscript, 1994.
- [322] CARL FRIEDRICH GAUSS. *Disquisitiones Arithmeticae*. Leipzig, 1801.
- [323] CARL FRIEDRICH GAUSS. Nachlass: Theoria interpolationis methodo nova tractata [Theory of interpolation, new methods (posthumous)]. In E. Schering, F. Klein, M. Brendel, and L. Schlesinger, editors, *Carl Friedrich Gauss Werke, Band 3*, pages 265–327. Königlichen Gesellschaft der Wissenschaften, Göttingen, 1866.
- [324] WILLIAM MORVEN GENTLEMAN. Some complexity results for matrix computations on parallel processors. *Journal of the Association for Computing Machinery*, **25**(1):112–115, 1978.

- [325] WILLIAM MORVEN GENTLEMAN AND G. SANDE. Fast Fourier transforms for fun and profit. In *Fall Joint Computer Conference*, volume 29 of *AFIPS Conference Proceedings*, pages 563–578, San Francisco, CA, 7–10 November 1966. Spartan Books, Washington, 1966.
- [326] KURT GEORG AND RICK MIRANDA. Exploiting symmetry in solving linear equations. In Eugene Leo Allgower, Klaus Böhmer, and Martin Golubitsky, editors, *Bifurcation and Symmetry: Cross Influence Between Mathematics and Applications*, volume 104 of *International Series of Numerical Mathematics*, pages 157–168. Birkhäuser Verlag, Basel, Germany, 1992.
- [327] KURT GEORG AND RICK MIRANDA. Exploiting symmetry in solving linear equations. In Eugene Leo Allgower, Klaus Böhmer, and Martin Golubitsky, editors, *Bifurcation and Symmetry: Cross Influence Between Mathematics and Applications*, volume 104 of *International Series of Numerical Mathematics*, pages 157–168. Birkhäuser Verlag, Basel, Germany, 1992.
- [328] KURT GEORG AND JOHANNES TAUSCH. A generalized Fourier transform for boundary element methods with symmetries. Manuscript, 1994.
- [329] APOSTOLOS GERASOULIS. A fast algorithm for the multiplication of generalized Hilbert matrices with vectors. *Mathematics of Computation*, **70**(181):179–188, January 1988.
- [330] APOSTOLOS GERASOULIS, MICHAEL D. GRIGORIADIS, AND SUN LIPING. A fast algorithm for Trummer’s problem. *SIAM Journal on Scientific and Statistical Computing*, **8**(1):135–138, January 1987.
- [331] I. GERTNER AND RICHARD TOLIMIERI. The group theoretic approach to image representation. *Journal of Visual Communication and Image Representation*, **1**(1):67–82, September 1990.
- [332] EVGENI JAKOVLEVIČ GIK. *Schach und Mathematik* [Chess and mathematics]. MIR Moskau and Urania-Verlag, Leipzig, Jena, Berlin, 1986. Translated from the Russian by Wolfgang Hintze.
- [333] E.N. GILBERT. Gray codes and paths on the n -cube. *Bell Systems Technical Journal*, **37**:815–826, May 1958.
- [334] E.N. GILBERT. Synchronization of binary messages. *IRE Transactions on Information Theory*, **IT-6**:470–477, September 1960.
- [335] CHRISTOPHER DAVID GODSIL. *Algebraic Combinatorics*. Chapman and Hall Mathematics. Chapman and Hall, New York, 1993.
- [336] MAYA GOKHALE, WILLIAM HOLMES, ANDREW KOPSER, SARA LUCAS, RONALD MINNICH, DOUGLAS SWEELY, AND DANIEL LOPRESTI. Building and using a highly parallel programmable logic array. *Computer*, **24**(1):81–89, 1991.
- [337] JONATHAN S. GOLAN. *The Theory of Semirings with Applications in Mathematics and Theoretical Computer Science*, volume 54 of *Pitman Monographs and Surveys in Pure and Applied Mathematics*. Longman Scientific, Essex, England, 1992.
- [338] PETER GOLDREICH AND SCOTT TREMAINE. The dynamics of planetary rings. *Annual Review of Astronomy and Astrophysics*, **20**:249–283, 1982.
- [339] HERMAN HEINE GOLDSTINE. *The Computer: From Pascal to von Neumann*. Princeton University Press, Princeton, NJ, 1972.

- [340] HERMAN HEINE GOLDSTINE. *A History of Numerical Analysis From the 16th Through the 19th Century*, volume 2 of *Studies in the History of Mathematics and Physical Sciences*. Springer-Verlag, New York, 1977.
- [341] HARRY GOLOMBEK. *Chess: A History*. G. P. Putnam's Sons, New York, 1976.
- [342] GENE HOWARD GOLUB AND CHARLES FRANCIS VAN LOAN. *Matrix Computations*, volume 3 of *Johns Hopkins Series in the Mathematical Sciences*. The Johns Hopkins University Press, Baltimore, MD, 1983.
- [343] IRVING JOHN GOOD. The interaction algorithm and practical fourier analysis. *Journal of the Royal Statistical Society Series B*, **20**:361–372, 1958.
- [344] JOHN A. GRANATA, MICHAEL CONNER, AND RICHARD TOLIMIERI. A tensor product factorization of the linear convolution matrix. *IEEE Transactions on Circuits and Systems*, **38**(11):1364–1366, November 1991.
- [345] JOHN A. GRANATA, MICHAEL CONNER, AND RICHARD TOLIMIERI. The tensor product: a mathematical programming language for FFTs and other fast DSP operations. *IEEE Signal Processing Magazine*, **9**(1):40–48, January 1992.
- [346] JOHN A. GRANATA AND RICHARD TOLIMIERI. Matrix representations of the multidimensional overlap and add technique. *IEEE Transactions on Circuits and Systems for Video Technology*, **1**(3):289–90, September 1991.
- [347] GIOACHINO GRECO. *Trattato del Nobilissimo et Militare Essercitio de Scacchi nel Quale si Contengono Molti Bellissimi Trattati et la Vera Scienza di Esso Gioco* [treatise on the very noble and military exercise of chess: the science of that game], 1624.
- [348] LESLIE GREENGARD. *The Rapid Evaluation of Potential Fields in Particle Systems*. ACM distinguished dissertations. MIT Press, Cambridge, MA, 1988.
- [349] LESLIE GREENGARD AND WILLIAM D. GROPP. A parallel version of the fast multipole method. In Garry Rodrigue, editor, *Parallel Processing for Scientific Computing: Proceedings of the Third SIAM Conference on Parallel Processing for Scientific Computing*, pages 213–222, Los Angeles CA, 1–4 December 1987. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.
- [350] LESLIE GREENGARD AND VLADIMIR ROKHLIN. A fast algorithm for particle simulations. *Journal of Computational Physics*, **73**(2):325–348, December 1987.
- [351] RAY GREENLAW AND LARRY SNYDER. Achieving speedups for APL on an SIMD parallel computer. *APL Quote Quad*, **18**(4):3–8, June 1988.
- [352] STEPHEN GUATTERY AND GARY L. MILLER. On the performance of spectral graph partitioning methods. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 233–242, San Francisco, CA, 22–24 January 1995. Association for Computing Machinery/Society for Industrial and Applied Mathematics, New York/Philadelphia, 1995.
- [353] ANSHUL GUPTA AND VIPIN KUMAR. The scalability of FFT on parallel computers. *IEEE Transactions on Parallel and Distributed Systems*, **4**(8):922–932, August 1993.
- [354] HIMANSHU GUPTA AND P. SADAYAPPAN. Communication efficient matrix multiplication on hypercubes. In *6th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 320–329, Cape May, New Jersey, 27–29 June 1994. ACM Press, New York, 1994.

- [355] RICHARD J. HANSON, FREDERICK THOMAS KROGH, AND CHARLES LAWRENCE LAWSON. Improving the efficiency of portable software for linear algebra. *SIGNUM Newsletter*, **8**(4):16, 1973.
- [356] MALCOLM C. HARRISON. Implementation of the substring test by hashing. *Communications of the ACM*, **14**(12):777–779, December 1971.
- [357] THOMAS WILLIAM HAWKINS. The origins of the theory of group characters. *Archive for History of Exact Sciences*, **7**(2):142–170, 3 March 1971.
- [358] THOMAS WILLIAM HAWKINS. Hypercomplex numbers, Lie groups, and the creation of group representation theory. *Archive for History of Exact Sciences*, **8**(4):243–287, 25 April 1972.
- [359] THOMAS WILLIAM HAWKINS. New light on Frobenius’ creation of the theory of group characters. *Archive for History of Exact Sciences*, **12**(3):217–143, 14 August 1974.
- [360] TIMOTHY JAMES HEALEY AND J. A. TREACY. Exact block diagonalization of large eigenvalue problems for structures with symmetry. *International Journal for Numerical Methods in Engineering*, **31**(2):265–285, February 1991.
- [361] MICHAEL T. HEIDEMAN, DON H. JOHNSON, AND C. SIDNEY BURRUS. Gauss and the history of the fast Fourier transform. *Archive for History of Exact Sciences*, **34**(3):265–277, 31 October 1985.
- [362] OLAUS MAGNUS FRIEDRICH ERDMAN HENRICI. On a new harmonic analyzer. *London, Edinburgh and Dublin Philosophical Magazine*, Series 5, **38**(230):110–121, July 1894.
- [363] I. S. HERSCHBERG AND H. JAAP VAN DEN HERIK. Thompson’s new data-base results. *International Computer Chess Association Journal*, **9**(1):45–49, March 1986.
- [364] I.S. HERSCHBERG AND H. JAAP VAN DEN HERIK. A gauge of endgames. *International Computer Chess Association Journal*, **8**(4):225–229, December 1985.
- [365] WILLIAM DANIEL HILLIS. *The Connection Machine*. ACM Distinguished Dissertations. MIT Press, Cambridge, MA, 1985.
- [366] WILLIAM DANIEL HILLIS AND JOSHUA EDWARD BARNES. Programming a highly parallel computer. *Nature*, **326**(6108):27–30, March 1987.
- [367] WILLIAM DANIEL HILLIS AND GUY LEWIS STEELE, JR. Data parallel algorithms. *Communications of the ACM*, **29**(12):1170–1183, December 1986.
- [368] WILLIAM DANIEL HILLIS AND WASHINGTON TAYLOR IV. Exploiting symmetry in high-dimensional finite-difference calculations. *Journal of Parallel and Distributed Computing*, **8**(1):77–79, January 1990.
- [369] SUSAN HINRICHS, COREY KOSAK, DAVID R. O’HALLARON, THOMAS M. STRICKER, AND RIIICHIRO TAKE. An architecture for optimal all-to-all personalized communication. In *6th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 310–319, Cape May, New Jersey, June 27–29 1994. ACM Press, New York, 1994.
- [370] SEEMA HIRANANDANI, KEN KENNEDY, AND CHAU-WEN TSENG. Compiling Fortran D for MIMD distributed-memory architectures. *Communications of the ACM*, **35**(8):66–80, August 1992.
- [371] CHING-TIEN HO AND S. LENNART JOHNSON. Embedding meshes in Boolean cubes by graph decomposition. *Journal of Parallel and Distributed Computing*, **8**(4):325–339, April 1990.

- [372] CHING-TIEN HO, S. LENNART JOHNSON, AND ALAN EDELMAN. Matrix multiplication on hypercubes using full bandwidth and constant storage. In *Proceedings of the Sixth Distributed Memory Computing Conference*, pages 447–451, Portland, OR, 28 April–1 May 1991. IEEE Computer Society Press, Los Alamitos, CA 1991.
- [373] ROGER WILLIS HOCKNEY AND JAMES W. EASTWOOD. *Computer Simulation Using Particles*. McGraw Hill, New York, 1981.
- [374] ROGER WILLIS HOCKNEY AND CHRIS R. JESSHOPE. *Parallel Computers 2: Architecture, Programming and Algorithms*. Adam Hilger, Bristol/Philadelphia, 1988.
- [375] CHRISTOPH MARTIN HOFFMANN AND MICHAEL J. O'DONNELL. Pattern matching in trees. *Journal of the Association for Computing Machinery*, **29**(1):68–95, January 1982.
- [376] JOHN HENRY HOLLAND. A universal computer capable of executing an arbitrary number of subprograms simultaneously. In *Proceedings of the 16th Eastern Joint Computer Conference*, pages 108–113, Boston, MA, 1–3 December 1959. Eastern Joint Computer Conference, New York, 1959.
- [377] GERARD J. HOLZMANN. Backward symbolic execution of protocols. In Yechiam Yemini, Robert Strom, and Shaula Yemini, editors, *Protocol Specification, Testing and Verification IV: Proceedings of the IFIP WG 6.1 International Workshop*, pages 19–30, Skytop Lodge, PA, 1985. North Holland, Amsterdam, 1985.
- [378] DAVID HOOPER AND KENNETH WHYLD. *The Oxford Companion to Chess*. Oxford University Press, Oxford, 2nd edition, 1992.
- [379] R. MICHAEL HORD. *Parallel Supercomputing in MIMD architectures*. CRC Press, Boca Raton, FL, 1993.
- [380] FENG-HSIUNG HSU. A two-million moves/s CMOS single-chip chess move generator. *IEEE Journal of Solid-State Circuits*, **22**(5):841–846, October 1987.
- [381] FENG-HSIUNG HSU. *Large scale parallelization of alpha-beta search: an algorithmic and architectural study with computer chess*. PhD thesis, Carnegie-Mellon University, Pittsburgh, February 1990. Also published as CMU Technical Report number CMU-CS-90-108.
- [382] FENG-HSIUNG HSU, THOMAS ANANTHARAMAN, MURRAY S. CAMPBELL, AND ANDREAS NOWATZYK. A grandmaster chess machine. *Scientific American*, **263**(4):18–24, October 1990.
- [383] CHUA-HUANG HUANG, JEREMY R. JOHNSON, AND ROBERT W. JOHNSON. A tensor product formulation of Strassen's matrix multiplication algorithm. *Applied Mathematics Letters*, **3**(3):67–71, 1990.
- [384] CHUA-HUANG HUANG, JEREMY R. JOHNSON, AND ROBERT W. JOHNSON. A report on the performance of an implementation of Strassen's algorithm. *Applied Mathematics Letters*, **4**(1):99–102, 1991.
- [385] BARBARA JANE HUBERMAN. A program to play chess end games. Technical Report CS 106, Stanford Artificial Intelligence Project Memo AI-65, Stanford University Department of Computer Science, 1968.
- [386] STANLEY LEONARD HURST, D.M. MILLER, AND JON C. MUZIO. *Spectral Techniques in Digital Logic*. Academic Press, London and Orlando, 1985.

- [387] KAI HWANG AND FAYE ALAYE BRIGGS. *Computer Architecture and Parallel Processing*. McGraw-Hill Series in Computer Organization and Architecture. McGraw-Hill, New York, 1985.
- [388] ROBERT MORGAN HYATT. Parallel chess on the Cray X-MP/48. *International Computer Chess Association Journal*, **8**(2):90–99, June 1985.
- [389] ROBERT MORGAN HYATT, B.W. SUTER, AND HARRY L. NELSON. A parallel alpha/beta tree searching algorithm. *Parallel Computing*, **10**(3):299–308, May 1989.
- [390] RAMANA M. IDURY AND ALEJANDRO A. SCHÄFFER. Multiple matching of rectangular patterns (extended abstract). In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 81–89, San Diego, CA, 16–18 May 1993. ACM Press, New York, 1993.
- [391] [AMELUNG FAMILY]. [announcement]. *Düna-Zeitung*, page 52, 10–23 March 1909.
- [392] JOHN ISBELL. Sequencing certain dihedral groups. *Discrete Mathematics*, **85**:323–328, 1990.
- [393] JOHN ISBELL. The Gordon game of a finite group. *American Mathematical Monthly*, **99**(6):567–599, June–July 1992.
- [394] KENNETH EUGENE IVERSON. A common language for hardware, software, and applications. In *Proceedings of the AFIPS Fall Joint Computer Conference*, pages 121–129. American Federation of Information Processing Societies/National Press, Palo Alto, CA, 1962, 1962.
- [395] KENNETH EUGENE IVERSON. *A Programming Language*. Wiley, New York, 1962.
- [396] KENNETH EUGENE IVERSON. A programming language. In *Proceedings of the AFIPS Spring Joint Computer Conference*, volume 21, pages 345–351, San Francisco, CA, 1–3 May 1962. American Federation of Information Processing Societies/National Press, Palo Alto, CA.
- [397] JEFFREY ALAN JACKSON. Economics of automatic generation of rules from examples in a chess end-game. Master’s thesis, University of Illinois at Urbana-Champaign, 1984.
- [398] MICHEL JACQUEMIN AND J. ALLAN YANG. Crystal reference manual: Version 3.0. Technical Report TR-840, Yale University Department of Computer Science, October 1991.
- [399] GORDON DOUGLAS JAMES AND ADALBERT KERBER. *The Representation Theory of the Symmetric Group*, volume 16 of *Encyclopedia of Mathematics and its Applications, Section, Algebra*. Addison-Wesley, Reading, MA, 1981.
- [400] PETER JOZEF JANSEN. KQKR: Assessing the utility of heuristics. *International Computer Chess Association Journal*, **15**(4):179–191, December 1992.
- [401] PETER JOZEF JANSEN. KQKR: Awareness of a fallible opponent. *International Computer Chess Association Journal*, **15**(3):111–131, September 1992.
- [402] PETER JOZEF JANSEN. *Using Knowledge about the Opponent in Game-Tree Search*. PhD thesis, Carnegie Mellon University Department of Computer Science, Pittsburgh, PA, September 1992. Also published as Tech. Report CMU-CS-92-192.
- [403] B.J. JECHEV. Full recursive form of the algorithms for fast generalized Fourier transforms. In *CONPAR 86: Conference on Algorithms and Hardware for Parallel Processing*, volume 237 of *Lecture Notes in Computer Science*, pages 112–119, Aachen, Germany, 17–19 September 1986. Springer-Verlag, Berlin/New York, 1986.

- [404] MICHAEL ALEXANDER GEORGE JENKINS. A comparison of array theory and a mathematics of arrays. In Lenore M. Restifo Mullin et. al., editor, *Arrays, functional languages and parallel systems*, pages 237–267. Kluwer Academic Publishers, Boston, MA, 1991.
- [405] MARK JERRUM. A compact representation for permutation groups. *Journal of Algorithms*, **7**(1):60–78, March 1986.
- [406] JEREMY R. JOHNSON, ROBERT W. JOHNSON, DOMINGO RODRIGUEZ, AND RICHARD TOLIMIERI. A methodology for designing, modifying and implementing Fourier transform algorithms on various architectures. *Circuits, Systems, and Signal Processing*, **9**(4):449–500, 1990.
- [407] ROBERT W. JOHNSON. Automatic implementation of tensor products. Manuscript, 24 April 1989.
- [408] ROBERT W. JOHNSON, CHUA-HUANG HUANG, AND JEREMEY R. JOHNSON. Multilinear algebra and parallel programming. *Journal of Supercomputing*, **5**(2–3):189–217, October 1991.
- [409] S. LENNART JOHNSON, MICHEL JACQUEMIN, AND ROBERT L. KRAWITZ. Communication efficient multi-processor FFT. *Journal of Computational Physics*, **102**(2):381–387, October 1992.
- [410] S. LENNART JOHNSON. Communication efficient basic linear algebra computations on hypercube architectures. *Journal of Parallel and Distributed Computing*, **4**(2):133–172, April 1987.
- [411] S. LENNART JOHNSON. Solving tridiagonal systems on ensemble architectures. *SIAM Journal on Scientific and Statistical Computing*, **8**(3):354–329, May 1987.
- [412] S. LENNART JOHNSON, TIM HARRIS, AND KAPIL K. MATHUR. Matrix multiplication on the Connection Machine. Technical Report NA89-3, Thinking Machines Corporation, Cambridge, MA, 1989.
- [413] S. LENNART JOHNSON AND CHING-TIEN HO. Matrix multiplication on Boolean cubes using generic communication primitives. In Arthur Wouk, editor, *Parallel Processing and Medium-Scale Multiprocessors: Workshop Proceedings*, pages 108–156, Stanford, CA, January 1986. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.
- [414] S. LENNART JOHNSON AND CHING-TIEN HO. Matrix transposition on Boolean n-cube configured ensemble architectures. *SIAM J. Matrix Anal. Appl.*, **9**(3):419–454, July 1988.
- [415] S. LENNART JOHNSON AND CHING-TIEN HO. Optimum broadcasting and personalized communication in hypercubes. *IEEE Transactions on Computers*, **38**(9):1249–1268, September 1989.
- [416] S. LENNART JOHNSON AND CHING-TIEN HO. Spanning graphs for optimum broadcasting and personalized communication in hypercubes. *IEEE Transactions on Computers*, **38**(9):1249–1268, September 1989.
- [417] S. LENNART JOHNSON, MICHEL JACQUEMIN, AND CHING-TIEN HO. High radix FFT on Boolean cube networks. Technical Report NA89-7, Thinking Machines Corporation, Cambridge, MA, 1989.
- [418] S. LENNART JOHNSON AND ROBERT L. KRAWITZ. Cooley-Tukey FFT on the Connection Machine. *Parallel Computing*, **18**(11):1201–1221, November 1992.

- [419] S. LENNART JOHNSON, ROBERT L. KRAWITZ, ROGER FRYE, AND D. MACDONALD. A radix-2 FFT on the Connection Machine. In *Proceedings of Supercomputing '89*, pages 809–819, Reno, NV, 13–17 November 1989. ACM, New York, 1989.
- [420] S. LENNART JOHNSON AND KAPIL K. MATHUR. Data structures and algorithms for the finite element method on a data parallel supercomputer. *International Journal of Numerical Methods in Engineering*, **29**(4):881–908, 1990.
- [421] RAY A. KAMIN AND GEORGE BUNCH ADAMS III. Fast Fourier transform algorithm design and tradeoff on the CM-2. *International Journal of High Speed Computing*, **1**(2):207–231, June 1989.
- [422] НиколайФедорович Канунов [NIKOLAI FEDOROVICH KANUNOV]. Федор Эдуардович Молин 1861–1941 [Fedor Eduardovich Molin 1861–1941]. Издательство «Наука», Москва, 1983.
- [423] EHUD DOV KARNIN. A parallel algorithm for the knapsack problem. *IEEE Trans. Comput.*, **C-33**:404–408, May 1984.
- [424] RICHARD MANNING KARP, RAYMOND EDWARD MILLER, AND ARNOLD LEONARD ROSENBERG. Rapid identification of repeated patterns in strings, trees and arrays. In *Conference Record: Fourth Symposium on Theory of Computing*, pages 125–136, Denver, CO, 1–3 May 1972. ACM Order Department, New York, 1972.
- [425] RICHARD MANNING KARP AND MICHAEL O. RABIN. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, **31**(2):249–260, March 1987.
- [426] RICHARD MANNING KARP AND YANJUN ZHANG. On parallel evaluation of game trees. In *Proceedings of the 1989 ACM Symposium on Parallel Algorithms and Architectures*, pages 409–420, Santa Fe, NM, 19–21 June 1989. ACM Press, New York, 1989.
- [427] MAREK KARPINSKI AND WOJCIECH RYTTER. An alphabet-independent optimal parallel search for three dimensional pattern. In Maxime Crochemore and Dan Gusfield, editors, *Combinatorial Pattern Matching: 5th Annual Symposium. Proceedings*, volume 807 of *Lecture Notes in Computer Science*, pages 125–135, Asilomar, CA, 5–8 June 1994. Springer-Verlag, Berlin, 1994.
- [428] MARK GIRSHEVICH KARPOVSKY. Fast Fourier transforms on finite non-Abelian groups. *IEEE Transactions on Computers*, **C-26**(10):1028–1030, October 1977.
- [429] MARK GIRSHEVICH KARPOVSKY AND E.A. TRACHTENBERG. Some optimization problems for convolution systems over finite groups. *Information and Control*, **34**(3):227–247, July 1977.
- [430] MARK GIRSHEVICH KARPOVSKY AND E.A. TRACHTENBERG. Fourier transform over finite groups for error detection and error correction in computation channels. *Information and Control*, **40**(3):335–358, March 1979.
- [431] MARK GIRSHEVICH KARPOVSKY, E.A. TRACHTENBERG, AND TATYANA D. ROZINER. *Computation of discrete Fourier transforms over finite Abelian groups using pipelined and systolic array architectures*, volume 3, pages 181–188. Birkhäuser, Boston, 1990.
- [432] JACOB KATZENELSON. Computational structure of the N-body problem. *SIAM Journal on Scientific and Statistical Computing*, **10**(4):787–815, July 1989.

- [433] SHIVNANDAN D. KAUSHIK, CHUA-HUANG HUANG, ROBERT W. JOHNSON, AND P. SADAYAPPAN. A methodology for generating efficient disk-based algorithms from tensor product formulas. In Utpal Banerjee, David Gelernter, Alexandru Nicolau, and David A. Padua, editors, *Languages and Compilers for Parallel Computing: Sixth International Workshop: Proceedings*, volume 768 of *Lecture Notes in Computer Science*, pages 358–373, Portland, OR, 12–14 August 1993. Springer-Verlag, Berlin/Heidelberg, 1994.
- [434] SHIVNANDAN D. KAUSHIK, SANJAY SHARMA, AND CHUA-HUANG HUANG. An algebraic theory for modeling multistage interconnection networks. *Journal of Information Science and Engineering*, **9**(1):1–26, March 1993.
- [435] SHIVNANDAN D. KAUSHIK, SANJAY SHARMA, CHUA-HUANG HUANG, JEREMY R. JOHNSON, ROBERT W. JOHNSON, AND P. SADAYAPPAN. An algebraic theory for modeling direct interconnection networks. In *Proceedings of Supercomputing '92*, pages 488–497, Minneapolis, MN, 16–20 November 1992. IEEE, ACM, IEEE Computer Society Press, Los Alamitos, CA 1992.
- [436] WILLIAM BRUNNER KEHL. Automatic data processing for the legal profession. In Walter F. Freiberger and William Prager, editors, *Applications of Digital Computers*, pages 42–57. Ginn and Company, Boston, 1963.
- [437] HEMACHANDRA B. KEKRE, MEGHANAD D. WAGH, AND SHARAD V. KANETKAR. On group theoretic transforms and the automorphism groups. *Information and Control*, **41**(2):147–155, May 1979.
- [438] KEN KENNEDY. Is parallel computing dead? *Parallel Computing Research*, **2**(4):2–, October 1994.
- [439] ADALBERT KERBER. *Algebraic Combinatorics Via Finite Group Actions*. Bibliographisches Institut & F.A. Brockhaus AG, Mannheim, 1991.
- [440] LESLIE ROBERT KERR. *The Effect of Algebraic Structure on the Computational Complexity of Matrix Multiplication*. PhD thesis, Cornell University, Department of Computer Science, Ithaca, NY, June 1970.
- [441] PETER BERNARD KESSLER. Discovering machine-specific code improvements. In *Proceedings of the SIGPLAN '84 Symposium on Compiler Construction*, pages 249–254, Montréal, Canada, 17–22 June 1984. Association for Computing Machinery, New York, 1984.
- [442] WAFAA KHALIL AND ROBERT F.C. WALTERS. An imperative language based on distributive categories. II. *Informatique Theorique et Applications*, **27**(6):503–522, 1993.
- [443] RONI KHARDON. On using the Fourier transform to learn Disjoint DNF. *Information Processing Letters*, **49**(5):219–222, 11 March 1994.
- [444] STEPHEN COLE KLEENE. *Representation of events in nerve nets and finite automata*, volume 34 of *Ann. Math. Stud.*, pages 3–41. Princeton University Press, Princeton, NJ, 1956.
- [445] JOSEF KLING AND BERNHARD HORWITZ. *Chess Studies, or Endings of Games. - Containing Upwards of Two Hundred Scientific Examples of Chess Strategy*. C. J. Skeet, Charing Cross, England, 1851.
- [446] OLIVER KNILL AND ROMAN E. MÄDER. The rotation group of Rubik's cube. *STGSAM Bulletin*, **21**(3):33–43, 1987.

- [447] DONALD ERVIN KNUTH. *The Art of Computer Programming: Volume 2: Semi-numerical Algorithms*. Addison-Wesley Series in Computer Science and Information Processing, Addison-Wesley, Reading, MA, 2nd edition, 1981.
- [448] DONALD ERVIN KNUTH, JR. JAMES H. MORRIS, AND VAUGHAN R. PRATT. Fast pattern matching in strings. *SIAM Journal on Computing*, **6**(2):323–350, June 1977.
- [449] DONALD ERVIN KNUTH AND RONALD W. MOORE. An analysis of alpha-beta pruning. *Artificial Intelligence*, **6**(4):293–326, Winter 1975.
- [450] PETER MICHAEL KOGGE AND HAROLD STUART STONE. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*, **C-22**(8):786–792, August 1973.
- [451] GALINA I. KOLESOVA, CLEMENT WING HONG LAM, AND LARRY THIEL. On the number of 8*8 Latin squares. *Journal of Combinatorial Theory, Series A*, **54**(1):143–148, May 1990.
- [452] E.A. КОМИССАРЧИК AND AARON L. FUTER. Computer analysis of a queen endgame. *International Computer Chess Association Journal*, **9**(4):189–198, December 1986.
- [453] Э. А. КОМИССАРЧИК, АРОН Л. ФУТЕР [E. A. КОМИССАРЧИК, AND AARON L. FUTER]. Об анализе ферзевого эндшпиля при помощи ЭВМ [Analysis of a queen endgame on an IBM computer]. Проблемы Кибернетики [*Problemy Kibernetiki*], **29**:211–220, 1974.
- [454] DANNY КОРЕС, BRENT LIBBY, AND CHRIS COOK. The endgame king, rook and bishop vs. king and rook (KRBKR). In *Proceedings: 1988 Spring Symposium Series: Computer Game Playing*, pages 60–61, Stanford University, Stanford, CA, 22–24 March 1988. American Association for Artificial Intelligence.
- [455] ALEKSEY GRIGORYEVICH КОРНИН. The exploitation of special positional features in endings with the material: Rook and knight against bishop and knight gbr class 0134. *EG*, **5**(74):221–?, 1983.
- [456] ALEKSEY GRIGORYEVICH КОРНИН. Some special features of the endgame struggle Rook and Knight against 2 Knights (GBR class 0107). *EG*, **5**(70):89–92, January 1983.
- [457] RICHARD EARL KORF. Iterative-deepening-A*: an optimal admissible tree search. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1034–1036, Los Angeles, CA, 18–23 August 1985. Morgan Kaufmann, Los Altos, 1985.
- [458] RICHARD EARL KORF. Planning as search: a quantitative approach. *Artificial Intelligence*, **33**(1):65–88, September 1987.
- [459] DAVID G. KORN AND JULES J. LAMBIOTTE, JR. Computing the fast Fourier transform on a vector computer. *Mathematics of Computation*, **33**(147):977–992, July 1979.
- [460] SAMBASIVA RAO KOSARAJU. *Computations on Iterative Automata*. PhD thesis, University of Pennsylvania, Department of Electrical Engineering, August 1969.
- [461] SAMBASIVA RAO KOSARAJU. Efficient tree pattern matching. In *30th Annual Symposium on Foundations of Computer Science*, pages 178–183, Research Triangle Park, NC, 30 October–1 November 1989. IEEE Computer Society Press, Los Alamitos, CA, 1989.
- [462] KAMALA KRITHIVASAN AND R. SITALAKSHMI. Efficient two-dimensional pattern matching in the presence of errors. *Information Sciences*, **43**:169–184, 1987.

- [463] DAVID WILLIAM KRUMME AND DAVID H. ACKLEY. A practical method for code generation based on exhaustive search. In *Proceedings of the SIGPLAN '82 Symposium on Compiler Construction*, pages 185–196, Boston, MA, 23–25 June 1982. Association for Computing Machinery, New York, 1982.
- [464] CLYDE P. KRUSKAL, LARRY RUDOLPH, AND MARC SNIR. The power of parallel prefix. In Douglas Degroot, editor, *Proceedings of the Fourteenth International Conference on Parallel Processing*, pages 180–185, Pennsylvania State University, 20–23 August 1985. IEEE Computer Society Press, Washington, DC, 1985.
- [465] CLYDE P. KRUSKAL, LARRY RUDOLPH, AND MARC SNIR. A complexity theory of efficient parallel algorithms. *Theoretical Computer Science*, **71**(1):95–132, 13 March 1990.
- [466] CLYDE P. KRUSKAL, LARRY RUDOLPH, AND MARC SNIR. Efficient parallel algorithms for graph problems. *Algorithmica*, **5**(1):43–64, 1990.
- [467] PRIYALAI KULASINGHE AND SAID BETTAYEB. On the multiply-twisted hypercube. In Michel Cosnard, Afonso Ferreira, and Joseph Peters, editors, *Proceedings of the First Canada-France Conference on Parallel and Distributed Computing: Theory and Practice*, volume 805 of *Lecture Notes in Computer Science*, Montréal, Canada, 19–21 May 1994. Springer-Verlag, Berlin, 1994.
- [468] VIPIN KUMAR AND ANSHUL GUPTA. Analyzing scalability of parallel algorithms and architectures. Technical Report Preprint 92-020, Army High Performance Computing Research Center, University of Minnesota, 1100 Washington Avenue South, Minneapolis, MN 55415, January 1992.
- [469] HSING-TSUNG KUNG AND JASPAL SUBHLOK. A new approach for automatic parallelization of block linear algebra computation. In *Proceedings of Supercomputing '91*, Albuquerque, NM, 18–22 November 1991. IEEE Computer Society Press, Los Alamitos, CA 1991.
- [470] SUSUMU KUNO AND ANTHONY G. OETTINGER. Multiple-path syntactic analyzer. In *Information Processing: Proceedings of the IFIP Congress*, Munich, 1962. North-Holland, Amsterdam.
- [471] [CARL KUPFFER]. Erinnerungen an Friedrich Amelung [Memories of Friedrich Amelung]. *Düna-Zeitung (Feuilleton-Beilage: Für Haus und Familie)*, **93**:100–101, 25 April–8 May 1909.
- [472] EYAL KUSHILEVITZ AND YISHAY MANSOUR. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, **22**(6):1331–1348, December 1993.
- [473] BRADLEY C. KUSZMAUL. *Synchronized MIMD Computing*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1994.
- [474] KIM KVECH AND BRYANT WHITTIER YORK. Permutation group computations on the Connection Machine CM-5. Technical Report NU-CCS-92-7, Northeastern University, Boston, MA, 1992.
- [475] RICHARD EMIL LADNER AND MICHAEL JOHN FISCHER. Parallel prefix computation. *Journal of the Association for Computing Machinery*, **27**(4):831–838, October 1980.
- [476] JOHN D. LAFFERTY AND DANIEL NAHUM ROCKMORE. Fast Fourier analysis for SL_2 over a finite field and related numerical experiments. *Experimental Mathematics*, **1**(2):115–139, 1992.
- [477] ROBERT LAKE, PAUL LU, AND JONATHAN SCHAEFFER. Using retrograde analysis to solve combinatorial search spaces. In E. D. Brooks and K. H. Warren, editors, *The 1992 MPCP Yearly Report*. Lawrence Livermore National Laboratory, 1992.

- [478] ROBERT LAKE, JONATHAN SCHAEFFER, AND PAUL LU. Solving large retrograde-analysis problems using a network of workstations. In H. Jaap van den Herik, I.S. Herschberg, and Jos W.H.M. Uiterwijk, editors, *Advances in Computer Chess 7*, pages 135–162. University of Limburg, Maastricht, Netherlands, 1994.
- [479] CLEMENT WING HONG LAM. Computational combinatorics—a maturing experimental approach. Course notes, December 1990.
- [480] CLEMENT WING HONG LAM. Application of group theory to combinatorial searches. In Larry Finkelstein and William Kantor, editors, *Groups and Computation: Workshop on Groups and Computation*, volume 11 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 133–138, Rutgers, NJ, 7–10 October 1991. American Mathematical Society, Providence, RI, 1993.
- [481] CLEMENT WING HONG LAM. The search for a finite projective plane of order 10. *American Mathematical Monthly*, **98**(4):305–318, April 1991.
- [482] CLEMENT WING HONG LAM, GALINA I. KOLESOVA, AND LARRY THIEL. A computer search for finite projective planes of order 9. *Discrete Mathematics*, **92**(1-3):187–195, 17 Nov 1991.
- [483] CLEMENT WING HONG LAM AND LARRY THIEL. Backtrack search with isomorph rejection and consistency check. *Journal of Symbolic Computation*, **7**(5):473–485, May 1989.
- [484] GAD M. LANDAU, BARUCH SCHIEBER, AND UZI VISHKIN. Parallel construction of a suffix tree. In *Automata, Languages, and Programming: 14th International Colloquium*, volume 267 of *Lecture Notes in Computer Science*, pages 314–325, Karlsruhe, Germany, 13–17 July 1987. Springer-Verlag, Berlin, 1987.
- [485] GAD M. LANDAU AND UZI VISHKIN. Efficient string matching in the presence of errors. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 125–136, Portland, OR, 21–23 October 1985. IEEE Computer Society Press, Washington, DC./Los Angeles, CA, 1985.
- [486] GAD M. LANDAU AND UZI VISHKIN. Efficient string matching with k mismatches. *Theoretical Computer Science*, **43**:239–249, 1986.
- [487] GAD M. LANDAU AND UZI VISHKIN. Introducing efficient parallelism into approximate string matching and a new serial algorithm. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 220–230, Berkeley, CA, 28–30 May 1986. ACM Press, New York, 1986.
- [488] GAD M. LANDAU AND UZI VISHKIN. Fast string matching with k differences. *Journal of Computer and System Sciences*, **37**(1):63–78, August 1988.
- [489] GAD M. LANDAU AND UZI VISHKIN. Fast parallel and serial approximate string matching. *Journal of Algorithms*, **10**(2):157–169, June 1989.
- [490] GAD M. LANDAU AND UZI VISHKIN. Two dimensional pattern matching in a digitized image. In Alberto Apostolico, Maxime Crochemore, Zvi Galil, and Udi Manber, editors, *Combinatorial Pattern Matching: 4th Annual Symposium. Proceedings*, volume 684 of *Lecture Notes in Computer Science*, Padova, Italy, 2–4 June 1993. Springer-Verlag, Berlin, 1993.
- [491] GAD M. LANDAU, UZI VISHKIN, AND RUTH NUSSINOV. An efficient string matching algorithm with k differences for nucleotide and amino acid sequences. *Nucleic Acids Research*, **14**(1):31–46, 10 January 1986.

- [492] SERGE LANG. *Algebra*. World Student Series. Addison-Wesley, Reading, MA, 1965.
- [493] GUY LAPALME. Arrays in Haskell. In Lenore M. Restifo Mullin et. al., editor, *Arrays, functional languages and parallel systems*. Kluwer Academic Publishers, Boston, MA, 1991.
- [494] LOREN C. LARSON. A bibliography on the mathematical aspects of chess, 1974.
- [495] SIMON HUGH LAVINGTON. *Early British Computers: The Story of Vintage Computers and the People Who Built Them*. Digital Press, Bedford, MA, 1980.
- [496] CHARLES LAWRENCE LAWSON, RICHARD J. HANSON, DAVID RONALD KINCAID, AND FREDERICK THOMAS KROGH. Basic linear algebra subprograms for Fortran usage. *ACM Transactions on Mathematical Software*, **5**(3):308–323, September 1979.
- [497] R.Y. LECHNER. Harmonic analysis of switching functions. In Amar Mahkopadhyay, editor, *Recent Developments in Switching Theory*, Electrical Science. Academic Press, New York, 1971.
- [498] JONG LEE, EUGENE SHRAGOWITZ, AND SARTAJ KUMAR SAHNI. A hypercube algorithm for the 0/1 knapsack problem. *Journal of Parallel and Distributed Computing*, **5**(4), August 1988.
- [499] DANIEL J. LEHMANN. Algebraic structures for transitive closure. *Theoretical Computer Science*, **4**(1):59–76, February 1977.
- [500] FRANK THOMSON LEIGHTON. *Complexity issues in VLSI: optimal layouts for the shuffle-exchange graph and other networks*. Foundations of Computing. MIT Press, Cambridge, MA, 1983.
- [501] FRANK THOMSON LEIGHTON. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. M. Kaufmann Publishers, San Mateo, CA, 1992.
- [502] CHARLES ERIC LEISERSON. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, **C-34**(10):892–901, October 1985.
- [503] CHARLES ERIC LEISERSON, ZAHİ S. ABUHAMDEH, DAVID C. DOUGLAS, CARL R. FEYNMAN, MAHESH N. GANMUKHI, JEFFREY V. HILL, WILLIAM DANIEL HILLIS, BRADLEY C. KUSZMAUL, MARGARET A. ST. PIERRE, DAVID S. WELLS, MONICA C. WONG, SHAW-WEN YANG, AND ROBERT ZAK. The network architecture of the CM-5. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 272–285, San Diego, CA, 29 June–1 July 1992. ACM, EATCS, ACM Press, New York, 1992.
- [504] THOMAS LENGAUER AND D. THEUNE. Unstructured path problems and the making of semirings. In Frank Dehne, Jörg-Rüdiger Sack, and Nicola Santoro, editors, *Algorithms and Data Structures: 2nd Workshop, WADS '91*, volume 519 of *Lecture Notes in Computer Science*, pages 189–200, Ottawa, Canada, 14–16 August 1991. Berlin, Springer Verlag, 1991.
- [505] REINER LENZ. Group invariant pattern recognition. *Pattern Recognition*, **23**(1–2):199–217, 1990.
- [506] REINER LENZ. *Group Theoretical Methods in Image Processing*, volume 413 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin/Heidelberg, 1990.
- [507] Wilhem Lenz, editor. *Deutschbaltisches biographisches Lexikon 1710–1960* [German-Baltic biographical dictionary 1710–1960]. Böhlau Verlag, Köln, 1970.
- [508] J. LEON. Computing automorphism groups of combinatorial objects. In Michael D. Atkinson, editor, *Computational Group Theory*, pages 321–337. Academic Press, London, England, 1984.

- [509] GIL LERMAN AND LARRY RUDOLPH. *Parallel Evolution of Parallel Processors*. Frontiers of Computer Science: Surveys in Computer Science. Plenum Press, New York, 1993.
- [510] WOODY LICHTENSTEIN AND S. LENNART JOHNSON. Block-cyclic dense linear algebra. *SIAM Journal on Scientific Computing*, **14**(6):1259–1288, November 1993.
- [511] DAVID JOHN LILJA. Exploiting parallelism available in loops. *Computer*, **27**(2):13–26, February 1994.
- [512] CALVIN LIN AND LAWRENCE SNYDER. ZPL: An array sublanguage. In Utpal Banerjee, David Gelernter, Alexandru Nicolau, and David A. Padua, editors, *Languages and Compilers for Parallel Computing: Sixth International Workshop: Proceedings*, volume 768 of *Lecture Notes in Computer Science*, pages 96–114, Portland, OR, 12–14 August 1993. Springer-Verlag, Berlin/Heidelberg, 1994.
- [513] J. LIN AND JAMES ANDREW STORER. Processor-efficient hypercube algorithms for the knapsack problem. *Journal of Parallel and Distributed Computing*, **13**(3):332–337, November 1991.
- [514] NATHAN LINIAL, YISHAY MANSOUR, AND NOAM NISAN. Constant depth circuits, Fourier transform, and learnability. In *30th Annual Symposium on Foundations of Computer Science*, pages 574–580, Research Triangle Park, NC, 30 October–1 November 1989. IEEE Computer Society Press, Los Alamitos, CA, 1989.
- [515] RICHARD J. LIPTON AND YECHESKEL ZLACSTEIN. Word problems solvable in logspace. *Journal of the Association for Computing Machinery*, **24**(3):522–526, July 1977.
- [516] PANGFENG LIU AND SANDEEP NAUTAM BHATT. Experiences with parallel N-body simulation. In *6th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 122–131, Cape May, New Jersey, June 27–29 1994. ACM Press, New York, 1994.
- [517] B.F. LOGAN AND L.A. SHEPP. A variational problem for Young tableaux. *Advances in Mathematics*, **26**(2):206–222, November 1977.
- [518] GIOVANNI BATTISTA LOLLI. *Osservazione teorico-pratiche sopra il giuoco degli scacchi* [theoretical-practical observations on the game of chess]. Bologna, 1763.
- [519] PETER S. LOMDAHL AND DAVID M. BEAZLEY. State-of-the-art parallel computing: molecular dynamics on the Connection Machine. *Los Alamos Science*, **22**:44–57, 1994.
- [520] MARGREET LOUTER-NOOL. Basic linear algebra subprograms (BLAS) on the CDC CYBER 205. *Parallel Computing*, **4**(2):143–166, April 1987.
- [521] MARGREET LOUTER-NOOL. LINPACK routines based on level 2 BLAS. *Journal of Supercomputing*, **3**(4):331–349, December 1989.
- [522] L.D.J.C. LOYENS AND J.R. MOONEN. ILIAS, a sequential language for parallel matrix computations. In Costas Halatsis, editor, *PARLE '94: Parallel Architectures and Languages Europe: 6th International PARLE Conference: Proceedings*, volume 817 of *Lecture Notes in Computer Science*, Athens, Greece, 4–8 July 1994. Springer-Verlag, Berlin/New York, 1994.
- [523] EUGENE MICHAEL LUKS. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, **25**(1):42–65, August 1982.
- [524] EUGENE MICHAEL LUKS. Parallel algorithms for permutation groups and graph isomorphism. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 292–302, Toronto, Ontario, 27–29 October 1986. IEEE Computer Society Press, Washington, DC./Los Angeles, CA, 1986.

- [525] EUGENE MICHAEL LUKS. Parallel algorithms for permutation groups and graph isomorphism. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 292–302, Toronto, Ontario, 27–29 October 1986. IEEE Computer Society Press, Washington, DC / Los Angeles, CA, 1986.
- [526] EUGENE MICHAEL LUKS AND PIERRE MCKENZIE. Parallel algorithms for solvable permutation groups. *Journal of Computer and System Sciences*, **37**(1):39–62, August 1988.
- [527] ROGER C. LYNDON AND M.P. SCHÜTZENBERGER. The equation $a^M = b^N c^P$ in a free group. *Michigan Math. Journal*, **9**:289–298, 1962.
- [528] MICHAEL G. MAIN AND RICHARD JOSEPH LORENTZ. An $O(n \log n)$ algorithm for finding all repetitions in a string. *Journal of Algorithms*, **5**(3):422–432, September 1984.
- [529] ERKKI MÄKINEN. On the subtree isomorphism problem for ordered trees. *Information Processing Letters*, **32**(5):271–273, 22 September 1989.
- [530] MARVIN MARCUS. *Finite Dimensional Multilinear Algebra: Part 1*. Marcel Dekker, Inc., New York, 1973.
- [531] MARVIN MARCUS. *Finite Dimensional Multilinear Algebra: Part 2*. Marcel Dekker, Inc., New York, 1973.
- [532] PETER D. MARK. Parallel computation of Sylow subgroups in solvable groups. In Larry Finkelstein and William Kantor, editors, *Groups and Computation: Workshop on Groups and Computation*, volume 11 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 177–187, Rutgers, NJ, 7–10 October 1991. American Mathematical Society, Providence, RI, 1993.
- [533] THOMAS ANTHONY MARSLAND. Workshop report: Theory and practice in computer chess. *International Computer Chess Association Journal*, **10**(4), December 1987.
- [534] THOMAS ANTHONY MARSLAND AND MURRAY S. CAMPBELL. Parallel search of strongly ordered game trees. *Computing Surveys*, **14**(4):533–551, December 1982.
- [535] THOMAS ANTHONY MARSLAND AND FRED POPOWICH. Parallel game tree search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **7**(4):442–452, July 1985.
- [536] DAVID KEITH MASLEN AND DANIEL NAHUM ROCKMORE. Separation of variables and the computation of Fourier transforms on finite groups, I. Manuscript, 15 November 1994.
- [537] DAVID KEITH MASLEN AND DANIEL NAHUM ROCKMORE. Adapted diameters and the efficient computation of Fourier transforms on finite groups. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 253–262, San Francisco, CA, 22–24 January 1995. Association for Computing Machinery/Society for Industrial and Applied Mathematics, New York/Philadelphia, 1995.
- [538] HENRY MASSALIN. Superoptimizer—a look at the smallest program. In *Proceedings of the Second International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 122–126, Palo Alto, CA, 1987.
- [539] ALEKSANDAR MATANOVIĆ. *Enciklopedija Šahovskih Završnica* [Encyclopedia of Chess Endings]. Šahovski informator, Beograd, 1985.
- [540] KAPIL K. MATHUR AND S. LENNART JOHNSON. All-to-all communication on the Connection Machine CM–200. Technical Report TMC–TR–243, Thinking Machines Corporation, Cambridge, MA, 1992.

- [541] ERNST W. MAYR. Basic parallel algorithms in graph theory. In Gottfried Tinhofer, Ernst W. Mayr, Hartmut Noltemeier, Maciej M. Syslo, and Rudolf Albrecht, editors, *Computational Graph Theory*, volume 7 of *Computing Supplementum*, pages 67–90. Springer-Verlag, Vienna, 1990.
- [542] EDWARD M. MCCREIGHT. A space-economical suffix tree construction algorithm. *Journal of the Association for Computing Machinery*, **23**(2):262–272, April 1976.
- [543] PIERRE MCKENZIE AND STEPHEN ARTHUR COOK. The parallel complexity of the Abelian permutation group membership problem. In *24th Annual Symposium on Foundations of Computer Science*, pages 1–10, Tucson, AZ, 7–9 November 1983. IEEE Computer Society Press, Silver Spring, MD, 1983.
- [544] EDMAR MEDNIS. Endgames with minor pieces. *New In Chess*, **1987/88**:86–97, 1987.
- [545] EDMAR MEDNIS. Endgames with minor pieces, part 2. *New in Chess*, pages 56–59, 1988.
- [546] RUSSELL MERRIS. Manifestations of Pólya’s counting theorem. *Linear Algebra and Applications*, **32**:209–234, August 1980.
- [547] RUSSELL MERRIS. Pattern inventories associated with symmetry classes of tensors. *Linear Algebra and Applications*, **29**:225–230, February 1980.
- [548] RUSSELL MERRIS. Pólya’s counting theorem via tensors. *American Mathematical Monthly*, **88**(3):179–185, March 1981.
- [549] RUSSELL MERRIS. Applications of multilinear algebra. *Linear and Multilinear Algebra*, **32**(3–4):211–224, 1992.
- [550] RUSSELL MERRIS AND WILLIAM WATKINS. Tensors and graphs. *SIAM Journal on Algebraic and Discrete Methods*, **4**(4):534–547, December 1983.
- [551] GERARD GILBERT LOUIS MEYER AND LOUIS J. PODRAZIK. A parallel first-order linear recurrence solver. Technical Report JHU/EECS-86/07, Electrical Engineering and Computer Science Department, The Johns Hopkins University, Baltimore, MD 21218, 1986.
- [552] ALBERT ABRAHAM MICHELSON AND SAMUEL WESLEY STRATTON. On a new harmonic analyzer. *London, Edinburgh and Dublin Philosophical Magazine*, Series 5, **45**(272):85–91, January 1898.
- [553] DONALD MICHIE AND IVAN BRATKO. Ideas on knowledge synthesis stemming from the KB-BKN endgame. *International Computer Chess Association Journal*, **10**(1):3–10, March 1987.
- [554] GERHARD O. MICHLER. Representations of groups over finite fields. In Rainer Janßen, editor, *Trends in Computer Algebra: International Symposium Bad Neuenahr, May 19–21, 1987: Proceedings*, volume 296 of *Lecture Notes in Computer Science*, pages 119–133. Springer-Verlag, Berlin, 1988.
- [555] TORSTEN MINKWITZ. *Algorithmensynthese für lineare Systeme mit Symmetrie* [Synthesizing algorithms for linear systems]. PhD thesis, Fakultät für Informatik der Universität Karlsruhe (Technische Hochschule), 9 June 1993.
- [556] Федор Эдуардович Молин [THEODOR MOLIEN]. Berichtigung zu dem Aufsätze ‘Ueber Systeme höherer complexer Zahlen’ [Correction to the article ‘on systems of higher complex numbers’]. *Mathematische Annalen*, **42**:308–312, 1893.

- [557] Федор Эдуардович Молин [THEODOR MOLIEN]. Ueber Systeme höherer complexer Zahlen [On systems of higher complex numbers]. *Mathematisch Annalen*, **41**:83–156, 1893.
- [558] Федор Эдуардович Молин [THEODOR MOLIEN]. Eine Bemerkung zur Theorie der homogenen Substitutionsgruppen [A remark on the theory of homogeneous substitution groups]. *Sitzungsberichte Naturforscher-Gesellschaft Dorpat (Yurev in Estonia)*, **11**:259–274, 24 April 1897.
- [559] Федор Эдуардович Молин [THEODOR MOLIEN]. Ueber di Anzahl der Variabeln einer irreductibelen Substitutionsgruppen [On the number of variables of irreducible substitution groups]. *Sitzungsberichte der ber Naturforscher-Gesellschaft Dorpat*, **11**:277–288, 1897.
- [560] Федор Эдуардович Молин [THEODOR MOLIEN]. Ziffermässig genaue Ausrechnung aller 12 millionen Gewinne und Remisen im Endspiel „Thurm gegen Läufer“ [Numerical exact computation of all 12 million wins and draws in the endgame “rook against bishop”], April 1897. Cited in the index of *Baltische Schachblätter* **8**, 1901, page 72.
- [561] Федор Эдуардович Молин [THEODOR MOLIEN]. [four endgame studies]. *Baltische Schachblätter*, **6**:208–209, 1898.
- [562] Федор Эдуардович Молин [THEODOR MOLIEN]. Ueber die Invarianten der linearen Substitutionsgruppen [On the invariants of linear substitution groups]. *Sitzungsberichte Akademie der Wissenschaft. Berlin*, pages 1152–1156, 1898.
- [563] [Федор Эдуардович Молин [THEODOR MOLIEN]]. [partie 73]. *Baltische Schachblätter*, **7**:346, 1900. Score of game against W. Sohn.
- [564] Bernhard Möller, Helmut A. Partsch, and Stephen A. Schuman, editors. *Formal Program Development: IFIP TC2/WG 2.1 State-of-the-Art Report*, volume 755 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1993.
- [565] EDUARDO MORALES. Learning chess patterns. In Stephen Muggleton, editor, *Proceedings of International Workshop on Inductive Logic Programming*, pages 291–307, March 2–4 1991.
- [566] SHLOMO MORAN. A note on ‘Is shortest path problem not harder than matrix multiplication?’. *Information Processing Letters*, **13**(2):85–86, 13 November 1981.
- [567] Z. GEORGE MOU AND XIAOJING WANG. Optimal mappings of m dimensional FFT communication to k dimensional mesh for arbitrary m and k . In Arndt Bode, Mike Reeve, and Gottfried Wolf, editors, *PARLE '93: Parallel Architectures and Languages Europe: 5th International PARLE Conference*, volume 694 of *Lecture Notes in Computer Science*, pages 104–119, Munich, Germany, 14–17 June 1993. Springer-Verlag, Berlin, 1993.
- [568] LENORE M. RESTIFO MULLIN. A mathematics of arrays. Technical Report 8814, CASE Center, Syracuse University, Syracuse, NY 13244, December 1988.
- [569] LENORE M. RESTIFO MULLIN. Psi, the indexing function: a basis for FFP with arrays. In Lenore M. Restifo Mullin et. al., editor, *Arrays, functional languages and parallel systems*, pages 185–200. Kluwer Academic Publishers, Boston, MA, 1991.
- [570] HANS MUNTHE-KAAS. Superparallel FFTs. *SIAM Journal on Scientific and Statistical Computing*, **14**(2):349–367, March 1993.
- [571] BERNIE G.J.P.T. MURRAY, PAUL ANTHONY BASH, AND MARTIN KARPLUS. Molecular dynamics on the Connection Machine system. Technical Report CB88, Thinking Machines Corporation, Cambridge, MA, May 1988.

- [572] HAROLD JAMES RUTHVEN MURRAY. *A History of Chess*. Oxford University Press, London, 1913.
- [573] HAROLD JAMES RUTHVEN MURRAY. *A History of Board-Games Other Than Chess*. Hacker Art Books, New York, 1978. First published 1952 by Oxford University at the Clarendon Press.
- [574] S. MUTHUKRISHNAN. *Searching For Strings and Searching in Presence of Errors*. PhD thesis, Department of Computer Science, New York University, July 1994.
- [575] S. MUTHUKRISHNAN AND RAMESH HARIHARAN. String matching under a general match relation. In Rudrapatna Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science: 12th Conference: Proceedings*, volume 652 of *Lecture Notes in Computer Science*, pages 356–367, New Delhi, India, 18–20 December 1992. Springer-Verlag, Berlin, 1992.
- [576] S. MUTHUKRISHNAN AND KRISHNA PALEM. Non-standard stringology: algorithms and complexity. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 770–779, Montréal, Québec, Canada, 23–25 May 1994. ACM Press, New York, 1994.
- [577] PAUL EDWARD NACOZY AND ROGER EARL DIEHL. A semianalytical theory for the long-term motion of Pluto. *Astronomical Journal*, **83**(5):522–530,, May 1978.
- [578] DAVID NASSIMI AND SARTAJ KUMAR SAHNI. Parallel permutation and sorting algorithms and a new generalized connection network. *Journal of the Association for Computing Machinery*, **29**(3):642–667, July 1982.
- [579] HARRY J. NEFKENS. Constructing data bases to fit a microcomputer. *International Computer Chess Association Journal*, **8**(4):217–224, December 1985.
- [580] DOUGLAS GEOFFREY NORTHCOTT. *Multilinear algebra*. Cambridge University Press, New York, 1984.
- [581] JOHN NUNN. *Secrets of Rook Endings*. Batsford Chess Library. H. Holt, New York, 1992.
- [582] JOHN NUNN. Extracting information from endgame databases. *International Computer Chess Association Journal*, **16**(4):191–200, December 1993.
- [583] JOHN NUNN. *Secrets of Pawnless Endings*. Batsford Chess Library. H. Holt, New York, 1994.
- [584] R. OHBUCHI. Overview of parallel processing in Japan. *Parallel Computing*, **2**:219–228, 1985.
- [585] VICTOR PAN. *How To Multiply Matrices Faster*, volume 179 of *Lecture notes in Computer Science*. Springer-Verlag, Berlin/New York, 1984.
- [586] VICTOR PAN. Complexity of parallel matrix computations. *Theoretical Computer Science*, **54**(1):65–85, September 1987.
- [587] VICTOR PAN. Parallel solution of sparse linear and path systems. In John H. Reif, editor, *Synthesis of Parallel Algorithms*, pages 621–678. Morgan Kaufmann Publishers, San Mateo, CA, 1993.
- [588] VICTOR PAN AND JOHN H. REIF. Efficient parallel solution of linear systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 143–152, Providence, R.I., 6–8 May 1985. Association for Computing Machinery, New York/Baltimore, 1985.
- [589] VICTOR PAN AND JOHN H. REIF. Parallel nested dissection for path algebra computations. *Operations Research Letters*, **5**(4):177–184, 1986.

- [590] EMANUEL PARZEN. Informal comments on the uses of power spectrum analysis. *IEEE Transactions on Audio and Electroacoustics*, **AU-15**(2):75–76, June 1967.
- [591] OREN PATASHNIK. Qubic: $4 \times 4 \times 4$ tic-tac-toe. *Mathematics Magazine*, **53**(4):203–216, September 1980.
- [592] DAVID ANDREW PATTERSON AND DAVID ROGER DITZEL. The case for the reduced instruction set computer. *Computer Architecture News*, **8**(6):25–33, October 1980.
- [593] DAVID ANDREW PATTERSON, GARTH GIBSON, AND RANDY H. KATZ. A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of the SIGMOD International Conference on Management of Data*, pages 109–116, Chicago, IL, 1–3 June 1988.
- [594] JUDEA PEARL. Optimal dyadic models of time-invariant systems. *IEEE Transactions on Computers*, **C-24**(6):598–603, June 1975.
- [595] JUDEA PEARL. Asymptotic properties of minimax trees and game-searching procedures. *Artificial Intelligence*, **14**(2):113–138, September 1980.
- [596] JUDEA PEARL. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.
- [597] BARAK A. PEARLMUTTER. Fast exact multiplication by the Hessian (of a neural net). *Neural Computation*, **6**(1):147–160, January 1994.
- [598] MARSHALL CARLETON PEASE. An adaptation of the fast Fourier transform for parallel processing. *Journal of the Association for Computing Machinery*, **15**(2):252–264, April 1968.
- [599] CRISPIN PERDUE AND HANS JACK BERLINER. EG—a case study in problem solving with King and Pawn endings. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI-77)*, pages 421–427, volume 1, Pittsburgh, PA, 1977. Department of Computer Science, Carnegie-Mellon University. August 22–25, 1977 at the Massachusetts Institute of Technology, Cambridge, MA.
- [600] JUAN BAUTISTA SANCHEZ PEREZ. *El Ajedrez de D. Alfonso el Sabio* [The chess of D. Alfonso the Wise]. Tip. La Franco Española, Madrid, 1929.
- [601] IVARS PETERSON. Computing a chess game’s end. *Science News*, **140**(22), 30 November 1991.
- [602] FRANÇOIS-ANDRÉ DANICAN PHILIDOR. *L’analyse des échecs: contenant une nouvelle methode pour apprendre en peu temps à se perfectionner dans ce noble jeu* [the analysis of chess: containing a new method for learning in a short time and towards perfection in the noble game]. London, [1749?].
- [603] F. PICHLER. On state space description of linear dyadic invariant systems. In R.W. Zeek and A.E. Showalter, editors, *Proceedings of the 2nd Symposium on the Applications of Walsh Functions*, Washington, DC., 13–15 April 1971. National Technical Information Service, Operations Division, Springfield, VA 1971.
- [604] IRA POHL. Bi-directional search. In Bernard Meltzer and Donald Michie, editors, *Machine Intelligence*, pages 127–140. Edinburgh University Press, Edinburgh, Scotland, 1971.
- [605] CONSTANTINE DEMETRIOS POLYCHRONOPOULOS AND DAVID J. KUCK. Guided self-scheduling: A practical scheduling scheme for parallel supercomputers. *IEEE Transactions on Computers*, **36**(12):1425–1439, December 1987.

- [606] RANDALL E. PORTER. The RW-400—a new polymorphic data system. *Datamation*, 6:8–14, January–February 1960.
- [607] CURT POWLEY, CHRIS FERGUSON, AND RICHARD EARL KORF. Parallel tree search on a SIMD machine. In *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing*, pages 249–256, Dallas, Texas, 2–5 December 1991. IEEE Computer Society Press, Los Alamitos, CA, 1991.
- [608] CURT POWLEY AND RICHARD EARL KORF. Single-agent parallel window search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):466–477, May 1991.
- [609] DHIRAJ K. PRADHAN AND MAHESWARA R. SAMATHAM. The de Bruijn multiprocessor network: a versatile parallel processing and sorting network for VLSI. *IEEE Transactions on Computers*, 38(4):567–581, April 1989.
- [610] FRANCO P. PREPARATA AND JEAN VUILLEMIN. The cube-connected cycles: a versatile network for parallel computation. *Communications of the ACM*, 24(5):300–309, May 1981.
- [611] JOHN ROSS QUINLAN. Discovering rules by induction from large collections of examples. In Donald Michie, editor, *Expert systems in the micro-electronic ages*, pages 168–201. Edinburgh University Press, 1979.
- [612] JOHN ROSS QUINLAN. Learning efficient classification procedures and their application to chess end games. In Ryszard Stanislaw Michalski, Jaime Guillermo Carbonell, and Tom Michael Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 463–482. Tioga Publishing Company, Palo Alto, CA, 1983.
- [613] JOHN ROSS QUINLAN. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
- [614] CHARLES M. RADER. Discrete Fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56(6):1107–1108, June 1968.
- [615] V. NAGESHWARA RAO AND VIPIN KUMAR. Parallel depth-first search, part i. Implementation. *International Journal of Parallel Programming*, 16(6):479–499, December 1987.
- [616] LARS RASMUSSEN. Ultimates in KQKR and KQKN. *International Computer Chess Association Journal*, 11(1):21–25, March 1988.
- [617] L. REICHEL. A matrix problem with application to rapid solution of integral equations. *SIAM Journal on Scientific and Statistical Computing*, 11(2):263–280, March 1990.
- [618] MARGARET REID-MILLER AND GUY E. BLELLOCH. List ranking and list scan on the CRAY C-90. Technical Report CMU-CS-94-101, Carnegie Mellon University School of Computer Science, 5000 Forbes Avenue, Pittsburgh, PA 15213–3891, 15 March 1994.
- [619] STEFAN REISCH. Gobang ist PSPACE-vollständig [Gobang is PSPACE-complete]. *Acta Informatica*, 13(1):59–66, January 1980.
- [620] STEFAN REISCH. Hex ist PSPACE-vollständig [Hex is PSPACE-complete]. *Acta Informatica*, 15(2):167–191, December II [1980] 1981.
- [621] KENDALL SQUARE RESEARCH. KSR1 principles of operation. Waltham, MA, 1991.
- [622] HUDSON B. RIBAS. *Automatic generation of systolic programs from nested loops*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, June 1990.

- [623] JOHN L. RICHARDSON. Block LU decomposition on the Connection Machine system. Technical Report NA89-5, Thinking Machines Corporation, Cambridge, MA, October 1989.
- [624] HENRI RINCK. *Las sorpresas de la teoria / Les surprises de la theorie* [Surprises of the theory]. Dossat, Madrid, 1947. Text in Spanish and French.
- [625] HENRI RINCK. *1414 Fins de partie [1414 endgames]*. Tipografia la Academica, Barcelona, 1950. 5th edition of *150 Fins de Partie*, 1909.
- [626] HENRI RINCK AND LOUIS MALPAS. *Dame contre tour et cavalier* [Queen against rook and knight]. L'Echiquier, Bruxelles, 1947.
- [627] J. M. ROBSON. N by N checkers is exptime complete. *SIAM Journal on Computing*, **13**(2):252–267, May 1984.
- [628] DANIEL NAHUM ROCKMORE. Efficient computation of Fourier transforms on the symmetric group. In Erich Kaltofen and Stephen M. Watt, editors, *Computers and Mathematics: Proceedings of Computers and Mathematics '89*, pages 156–165. Springer-Verlag, New York, 1989, Massachusetts Institute of Technology, 13–17 June 1989.
- [629] DANIEL NAHUM ROCKMORE. *Fast Fourier Analysis for Finite Groups*. PhD thesis, Harvard University, Department of Mathematics, Cambridge, MA, May 1989.
- [630] DANIEL NAHUM ROCKMORE. Fast Fourier analysis for abelian group extensions. *Advances in Applied Mathematics*, **11**:164–204, 1990.
- [631] DANIEL NAHUM ROCKMORE. Efficient computation of Fourier inversion for finite groups. *Journal of the Association for Computing Machinery*, **41**(1):31–66, January 1994.
- [632] DANIEL NAHUM ROCKMORE. Fast Fourier transforms for wreath products. Technical Report PMA-TR94-176, Dartmouth College Department of Mathematics, Hanover, NH 03755-3551, July 1994.
- [633] FRANCESCO ROMANI. Shortest-path problem is not harder than matrix multiplication. *Information Processing Letters*, **11**(3):134–136, 18 November 1980.
- [634] FRANCESCO ROMANI. Author's reply to S. Moran's Note on the shortest path problem. *Information Processing Letters*, **13**(2):87, 13 November 1981.
- [635] JOHN R. ROSE AND GUY LEWIS STEELE, JR. C*: an extended C language for data parallel programming. In *Second International Conference on Supercomputing: Proceedings*, volume 2, pages 2–16, San Francisco, CA, 4–7 May 1987. International Supercomputing Institute, St. Petersburg, FL, 1987.
- [636] ARNOLD LEONARD ROSENBERG. Shuffle-oriented interconnection networks. Technical Report COINS 88-84, University of Massachusetts at Amherst Computer and Information Science, 11 October 1988.
- [637] ARNOLD LEONARD ROSENBERG. Cayley graphs and direct-product graphs. In Larry Finkelstein and William Kantor, editors, *Groups and Computation: Workshop on Groups and Computation*, volume 11 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 245–251, Rutgers, NJ, 7–10 October 1991. American Mathematical Society, Providence, RI, 1993.
- [638] FRANK ROSENBLATT. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**(6):386–408, November 1958.

- [639] PHILLIP E. ROSS. Endless endgame? *Scientific American*, **265**(5), November 1991.
- [640] GÜNTER ROTE. Path problems in graphs. In Gottfried Tinhofer, Ernst W. Mayr, Hartmut Noltemeier, Maciej M. Syslo, and Rudolf Albrecht, editors, *Computational Graph Theory*, volume 7 of *Computing Supplementum*, pages 155–190. Springer-Verlag, Vienna, 1990.
- [641] JOSEPH J. ROTMAN. *An Introduction to the Theory of Groups*, volume 148. Springer-Verlag, New York, 4th edition, 1995.
- [642] ARTHUR JOHN ROYCROFT. A note on 2S's v R + B. *EG*, **1**:197–198, April 1967.
- [643] ARTHUR JOHN ROYCROFT. *Test Tube Chess: A Comprehensive Introduction to the Chess Endgame Study*. Faber and Faber Ltd., London, England, 1972.
- [644] ARTHUR JOHN ROYCROFT. A computer program for the ending wP v bB (4 men on the board). *EG*, **3**(40):202–203, May 1975.
- [645] ARTHUR JOHN ROYCROFT. Two bishops against knight. *EG*, **5**(75), April 1983.
- [646] ARTHUR JOHN ROYCROFT. A proposed revision of the '50-move rule': Article 12.4 of the Laws of Chess. *International Computer Chess Association Journal*, **7**(3):164–170, September 1984.
- [647] ARTHUR JOHN ROYCROFT. *C*. *EG*, **6**(98):641–645, October 1989.
- [648] ARTHUR JOHN ROYCROFT. Recent otb experience with GBR class 0312. *EG*, **6**(102 part 2):948–950, May 1992.
- [649] ARTHUR JOHN ROYCROFT. The 6-man pawnless endgame rook and bishop against two knights with the 223-move win. *EG*, **7**(114):496–512, December 1994.
- [650] TATYANA D. ROZINER, MARK GIRSHEVICH KARPOVSKY, AND LAZAR A. TRACHTENBERG. Fast Fourier transforms over finite groups by multiprocessor systems. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **38**(2):226–240, February 1990.
- [651] DAVID E. RUMELHART, GEOFFREY E. HINTON, AND RONALD J. WILLIAMS. Learning representations by back-propagating errors. *Nature*, **323**(6088):533–536, 9 October 1986.
- [652] CARL DAVID TOLMÉ RUNGE. Über die Zerlegung empirisch gegebener periodischer Funktionen in Sinuswellen [On the decomposition of empirically given periodic functions in sine waves]. *Zeitschrift für Mathematik und Physik: Organ für Angewandte Mathematik*, **48**:443–456, 1903.
- [653] CARL DAVID TOLMÉ RUNGE. Über die Zerlegung einer empirischen Funktion in Sinuswellen [On the decomposition of empirically given periodic functions in sine waves]. *Zeitschrift für Mathematik und Physik: Organ für Angewandte Mathematik*, **52**:117–123, 1905.
- [654] JAN RUSINEK. *Almost Miniatures: 555 Studies With Eight Chessmen*, volume 3 of *Selected Endgame Studies*. University of Limburg, 1994.
- [655] MICHEL JACQUEMIN S. LENNART JOHNSON, CHING-TIEN HO AND ALAN RUTTENBERG. *Computing fast Fourier Transforms on Boolean cubes and related networks*, volume 826, pages 223–231. Society of Photo-Optical Instrumentation Engineers, 1987.
- [656] GARY W. SABOT. Paralation LISP reference manual. Technical Report TMC-145, Thinking Machines Corporation, Cambridge, MA, 5 May 1985.
- [657] GARY W. SABOT. *The Paralation Model: Architecture-Independent Parallel Programming*. The MIT Press Series in Artificial Intelligence. MIT Press, Cambridge, MA, 1988.

- [658] GARY W. SABOT. Optimizing CM Fortran compiler for Connection Machine computers. *Journal of Parallel and Distributed Computing*, **23**(2):224–338, November 1994.
- [659] BRUCE ELI SAGAN. *The Symmetric Group: Representations, Combinatorial Algorithms and Symmetric functions*. Wadsworth & Brooks/Cole Mathematics Series. Wadsworth & Brooks/Cole Advanced Books and software, Pacific Grove, CA, 1991.
- [660] ALESSANDRO SALVIO. *Trattato dell'inventione et art liberale del gioco di scacchi* [treatise of the invention and liberal art of the game of chess]. Napoli, 1634.
- [661] ARTHUR L. SAMUEL. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, **2**(3):211–229, July 1959.
- [662] David Sankoff and Joseph B. Kruskal, editors. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA, 1983.
- [663] JONATHAN SCHAEFFER. Distributed game-tree searching. *Journal of Parallel and Distributed Computing*, **6**(1):90–114, February 1989.
- [664] JONATHAN SCHAEFFER, JOSEPH CULBERSON, NORMAN TRELOAR, BRENT KNIGHT, PAUL LU, AND DUANE SZAFRON. A world championship caliber checkers program. *Artificial Intelligence*, **53**(2–3):273–289, February 1992.
- [665] JONATHAN SCHAEFFER, JOSEPH CULBERSON, NORMAN TRELOAR, BRENT KNIGHT, PAUL LU, AND DUANE SZAFRON. Reviving the game of checkers. Technical Report TR 90–31, University of Alberta, September 1990.
- [666] STEPHEN T. SCHIBELL AND RICHARD M. STAFFORD. Processor interconnection networks from Cayley graphs. *Discrete Applied Mathematics*, **40**(3):333–357, 14 December 1992.
- [667] GUNTHER SCHMIDT AND THOMAS STRÖHLEIN. *Relationen und Graphen* [Relations and graphs]. Mathematik für Ingenieure. Springer-Verlag, Berlin, 1989.
- [668] K. E. SCHMIDT AND M. A. LEE. Implementing the fast multipole multipole method in three dimensions. *Journal of Stat. Physics*, **63**(5–6):1223–35, June 1991.
- [669] MARTIN SCHÖNERT ET AL. GAP: Groups, algorithms and programming, November 1993. Computer software.
- [670] ARNOLD SCHÖNHAGE AND VOLKER STRASSEN. Schnelle Multiplikation großer Zahlen [Fast multiplication of large numbers]. *Computing: Archiv für Elektronisches Rechnen*, **7**(3–4):281–292, 1971.
- [671] RICHARD SCHROEPEL AND ADI SHAMIR. A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems. *SIAM Journal on Computing*, **10**(3):456–464, August 1981.
- [672] JACOB T. SCHWARTZ. Ultracomputers. *ACM Transactions on Programming Languages and Systems*, **2**(4):484–521, October 1980.
- [673] K. H. SCHWARZ. Versuch eines mathematischen Schachprinzips [Essay on mathematical chess principles]. *Deutsche Schachzeitung*, **53**(11):321–324, November 1925.
- [674] WALTER SCHWARZ. Acorn Run-Time system for the CM-2. In Lenore M. Restifo Mullin et. al., editor, *Arrays, functional languages and parallel systems*, pages 35–57. Kluwer Academic Publishers, Boston, MA, 1991.

- [675] FRANCESCO SCIORTINO AND S.L. FORNILI. Hydrogen bond cooperativity in simulated water: time dependence analysis of pair interactions. *Journal Chemical Physics*, **90**(5):2786–2792, 1 March 1989.
- [676] FRANCESCO SCIORTINO, PETER H. POOLE, H. EUGENE STANLEY, AND SHLOMO HAVLIN. Lifetime of the bond network and gel-like anomalies in supercooled water. *Physical Review Letters*, **64**(14):1686–1691, 2 April 1990.
- [677] R. SEIDEL. What constitutes optimal play? *International Computer Chess Association Journal*, **9**(1):37–43, March 1986.
- [678] JEAN PIERRE SERRE. *Linear Representations of Finite Groups*, volume 42 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1977.
- [679] CLAUDE ELWOOD SHANNON. Programming a digital computer for playing chess. *Philosophical Magazine, Seventh Series*, **41**(314):256–275, March 1950.
- [680] SEBASTIAN SHAUMYAN. Genotype—a pure functional array language. In Lenore M. Res-tifo Mullin et. al., editor, *Arrays, functional languages and parallel systems*, pages 201–236. Kluwer Academic Publishers, Boston, MA, 1991.
- [681] ROLAND SILVER. The group of automorphisms of the game of 3-dimensional Ticktacktoe. *American Mathematical Monthly*, **74**(3):247–254, March 1967.
- [682] B.D. SILVERMAN AND R. LINSKER. A measure of DNA periodicity. *Journal of Theoretical Biology*, **118**(3):295–300, 7 February 1986.
- [683] GEOFF LESLIE SIMONS. *Is Man a Robot?* John Wiley & Sons, Chichester, England, 1986.
- [684] CHARLES COFFIN SIMS. Computations with permutation groups. In Stanley Roy Petrick, editor, *Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation*, pages 23–28, Los Angeles, 23–25 March 1971. ACM Press, New York 1971.
- [685] CHARLES COFFIN SIMS. *Computation With Finitely Presented Groups*, volume 48 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, England, 1994.
- [686] ALEXANDER SINGER. Implementations of artificial neural networks on the connection machine. *Parallel Computing*, **14**(3):305–315, August 1990.
- [687] JASWINDER PAL SINGH, CHRIS HOLT, TAKASHI TOTSUKA, ANOOP GUPTA, AND JOHN L. HENNESSY. Load balancing and data locality in hierarchical N-body methods. Technical Report CSL-TR-92-505, Stanford University, Stanford, CA, 1992.
- [688] P. SINGH AND GURPRIT KAUR. On quasi-rings. *Bulletin of the Calcutta Mathematical Society*, **76**(6):325–336, 1984.
- [689] RICHARD C. SINGLETON. On computing the fast Fourier transform. *Communications of the ACM*, **10**(10):647–654, October 1967.
- [690] STEVEN SOL SKIENA. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. The Advanced Book Program. Addison-Wesley, Redwood City, CA, 1990. Contains programs by Steven Skiena and Anil Bhansali.
- [691] DAVID BENSON SKILLICORN. Architecture-independent parallel computation. *Computer*, **23**(12):38–50, December 1990.

- [692] DAVID BENSON SKILLICORN. Models for practical parallel computation. *International Journal of Parallel Programming*, **20**(2):133–158, April 1991.
- [693] DAVID BENSON SKILLICORN. Deriving parallel programs from specifications using cost information. *Science of Computer Programming*, **20**(3):205–221, June 1993.
- [694] DAVID BENSON SKILLICORN. Structuring data parallelism using categorical data types. In *Proceedings of the 1993 Workshop on Programming Models for Massively Parallel Computers*, pages 110–115, Berlin, Germany, 20–23 September 1993. IEEE Computer Society Press, Los Alamitos, CA, 1993.
- [695] DAVID BENSON SKILLICORN. The categorical data type approach to general-purpose parallel computation. In Bjorn Perhson and Imre Simon, editors, *Proceedings of the IFIP 13th World Computer Congress*, volume A-51 of *IFIP Transactions: Computer Science and Technology*, pages 565–570, Hamburg, Germany, 28 August–2 September 1994. North-Holland, Amsterdam/London, 1994.
- [696] DAVID BENSON SKILLICORN AND WENTONG CAI. A cost calculus for parallel and functional programming. Technical Report ISSN-0836-0227-92-329, Department of Computing and Information Science, Queen’s University, Kingston, Ontario, K7L 3N6, May 1992.
- [697] DANIEL L. SLOTNICK, W. CARL BORCK, AND ROBERT C. MCREYNOLDS. The SOLOMON computer. In *Proceedings of the AFIPS Spring Joint Computer Conference*, volume 21, pages 97–107, San Francisco, CA, 1–3 May 1962. American Federation of Information Processing Societies/National Press, Palo Alto, CA.
- [698] DAVID SMITLEY AND KENT IOBST. Bit-serial SIMD on the CM-2 and the Cray-2. *Journal of Parallel and Distributed Computing*, **11**(2):135–145, February 1991.
- [699] A. J. SOBEY. Pawnlessness. *EG*, **1**?(12):335–341, March 1968.
- [700] LEONARD H. SOICHER. GRAPE: A system for computing with graphs and groups. In Larry Finkelstein and William Kantor, editors, *Groups and Computation: Workshop on Groups and Computation*, volume 11 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 287–292, Rutgers, NJ, 7–10 October 1991. American Mathematical Society, Providence, RI, 1993.
- [701] EDWIN HENRY SPANIER. *Algebraic Topology*. Springer-Verlag, New York, 1966.
- [702] PHILIP STAMMA. *Essai sur le jeu des échecs* [essay on the game of chess]. Paris, 1737.
- [703] CRAIG STANFILL. Communications architecture in the Connection Machine system. Technical Report HA87-3, Thinking Machines Corporation, Cambridge, MA, 1987.
- [704] WILLI-HANS STEEB. *Kronecker Product of Matrices and Applications*. Bibliographisches Institut, Mannheim/Vienna, 1991.
- [705] GUY LEWIS STEELE, JR. AND WILLIAM DANIEL HILLIS. Connection Machine Lisp: fine-grained parallel symbolic processing. Technical Report TMC-150, Thinking Machines Corporation, Cambridge, MA, 1986.
- [706] IGOR STEINBERG AND MARVIN SOLOMON. Searching game trees in parallel. In Pen-Chung Yew, editor, *Proceedings of the 1990 International Conference on Parallel Processing*, volume 3, pages 9–17, Pennsylvania State University, 13–17 August 1990. Pennsylvania State University Press, University Park, PA, 1990.

- [707] LEWIS BENJAMIN STILLER. Parallel analysis of certain endgames. *International Computer Chess Association Journal*, **12**(2):55–64, June 1989.
- [708] LEWIS BENJAMIN STILLER. Group graphs and computational symmetry on massively parallel architecture. *Journal of Supercomputing*, **5**(2/3):99–117, November 1991.
- [709] LEWIS BENJAMIN STILLER. Karpov and Kasparov: the end is perfection. *International Computer Chess Association Journal*, **14**(4):198–201, December 1991.
- [710] LEWIS BENJAMIN STILLER. Some results from a massively parallel retrograde analysis. *International Computer Chess Association Journal*, **14**(3):129–134, September 1991.
- [711] LEWIS BENJAMIN STILLER. An algebraic foundation for Fortran 90 communication intrinsics. Technical Report LA-UR-92-5211, Los Alamos National Laboratory, Los Alamos, NM 87545, August 1992.
- [712] LEWIS BENJAMIN STILLER. An algebraic paradigm for the design of efficient parallel programs. Technical Report JHU-92/26, Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218, November 1992.
- [713] LEWIS BENJAMIN STILLER. Endgame source code goes public. *International Computer Chess Association Journal*, **15**(2):107, June 1992.
- [714] LEWIS BENJAMIN STILLER. How to write fast and clear parallel programs using algebra. Technical Report LA-UR-92-2924, Los Alamos National Laboratories, Los Alamos, NM, 1992.
- [715] LEWIS BENJAMIN STILLER. White to play and win in 223 moves: a massively parallel exhaustive state space search. *Journal of the Advanced Computing Laboratory*, **1**(4):1, 14–19, July 1992.
- [716] LEWIS BENJAMIN STILLER. Multilinear algebra and chess endgames. In Richard Nowakowski, editor, *Proceedings of the 1994 Workshop on Combinatorial Games*, Berkeley, CA, 11–22 July 1994. To appear.
- [717] LEWIS BENJAMIN STILLER, LUKE L. DAEMEN, AND JAMES E. GUBERNATIS. n -body simulations on massively parallel architectures. *Journal of Computational Physics*, **115**(2):550–552, December 1994.
- [718] LEWIS BENJAMIN STILLER AND ANGEL E. GARCÍA. Parallelization of a convolution arising in the computation of the residence time of water in the hydration shells of biomolecules. Technical Report LA-UR-29-3717, Los Alamos National Laboratory, Los Alamos, NM 87545, September 1992.
- [719] THOMAS GREENWAY STOCKHAM, JR. High-speed convolution and correlation. In *Proceedings of the AFIPS Spring Joint Computer Conference*, volume 28. Spartan Books, Washington, D.C., 1966, 1966.
- [720] LARRY JOSEPH STOCKMEYER AND ASHOK KUMAR CHANDRA. Provably difficult combinatorial games. *SIAM Journal on Computing*, **8**(2):151–174, May 1979.
- [721] HAROLD STUART STONE. Parallel processing with the perfect shuffle. *IEEE Transactions on Computers*, **C-20**:153–161, 1971.
- [722] HAROLD STUART STONE. *High-Performance Computer Architecture*. Addison-Wesley, Reading, MA, 1993.

- [723] HAROLD STUART STONE AND JANICE M. STONE. Efficient search techniques—an empirical study of the n -queens problem. *IBM Journal of Research and Development*, **31**(4):464–474, July 1987.
- [724] JAMES ANDREW STORER. A note on the complexity of chess. In *Proceedings of the 1979 Conference on Information Sciences and Systems*, pages 160–166, Baltimore, MD, 28–30 March 1979. Department of Electrical Engineering, Johns Hopkins University, 1979.
- [725] VOLKER STRASSEN. Gaussian elimination is not optimal. *Numerische Mathematik*, **13**(4):354–356, 14 August 1969.
- [726] VOLKER STRASSEN. The asymptotic spectrum of tensors and the exponent of matrix multiplication. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 49–54, Toronto, Ontario, 27–29 October 1986. IEEE Computer Society Press, Washington, DC / Los Angeles, CA, 1986.
- [727] THOMAS STRÖHLEIN. *Untersuchungen über Kombinatorische Spiele* [Investigations of combinatorial games]. PhD thesis, Fakultät für Allgemeine Wissenschaften der Technischen Hochschule München, February 1970.
- [728] THOMAS STRÖHLEIN AND L. ZAGLER. Analyzing games by boolean matrix iteration. *Discrete Mathematics*, **19**(2):183–193, August 1977.
- [729] P. STRUIK. A systematic design of a parallel program for Dirichlet convolution. *Science of Computer Programming*, **15**(2–3):185–200, December 1990.
- [730] DAN SUCIU AND VAL TANNEN. Efficient compilation of high-level data-parallel algorithms. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 57–66, Cape May, New Jersey, June 27–29 1994. ACM Press, New York, 1994.
- [731] [SCIENTIFIC AMERICAN SUPPLEMENT]. Torres and his remarkable automatic devices. *Scientific American Supplement*, **53**(2079):296–298, November 1915.
- [732] PAUL N. SWARZTRAUBER. FFT algorithms for vector computers. *Parallel Computing*, **1**:45–63, 1984.
- [733] PAUL N. SWARZTRAUBER. Multiprocessor FFTs. *Parallel Computing*, **5**(1–2):197–210, July 1987.
- [734] BOLESŁAW K. SZYMANSKI, JAMES HICKS, R. JAGANNATHAN, VIVEK SARKAR, DAVID BENSON SKILLICORN, AND ROBERT K. YATES. Is there a future for functional languages in parallel programming? In *Proceedings of the 1994 International Conference on Computer Languages*, pages 299–304, Toulouse, France, 16–19 May 1994. IEEE Computer Society Press, Los Alamitos, CA, 1994.
- [735] SHOUWEN TANG, KAIZHONG ZHANG, AND XIAOLIN WU. Matching with matrix norm minimization. In Maxime Crochemore and Dan Gusfield, editors, *Combinatorial Pattern Matching: 5th Annual Symposium. Proceedings*, volume 807 of *Lecture Notes in Computer Science*, pages 250–258, Asilomar, CA, 5–8 June 1994. Springer-Verlag, Berlin, 1994.
- [736] ROBERT ENDRE TARJAN. Fast algorithms for solving path problems. *Journal of the Association for Computing Machinery*, **28**(3):594–614, July 1981.
- [737] ROBERT ENDRE TARJAN. A unified approach to path problems. *Journal of the Association for Computing Machinery*, **28**(3):577–593, July 1981.

- [738] JOHANNES TAUSCH. A generalization of the discrete Fourier transformation. In Eugene Leo Allgower, Klaus Böhmer, and Martin Golubitsky, editors, *Bifurcation and Symmetry: Cross Influence Between Mathematics and Applications*, volume 104 of *International Series of Numerical Mathematics*, pages 405–412. Birkhäuser Verlag, Basel, Germany, 1992.
- [739] WILLIE TAYLOR. New paths from dead ends. *Nature*, **356**(6369):478–479, 9 April 1992.
- [740] GERALD TESAURO AND TERRENCE JOSEPH SEJNOWSKI. A parallel network that learns to play backgammon: recent results. In *Proceedings: 1988 Spring Symposium Series: Computer Game Playing*, pages 41–45, Stanford University, Stanford, CA, 22–24 March 1988. American Association for Artificial Intelligence.
- [741] LARRY H. THIEL, CLEMENT WING HONG LAM, AND S. SWIERCZ. Using a CRAY-1 to perform backtrack search. In *Proceedings of the Second International Conference on Supercomputing, Supercomputing '87*, volume 3, pages 92–99, San Francisco, CA, May 1987. International Supercomputing Institute, St. Petersburg, FL, 1987.
- [742] Thinking Machines Corporation, Cambridge, MA. *Connection Machine Model CM-2 Technical Summary*, November 1990.
- [743] Thinking Machines Corporation, Cambridge, MA. **Lisp Dictionary*, version 5.2 edition, February 1990.
- [744] Thinking Machines Corporation, Cambridge, MA. *Connection Machine CM-200 Series Technical Summary*, June 1991.
- [745] Thinking Machines Corporation, Cambridge, MA. *Connection Machine CM-5 Technical Summary*, January 1992.
- [746] Thinking Machines Corporation, Cambridge, MA. *CM-5 CM Fortran Performance Guide*, version 2.2 edition, October 1994.
- [747] Thinking Machines Corporation, Cambridge, MA. *CM Fortran Language Reference Manual*, version 2.2 edition, October 1994.
- [748] Thinking Machines Corporation, Cambridge, MA. *CMSSL for CM Fortran*, version 3.2 edition, April 1994.
- [749] L.H. THOMAS. Using computers to solve problems in physics. In Walter F. Freiberger and William Prager, editors, *Applications of Digital Computers*, pages 42–57. Ginn and Company, Boston, 1963.
- [750] KENNETH LANE THOMPSON. Regular expression search algorithm. *Communications of the ACM*, **11**(6):419–422, June 1968.
- [751] [KENNETH LANE THOMPSON]. Chess games. *EG*, **5**(74), November 1983.
- [752] KENNETH LANE THOMPSON. *C* the programs that generate endgame data bases. *EG*, **6**(83):2, May 1986. Apparently a letter.
- [753] KENNETH LANE THOMPSON. Retrograde analysis of certain endgames. *International Computer Chess Association Journal*, **9**(3):131–139, September 1986.
- [754] KENNETH LANE THOMPSON. personal communication, April 1990.
- [755] TOMMASO TOFFOLI. Cellular automata mechanics. Technical Report 208, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, MI, 1977.

- [756] RICHARD TOLIMIERI. Multiplicative characters and the discrete Fourier transform. *Advances in Applied Mathematics*, **7**(3):344–380, 1986.
- [757] RICHARD TOLIMIERI AND MYOUNG AN. *Computations in X-ray crystallography*, volume 315 of *NATO ASI Series C: Mathematical and Physical Sciences*, pages 237–250. Kluwer Academic Publishers, Dordrecht/Boston, 1990.
- [758] RICHARD TOLIMIERI, MYOUNG AN, AND CHAO LU. *Algorithms for Discrete Fourier Transform and Convolution*. Springer-Verlag, New York, 1989.
- [759] RICHARD TOLIMIERI, MYOUNG AN, AND CHAO LU. *Mathematics of Multidimensional Fourier Transform Algorithms*. Springer-Verlag, New York, 1993.
- [760] CHARLES TONG AND PAUL N. SWARZTRAUBER. Ordered fast Fourier transforms on a massively parallel computer. *Journal of Parallel and Distributed Computing*, **12**(1):50–59, May 1991.
- [761] GONZALO TORRES-QUEVEDO. Présentation des appareils de Leonardo Torres-Quevedo [Presentation of the apparatus of Leonardo Torres-Quevedo]. In *Les machines à calculer et la pensée humaine*, volume 37 of *Colloques internationaux du Centre National de la Recherche Scientifique*, pages 383–406. Service des Publications du Centre National de la Recherche Scientifique, Paris, 1953, 9–13 January 1951.
- [762] E.A. TRACHTENBERG AND MARK GIRSHEVICH KARPOVSKY. Filtering in a communication by Fourier transforms over finite groups. In Mark Girshevich Karpovsky, editor, *Spectral Techniques and Fault Detection*, volume 11 of *Notes and Reports in Computer Science and Applied Mathematics*, pages 179–216. Academic Press, Orlando, FL, 1985.
- [763] BORIS AVRAAMOVICH TRAKHTENBROT. A survey of Russian approaches to *perebor* (brute-force search) algorithms. *Annals of the History of Computing*, **6**(4):384–399, October 1984.
- [764] Алексей Алексеевич Троицкий [ALEKSEI ALEKSEEVICH TROITZKY]. König und zwei Springer gegen König und Bauer [King and two knights against king and pawn]. *Deutsche Schachzeitung*, **61**(5):129–131, May 1906. Part 1. Other parts: **61**(6) 161–166, June 1906; **61**(7) 193–197, July 1906; **61**(9) 257–260, September, 1906; **62**(1) 1–5, January, 1907; **62**(4) 97–100, February, 1907; **62**(6) 161–164, June, 1907; **62**(8) 225–228, August, 1907; **63**(1) 1–6, January, 1908; **63**(4) 101–104, April, 1908; **63**(7) 197–200, July, 1908; **63**(10) 293–294, October, 1908; **64**(2) 33–36, February, 1909; **64**(4) 97–99 April, 1909; **64**(6) 161–163 June, 1909; **64**(8) 225–227 August, 1909; **64**(11) 321–323 November, 1909; **65**(3) 65–67, March, 1910; **65**(5) 129–133, May, 1910.
- [765] Алексей Алексеевич Троицкий [ALEKSEI ALEKSEEVICH TROITZKY]. *500 Endspielstudien [500 endgame studies]*. Kagan, Berlin, 1924. Alternative spelling as Troitskii.
- [766] Алексей Алексеевич Троицкий [ALEKSEI ALEKSEEVICH TROITZKY]. Два коня против пешек (теоретический очерк). In Сборник Шахматных Этюдов, с Приложением Краткой Теории Эндшпиля: Два Коня Против Пешек [Collection of chess studies, with a theoretical supplement on the the endgame of two knights against pawn], pages 248–288. ОГИЗ, Физкультура и туризм, Москва, 1934.
- [767] PINGH-SHENG TSENG. *A parallelizing compiler for distributed memory parallel computers*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, May 1989.

- [768] JOHN TUREK, JOEL L. WOLF, AND PHILIP S. YU. Approximate algorithms for scheduling parallelizable tasks. In *4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 323–332, San Diego, CA, 29 June–1 July 1992. ACM Press, New York, 1992.
- [769] EDWARD C. TURNER AND KAREN F. GOLD. Rubik’s groups. *American Mathematical Monthly*, **92**(9):617–629, November 1985.
- [770] JEFFREY DAVID ULLMAN. *Computational Aspects of VLSI*. Principles of Computer Science Series. Computer Science Press, 1984.
- [771] ROBERT JOSEPH VALENZA. A representation-theoretic approach to the DFT with noncommutative generalizations. *IEEE Transactions on Signal Processing*, **40**(4):814–822, April 1992.
- [772] LESLIE G. VALIANT. General context-free recognition in less than cubic time. *Journal of Computer and System Sciences*, **10**(2):308–315, April 1975.
- [773] LESLIE G. VALIANT. Optimally universal parallel computers. *Philosophical Transactions of the Royal Society of London Series A*, **326**:373–376, 1988.
- [774] LESLIE G. VALIANT. A bridging model for parallel computation. *Communications of the ACM*, **33**(8):103–111, August 1990.
- [775] LESLIE G. VALIANT. General purpose parallel architectures. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science: Algorithms and Complexity*, volume A, chapter 18, pages 945–971. Elsevier Science Publishers, Amsterdam, 1990.
- [776] ERIC F. VAN DE VELDE. *Concurrent Scientific Computing*, volume 16 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 1994.
- [777] H. JAAP VAN DEN HERIK, I. S. HERSCHBERG, AND NAJIB NAKAD. A six-men-endgame database: KRP(a2)KbBP(a3). *International Computer Chess Association Journal*, **10**(4):163–180, December 1987.
- [778] H. JAAP VAN DEN HERIK AND I. S. HERSCHBERG. The construction of an omniscient endgame database. *International Computer Chess Association Journal*, **8**(2):66–87, June 1985.
- [779] H. JAAP VAN DEN HERIK AND I. S. HERSCHBERG. Elementary theory improved, a conjecture refuted. *International Computer Chess Association Journal*, **8**(3):141–149, September 1985.
- [780] H. JAAP VAN DEN HERIK AND I.S. HERSCHBERG. A data base on data bases. *International Computer Chess Association Journal*, **9**(1):29–34, March 1986.
- [781] H. JAAP VAN DEN HERIK, I.S. HERSCHBERG, T.R. HENDRIKS, AND J.P. WIT. Computer checks on human analyses of the KRKB endgame. *International Computer Chess Association Journal*, **11**(1):26–31, March 1988.
- [782] DENIS F. VERHOEF AND JACCO H. WESSELIUS. Two-ply KRKN: Safely overtaking Quinlan. *International Computer Chess Association Journal*, **10**(4):181–190, December 1987.
- [783] RAKESH MOHAN VERMA. Strings, trees, and patterns. *Information Processing Letters*, **41**(3):157–161, 6 March 1992.
- [784] HENRI VIGNERON. Les automates [The automatons]. *La Nature*, pages 56–61, 1914.
- [785] UZI VISHKIN. Optimal parallel pattern matching in strings. *Information and Control*, **67**(1–3):91–113, October–December 1985.

- [786] UZI VISHKIN. A case for the PRAM as a standard programmer's model. In Friedhelm Meyer auf der Heide, Burkhard Monien, and Arnold Leonard Rosenberg, editors, *Parallel Architectures and Their Efficient Use: First Heinz Nixdorf Symposium Proceedings*, volume 678 of *Lecture Notes in Computer Science*, pages 11–19, Paderborn, Germany, 11–13 November 1992. Springer-Verlag, Berlin, 1993.
- [787] JOHN VON NEUMANN. First draft of a report on the EDVAC, 1945. Report prepared for U.S. Army Ordinance Department under Contract W-670-ORD-4926. Reprinted in *Papers of John von Neumann on Computers and Computer Theory*. William Aspray and Arthur Walter Burks, (Editors), Volume 12 of *The Charles Babbage Institute Reprint Series for the History of Computing*, The MIT Press, Cambridge, MA, 1987, pp. 17–82.
- [788] JOHN VON NEUMANN AND OSKAR MORGENSTERN. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, 1944.
- [789] JOACHIM VON ZUR GATHEN. Parallel linear algebra. In John H. Reif, editor, *Synthesis of Parallel Algorithms*, pages 573–618. Morgan Kaufmann Publishers, San Mateo, CA, 1993.
- [790] STEPHAN WAACK. The parallel complexity of some constructions in combinatorial group theory. In *Mathematical Foundations of Computer Science*, pages 492–498, Banská Bystrica, Czechoslovakia, 27–31 August 1990. Springer-Verlag, Berlin, 1990.
- [791] STEPHAN WAACK. On the parallel complexity of linear groups. *Informatique théorique et Applications*, **25**(4):323–354, 1991.
- [792] BARTEL LEENDERT WAERDEN. *A History of Algebra: From Al-Khwārizmī to Emmy Noether*. Springer-Verlag, Berlin/New York, 1985.
- [793] DAVID LEIGH WALTZ. Massively parallel AI. *International Journal of High Speed Computing*, **5**(3):491–501, September 1993.
- [794] MICHAEL S. WARREN AND JOHN K. SALMON. Astrophysical N -body simulations using hierarchical tree data structures. In *Proceedings. Supercomputing '92*, pages 570–576, Minneapolis, MN, 16–20 November 1992. IEEE Computer Society Press, Los Alamitos, CA, 1992.
- [795] MICHAEL S. WARREN AND JOHN K. SALMON. A fast tree code for many-body problems. *Los Alamos Science*, **22**:88–97, 1994.
- [796] WOLFGANG WECHLER. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin/New York, 1992.
- [797] PETER WEINER. Linear pattern matching algorithm. In *Proceedings of the 14th IEEE Symposium on Switching and Automata Theory*, pages 1–11, University of Iowa, 15–17 October 1973. IEEE Computer Society, Publications Office. Northridge, California.
- [798] JÜRGEN WEISS. An $n^{3/2}$ lower bound on the monotone network complexity of the Boolean convolution. *Information and Control*, **59**(1–3):184–188, October–December 1983.
- [799] BURTON WENDROFF, TONY WARNOCK, LEWIS BENJAMIN STILLER, DEAN MAYER, AND RALPH BRICKNER. Bits and pieces: constructing chess endgame databases on parallel and vector architectures. *Applied Numerical Mathematics*, **12**(1–3):285–95, May 1993.
- [800] ARTHUR THOMAS WHITE. *Graphs, Groups, and Surfaces*. North-Holland/Elsevier, New York, 1984.
- [801] DENNIS EDWARD WHITE. Multilinear enumerative techniques. *Linear and Multilinear Algebra*, **2**:341–352, 1975.

- [802] HELMUT WIELANDT. *Finite Permutation Groups*. Academic Press, New York, 1964.
- [803] JACK WISDOM. The origin of the Kirkwood gaps: A mapping for asteroidal motion near the 3/1 commensurability. *Astronomical Journal*, **87**(3):577, March 1982.
- [804] JACK WISDOM. Chaotic behavior and the origin of 3/1 Kirkwood gap. *Icarus*, **56**(1):51–74, October 1983.
- [805] MICHAEL E. WOLF AND MONICA S. LAM. An algorithmic approach to compound loop transformations. In Alexandru Nicolau, David Gelernter, Thomas Gross, and David Padua, editors, *Advances in Languages and Compilers for Parallel Processing*, Research Monographs in Parallel and Distributed Computing, pages 243–259. Pitman, London, 1991.
- [806] MICHAEL JOSEPH WOLFE. *Optimizing Supercompilers for Supercomputers*. Research Monographs in Parallel and Distributed Computing. MIT Press; Pitman, Cambridge, MA/London, England, 1989.
- [807] AHNONT WONGSEELASHOTE. Semirings and path spaces. *Discrete Mathematics*, **26**:55–78, 1979.
- [808] I-CHEN WU AND HSING-TSUNG KUNG. Communication complexity for parallel divide-and-conquer. In *Proceedings: 32nd Annual Symposium on Foundations of Computer Science*, pages 151–162, San Juan, Puerto Rico, 1–4 October 1991. IEEE Computer Society Press, Los Alamitos, CA, 1991.
- [809] HANS WUSSING. *The Genesis of the Abstract Group Concept: A Contribution to the History of the Origin of Abstract Group Theory*. The MIT Press, Cambridge, MA, 1984. Translation by Abe Shenitzer of Hans Wussing’s *Die Genesis des abstrakten Gruppenbegriffes: ein Beitrag zur Entstehungsgeschichte der abstrakten Gruppentheorie*, Deutscher Verlag der Wissenschaften, Berlin, 1969.
- [810] J. ALLAN YANG AND YOUNG IL CHOO. Formal derivation of an efficient parallel Gauss-Seidel method on a mesh of processors. Technical Report TR-870, Yale University Department of Computer Science, October 1991.
- [811] J. ALLAN YANG AND YOUNG IL CHOO. Formal derivation of an efficient parallel Gauss-Seidel method on a mesh of processors. Technical Report TR-870, Yale University Department of Computer Science, October 1991.
- [812] J. ALLAN YANG AND YOUNG IL CHOO. Formal derivation of an efficient parallel 2-D Gauss-Seidel method. In *Proceedings of the 6th International Parallel Processing Symposium*, pages 204–207, Beverly Hills, CA, 23–26 March 1992. IEEE Computer Society Press, Los Alamitos, CA, 1992.
- [813] J. ALLAN YANG AND YOUNG IL CHOO. Metalinguistic features for formal parallel-program transformation. In *4th IEEE International Conference on Computer Languages*, pages 65–75, San Francisco, 20–23 April 1992. IEEE Computer Society Press, Los Alamitos, 1992.
- [814] MICHAEL YOELI. A note on a generalization of boolean matrix theory. *American Mathematical Monthly*, **68**:552–557, 1961.
- [815] MICHAEL YOELI. Binary ring sequences. *American Mathematical Monthly*, **69**:852–855, 1962.

- [816] BRYANT WHITTIER YORK. Implications of parallel architectures for permutation group computation. In Larry Finkelstein and William Kantor, editors, *Groups and Computation: Workshop on Groups and Computation*, volume 11 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 293–313, Rutgers, NJ, 7–10 October 1991. American Mathematical Society, Providence, RI, 1993.
- [817] BRYANT WHITTIER YORK AND OTTORINO ORI. A fast parallel method for exhaustive C_{60} enumeration. In Richard Sincovec, editor, *Proceedings of the 6th SIAM Conference on Parallel Processing for Scientific Computing*, volume 1, pages 282–285, Norfolk, VA, March 22–24 1993.
- [818] G. YUVAL. An algorithm for finding all shortest paths using $N^{2.81}$ infinite-precision multiplications. *Information Processing Letters*, **4**(6):155–156, March 1976.
- [819] ERNST ZERMELO. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels [On an application of set theory to the theory of playing chess]. In Ernest William Hobson and Augustus Edward Hough Love, editors, *Proceedings of the Fifth International Congress of Mathematicians*, pages 501–504, Vol. 2, Cambridge, England, 22–28 August 1912. Cambridge University Press, Cambridge, England, 1913.
- [820] ERNST ZERMELO. An application of set-theory to the theory of chess-playing. *FIRBUSH News*, **6**:37–42, 1976. Translated from the German by M.D. Grant.
- [821] XIRU ZHANG, MICHAEL MCKENNA, JILL P. MESIROV, AND DAVID LEIGH WALTZ. The back-propagation algorithm on grid and hypercube architectures. *Parallel Computing*, **14**(3):317–327, August 1990.
- [822] XIRU ZHANG, MICHAEL MCKENNA, JILL P. MESIROV, AND DAVID LEIGH WALTZ. An efficient implementation of the back-propagation algorithm on the Connection Machine CM-2. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 801–809. Morgan Kaufmann, San Mateo, CA, 1990.
- [823] FENG ZHAO. An $O(N)$ algorithm for three-dimensional N -body simulations. Technical Report AI-995, MIT Artificial Intelligence Laboratory, Cambridge, MA, October 1987.
- [824] FENG ZHAO AND S. LENNART JOHNSON. The parallel multipole method on the Connection Machine. *SIAM Journal on Scientific and Statistical Computing*, **12**(6):1420–1437, November 1991.
- [825] O. A. ZHAROV AND L. S. KAZARIN. Towards a theory of group convolution. *Problems of Information Transmission*, **29**(3):292–294, July–September 1993.
- [826] SI-QING ZHENG. SIMD data communication algorithms for multiply twisted hypercube. In V.K. Prasanna Kumar, editor, *Proceedings: the Fifth International Parallel Processing Symposium*, pages 120–125, Anaheim, CA, 30 April–2 May 1991. IEEE Computer Society Press, Los Alamitos, CA, 1991.
- [827] HANS P. ZIMA, H.-J. BAST, AND HANS MICHAEL GERNDT. SUPERB: a tool for semi-automatic MIMD/SIMD parallelization. *Parallel Computing*, **6**(1):1–18, January 1988.
- [828] HANS P. ZIMA, PETER BREZANY, BARBARA CHAPMAN, PIYUSH MEHROTRA, AND ANDREAS SCHWALD. Vienna Fortran—A language specification version 1.1. Technical Report ACPC/TR 92-4, Austrian Center for Parallel Computation, [Vienna, Austria], March 1992.
- [829] HANS P. ZIMA AND BARBARA CHAPMAN. *Supercompilers for Parallel and Vector Computers*. ACM Press Frontier Series. ACM Press; Addison-Wesley, New York/Wokingham, England/Reading, MA, 1990.