Research Notes in Artificial Intelligence

STEPHANIE FORREST

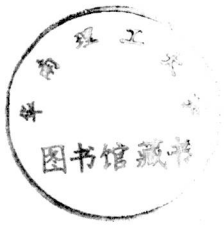
Parallelism and Programming in Classifier Systems

Pitman, London
Morgan Kaufmann Publishers, Inc., San Mateo, California

TP18
F729
9262560

Stephanie Forrest
Center for Nonlinear Studies and Computing Division
Los Alamos National Laboratory

Parallelism and Programming in Classifier Systems



Pitman, London

Morgan Kaufmann Publishers, Inc., San Mateo, California

PITMAN PUBLISHING
128 Long Acre, London WC2E 9AN

A Division of Longman Group UK Limited

© Stephanie Forrest 1991

First published 1991

Available in the Western Hemisphere from
MORGAN KAUFMAN PUBLISHERS, INC.
2929 Campus Drive, San Mateo, California 94403

ISSN 0268-7526

British Library Cataloguing in Publication Data

Forrest, Stephanie
Parallelism and programming in classifier systems. –
(Research notes in artificial intelligence; 0268-7526).
1. Computer systems. Parallel programming
I. Title II. Series
004.35

ISBN 0-273-08825-4

Library of Congress Cataloging-in-Publication Data

Forrest, Stephanie
Parallelism and programming in classifier systems/Stephanie Forrest.
p. cm. – (Research notes in artificial intelligence (London, England), ISSN 0268-7526)
Includes bibliographical references and index.
ISBN 1-55860-107-4
1. Parallel processing (Electronic computers) 2. Parallel programming (Computer science) I. Title II. Series.
QA76.58.F66 1990
005.2—dc20 90-41616
CIP

All rights reserved; no part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise without either the prior written permission of the Publishers or a licence permitting restricted copying issued by the Copyright Licensing Agency, 33-34 Alfred Place, London WC1E 7DP. This book may not be lent, resold, hired out or otherwise disposed of by way of trade in any form of binding or cover other than that in which it is published, without the prior consent of the Publishers.

Reproduced and printed by photolithography in Great Britain
by Biddles Ltd, Guildford and King's Lynn

Parallelism and Programming in Classifier Systems

Preface

This work originally appeared in 1985 as my Ph.D. thesis at the University of Michigan in Ann Arbor. In updating it for this volume, I have tried to place it in a broader context of research on parallelism, intelligent systems, and emergent computation. Many people have contributed constructive criticism, new insights, and moral support over the years, and it is a pleasure to acknowledge their contributions.

At the University of Michigan, my fellow students had a strong influence on the work. They helped me refine vague and imprecise ideas, discouraged me from pursuing dead-ends, and faithfully read and criticized every word of the original dissertation. Large sections of Chapter 4 in particular are the result of their efforts as well as mine. Specifically, I would like to thank Rik Belew, Lashon Booker, Laurette Bradley, Paul Grosso, Dan Kaiser, Chris Langton, Melanie Mitchell, and Rick Riolo for all of the hours they have spent helping me think about parallelism, classifier systems, and KL-ONE.

I am also grateful to John Holland and the Logic of Computers Group for convincing me to join the Computer and Communication Sciences Department, and for the intellectual and financial support that they provided through graduate school. The other members of my doctoral committee, Arthur Burks, Michael Cohen, and Paul Scott, each helped me in their own way and the work profitted tremendously from their suggestions. Although not an official committee member, Quentin Stout read the original thesis carefully and provided a number of important suggestions.

Tom Lipkis and Bill Mark, who were at USC/Information Sciences Institute during my graduate years, answered endless questions about KL-ONE and got me started on the problem of classification.

N.S. Sridharan first encouraged me to consider the *Research Notes in Artificial Intelligence* series, and I would like to thank him and the other editors at Pitman Publishing for their support and patience. In preparing the revisions, Ron Brachman, Michael Fehling, Stevan Harnad, and Robert MacGregor made helpful suggestions for updating the references. Erica Jen and Pat McGee read the revised manuscript in several versions. Ronda Villa-Butler and Della Ulibarri of the Santa Fe Institute typeset and copyedited the manuscript. Jane Self designed the original figures, which appear largely unchanged in the revised version. The anonymous reviewers read the manuscript very carefully and made many helpful suggestions. Finally, I would like to thank my husband Fred Carey for putting up with me and the thesis for all of these years.

This research was supported by National Science Foundation Grant DCR-8305830.

此为试读，需要完整PDF请访问：www.ertongbook.com

For my mother,
Who gave me the opportunities she never had.

Contents

List of Figures

List of Appendices

1 Introduction

A classifier system is a computational model of cognition based on the principles of learning, intermittent feedback from the environment, and construction of internal models [73]. Classifier systems are relevant to the study of intelligence both as a theory of cognition and as a parallel architecture for implementing artificial intelligence (AI) models efficiently. These two aspects of classifier systems are interrelated, and it is their interplay that is the subject of this book. The possibility that parallel architectures such as classifier systems could implement AI models directly has focused attention on models that might have both efficient hardware implementations and plausibility as intelligent systems. The interplay between cognitive modeling considerations and the requirement for reasonable performance is captured by the following questions: What kinds of architectures can most effectively implement computational models of intelligence? What computational systems are most appropriate for modeling intelligent behavior? How does intelligent behavior emerge from the interactions of many components?

A classifier system consists of three layers: a parallel rule-based message-passing system, the bucket-brigade learning algorithm, and the genetic algorithm. The message-passing system is the fundamental computational engine of the system. It consists of a database of condition/action rules, called classifiers, that read and write messages on a short-term message list. At the second level is the bucket-brigade learning algorithm which manages credit assignment among competing classifiers, distributing external reward to the rules that contribute to successful behavior. The bucket brigade plays a role similar to that of back propagation in neural networks. Finally, at the highest level are genetic operators that create new classifiers. A more detailed description of classifier systems appears in Chapter 2.

The message-passing part of a classifier system can be viewed as an abstract parallel machine, and the learning algorithms as mechanisms for adjusting the configuration of the machine over time. Under this view, each classifier is a separate processor, reading in messages as input and writing out others as output. The system exhibits parallelism at several levels, including: matching all of the bits in a message simultaneously, matching all conditions against one (or all) message(s) in the system simultaneously, and allowing more than one classifier to be active simultaneously. In the expected case of many classifiers and a small message list, massive parallelism is obtained through the parallel matching process and implicit parallelism arises when many classifiers share the same message.

The classifier system model can also be viewed as a parallel programming language in which correct programs are either pre-specified ("programmed") or learned dynamically. Any particular configuration of a classifier system is interpreted as a program, or set of instructions. There have been many attempts to apply machine learning algorithms to the problem of generating and debugging computer programs. These efforts have been

此为试读,需要完整PDF请访问: www.ertongbook.com

largely unsuccessful due to the brittle nature of conventional programming languages, in which a program's behavior can be changed dramatically by one misplaced character. In a classifier system, however, the restricted syntax of each instruction allows almost any combination of instructions to form a legal program. Additionally, the relative position of a single instruction does not determine its effect on the program. These two properties of classifier systems support the notion of a program as an "ecology" of individual instructions, each instruction filling some useful niche in the overall program and evolving in the context of the other instructions. This aspect of classifier systems emphasizes their potential for complex high-level behavior, called "emergent computation," in that local interactions among primitive elements can result in a global computation.

Unlike many AI systems, low-level mechanistic models of intelligence, such as classifier systems and connectionist networks, are based on analogies with biological, physical, or psychological phenomena. Their representations and learning algorithms operate at the "sub-symbolic" level in task environments where performance is measured in terms of input/output pairs. Classifier systems are one of the few alternative models to connectionism that have the same basic assumptions of massive parallelism, low-level domain-independent learning algorithms, and an emphasis on dynamic processes over static representations.

A common criticism of such systems is that they do not explicitly manipulate high-level symbolic structures such as those that are the basis of many current knowledge representation systems [18], programs that model human expertise [21], and systems that perform goal-directed planning [40]. By describing a classifier system implementation that represents and processes high-level symbolic knowledge structures, this book addresses the symbolic/subsymbolic issue. Here, the term "knowledge representation" includes both the data structures which store information and the algorithms that manipulate those structures.

The learning properties of classifier systems have been explored in much greater detail than the message-passing system. Holland described both the mechanics and underlying theory of genetic algorithms [67], and more recently Riolo investigated the bucket brigade's ability to maintain hierarchies and sequences of rules [101]. Several other researchers have recently explored the mathematics of bucket brigades [2, 51, 110]. This book is concerned with the computational properties of the underlying parallel machine, including computational completeness, programming and representation techniques, and efficiency of algorithms. In particular, efficient classifier system implementations of symbolic data structures and reasoning procedures are presented and analyzed in detail.

1.1 Parallelism and Classifier Systems

In Chapter 2, research on parallelism is described in terms of two categories: "coarse-grained" and "fine-grained." Coarse-grained architectures are composed of a small number of computationally complete heterogeneous processing units; they have a relatively low rate of communication between processors and often execute asynchronously. Fine-grained structures are composed of a large number of simple processors that also have a high rate of communication. Complex behavior can emerge from fine-grained systems as a higher-level phenomenon built up from the interactions among very simple units—an

example of emergent computation [47, 48]. In these systems, the patterns of activity produced by the lower-level processors are only meaningful when interpreted at the higher level. This is in contrast to coarse-grained organizations in which each processor works on a part of a larger problem, but each part has meaning independently of the remainder of the system. As an ecology of instructions, classifier systems illustrate many of the features of emergent computation [49].

Classifier systems provide an excellent example of fine-grained parallelism. The underlying action of the system is extremely simple, yet its global behavior can be highly complex. Each processor is limited computationally and there are many of them in a typical classifier system. Each unit (classifier) is locally controlled because its local properties (the condition and action parts of the classifier) determine which messages it will match and which messages it will produce.

The organization of information into fine-grained units is important for systems that learn. As Holland [67] and Lenat [83] have both pointed out, it is desirable to build up representations out of small units, or building blocks, so that simple learning rules can be applied to the parts rather than to their aggregates. Thus, it is reasonable to expect that the fundamental units of such systems will be "fine-grained." From the architectural point of view, fine-grained parallel models are promising implementation vehicles because they allow the large amounts of parallelism that are required to make experimental systems useful in real-time settings. Fine-grained organizations are appropriate for knowledge representation systems in which small amounts of information are added incrementally to existing systems. Additional processing power can be added as the system grows and high-level behavior, such as that exhibited by semantic networks, can be introduced through local interactions among small units.

Knowledge representation is an appropriate problem domain for studying the parallelism of classifier systems. Knowledge-based systems have traditionally been applied to static off-line domains where the facts of a case remain constant throughout the problem-solving process. For example, there are many knowledge-based systems that diagnose mechanical failures. These applications can be carried out with dedicated machines in environments where real-time response is not required. These systems also enjoy the advantage of a relatively stable knowledge base in which the system's knowledge about a given domain changes very slowly after the initial expertise has been "acquired." Once the description of a particular problem (e.g., symptoms of the mechanical failure) has been entered, there is little additional information to be absorbed in the course of its use. This approach is appropriate for problems that have a high payoff and can be solved "off-line," such as diagnosis, location of drilling sites, and configuring computers.

There is, however, increasing interest in applying knowledge-based systems technology in domains that require real-time interactions with ongoing processes [1, 27, 41, 56]. Examples include: natural language processing, autonomous vehicles and other robots, process control, aircraft control, medical applications (e.g., monitoring inhalation therapy), and intelligent computing environments. Current technology for building knowledge-based systems does not lend itself to continuous, real-time operation in dynamic environments. A system that is connected to its environment must produce output on a time scale that is appropriate for that environment. For example, it would not be appropriate to spend two hours processing an alarm condition that needs to be

handled within seconds. Reliable real-time behavior is difficult to achieve because the performance of AI systems can vary dramatically with different problem configurations.

For these systems to succeed, fast and frequent access to and on-line augmentation of large, dynamic knowledge bases are required. This implies a need for knowlege bases that have two properties: (1) predictable retrieval times and (2) the ability to add information dynamically. The KL-ONE family of languages addresses both of these issues.

1.2 Classification and KL-ONE

At the heart of any knowledge representation system is the problem of classification. In its most general formulation, classification relates incoming information to an existing knowledge base. In network-based systems, this is the problem of deciding which links to add between new and old nodes when incorporating new structures into the network. In expert systems, the classification problem arises when, for example, the system is asked to associate a set of symptoms with a particular disease [24]. In analogical reasoning systems [23] the problem of classification is implicit in deciding how to organize the growing data base of solved problems from which analogies can be drawn.

Classification is also central to retrieval operations in knowledge-based systems. The way in which information is organized determines how difficult various retrievals will be. For example, a representation system that organizes cities according to size rather than location will make it difficult to discover the name of all cities within a hundred mile radius of Spokane. Thus, any system that organizes or modifies information on an ongoing basis must address the issue of classification of new facts with respect to an existing information structure and with respect to the ways in which it may later be retrieved. In addition, access and storage patterns may change over time, creating a need for databases that can adaptively reorganize themselves.

Of the various knowledge representation paradigms in use today, one family of semantic network formalisms has focused directly on the problem of classification. This is the KL-ONE family [14], including KL-ONETalk [40], Krypton [16], KL-TWO [120], NIKL [103, 77], KANDOR [93], BACK [90], LOOM [86, 87] and CLASSIC [13, 20]. In all of these systems there is a well-defined notion of classification that allows the system to incorporate new concepts into existing network structures automatically. This is important because manual classification is only feasible in small systems that are relatively static, where the efficiencies of automation are not required. For large systems that change over time, some sort of automatic classification procedure is required. The classification issues in KL-ONE have been studied extensively and are reasonably well understood. This feature makes the KL-ONE family a particularly good place to begin considering the advantages or disadvantages of parallelism for knowledge representation systems.

In the KL-ONE formalisms, a distinction is drawn between definitional and assertional knowledge. This divides each system into two components: the definitional part, where descriptions are stored, and the assertional part, where extensions (the actual objects being described) and facts about those extensions are represented. The definitional part of the system is represented as a structured semantic network. Thus, a collection of definitions can be thought of as a graph, and individual descriptions within the network

can be regarded as subgraphs of the larger structure. In the assertional part of the system, propositions about the world (or some possible world) are represented as sentences in a formal logic.

An instance of the general classification problem arises in KL-ONE systems when a new description is added to the network. A new description must be attached to the existing network someplace and classification is the process of deciding where that place is [85]. The new description is a subgraph built out of other subgraphs whose relationship to the existing network is already known. The classification procedure is reducible to the problem of deciding subsumption relations between concepts (see Section 2.3will refer to the technical decision question in KL-ONE, while in other contexts "classification" will refer to the more general form of the problem described above.

Subsumption in KL-ONE is provably time-consuming. Brachman and Levesque showed that complete subsumption is NP-Complete even for restricted languages of the KL-ONE family [17]. More recently, Patel-Schneider showed that it is undecidable [94]. As a result, existing subsumption algorithms are designed to be sound but not complete. These results suggest that the more general problem of classification is inherently difficult. However, formal complexity analyses are rarely available for corresponding operations in most other systems (in other representation languages, classification is usually not isolated as a separate well-defined operation). If the subsumption problem can be made tractable through the use of parallelism, and if simple retrieval operations can be made more efficient, then parallelism will be a demonstrably useful technique for increasing the efficiency of current knowledge representation formalisms.

1.3 Subsymbolic Models of Intelligence

Classifier systems and connectionist models are both subsymbolic models of intelligence in the sense that their primitive components have meaning only in the context of the rest of the system. That is, an individual synapse in a connectionist model or a single classifier in a classifier system is uninterpretable without the context of the rest of the system. Intelligent behavior arises in subsymbolic systems through complex interactions among many low-level components and is a collective property of the units. The global behavior of the system is therefore intimately connected with functioning of the underlying components. In contrast, the Physcial Symbol System Hypothesis asserts that the essence of intelligence lies in the logic of symbol processing systems and that the details of how such systems are implemented are not important [92].

There has been a great deal of debate over which of these two approaches is preferable for modeling intelligent behavior (see for example [95, 114]). There is a growing consensus, however, that each view makes important contributions and that a full account of intelligent processes must take both into account [7]. The work reported here contributes to the growing body of research that studies how the two approaches fit together. Generally, these studies are existence proofs that show how connectionist architectures can implement a given symbolic procedure. Examples include Shastri's work on evidential reasoning in semantic networks [109], Touretzky and Hinton's connectionist implementation of a production system interpreter [115], and other connectionist implementations of AI systems [32, 116]. This work focuses on how symbolic structures can be

implemented efficiently in a subsymbolic massively parallel system. The criterion of efficiency is important since virtually all intelligent systems operate in resource-constrained environments of some form and cannot afford computationally unreasonable solutions.

1.4 Overview

This book focuses on the following specific questions:

(1) What are the basic computational properties of classifier systems?
(2) How does one represent symbolic structures with classifier systems?
(3) Can the parallelism of the classifier system be exploited to implement symbolic reasoning efficiently?
(4) Which operations are efficient and natural in classifier systems and which are not?

In the following chapters, I show how classifier systems can be used to implement a set of useful operations for the classification of knowledge in semantic networks. A subset of the KL-ONE language was chosen to demonstrate these operations. Specifically, the system performs the following tasks: (1) given the KL-ONE description of a particular semantic network, the system produces a set of production rules (classifiers) that represent the network, and (2) given the description of a new term, the system determines the proper location of the new term in the existing network. Chapter 5 describes these two parts of the system in detail.

The implementation reveals certain computational properties of classifier systems, including completeness, operations that are particularly natural and efficient, and those that are quite awkward. The work shows how high-level symbolic structures can be built up from classifier systems, and it demonstrates that the parallelism of classifier systems can be exploited to implement them efficiently. This is significant since classifier systems must construct large sophisticated models and reason about them if they are to be truly "intelligent."

The remaining chapters are organized as follows: (2) Background (3) Approach, (4) Classifier Systems, (5) Classifier System Implementation of KL-ONE, (6) Analysis of Results, and (7) Conclusions. The chapter on Background reviews previous results and develops a framework for organizing various classes of parallelism. In addition, it provides a general introduction to classifier systems and to the subset of KL-ONE that was implemented. Chapter 3, Approach, describes how the implementation is organized and the tools and methods that were used. Chapter 4 presents theoretical results and practical algorithms for classifier systems that are independent of the KL-ONE part of the project. Chapter 5 presents the implementation of KL-ONE using classifier systems. Chapter 6 analyses the material of Chapters 4 and 5, and the final chapter contains the conclusions.

The KL-ONE algorithms were implemented on top of the generic operations developed in Chapter 4 (Boolean operations, simple numerical processing, transitive closure, and synchronization). As a result, the parallel algorithms for KL-ONE could be implemented on any machine that supports efficient computation of these intermediate operations. These intermediate operations are similar to those that Fahlman [38] has argued are necessary for "intelligent systems." The ability to perform these computations

efficiently in classifier systems demonstrates the appropriateness of classifier systems as an architecture for artificial intelligence.

Parallel organizations are of interest to many areas of computer science, such as hardware specification, programming language design, configuration of networks of separate machines, and artificial intelligence. These applications are widely varied and raise many issues related to their own domains. Yet, they also share many of the same organizational concerns, such as communication, modularity, synchronization, control, and the overhead of dividing up a task into its parallel components. This book concentrates on a particular type of parallel organization and a particular problem in the area of AI, but the principles that are elucidated are applicable in the wider setting of computer science.

Since any well-defined algorithm can in principle be implemented as a serial computation on a Turing machine (Turing's Thesis), the contribution of parallel architectures and languages is not one of absolute computational power. Rather, the potential advantages lie in the areas of efficiency, ease of representation, and flexibility. It is therefore necessary to discover what the advantages or disadvantages are with respect to other architectures. One weakness with many research projects in this area is that they only show the possibility for solving certain classes of problems on particular architectures. Rigorous complexity analysis and consideration of fundamental computational tradeoffs are rarely provided. These tradeoffs include: space versus time, global versus local communication, centralized control versus local autonomy, and the cost of dividing up a task into its parallel components and managing the parallel processes versus the speedups that are obtained.

Since there is no unifying formal framework within which to examine these questions generally, analysis of results and comparison with other systems are difficult. However, classifier systems are powerful and flexible enough to support examination of these tradeoffs in the context of the particular problem, classification in KL-ONE. Chapter 6 presents a detailed analysis of the results of this project, discussing the complexity of the algorithms and the computational tradeoffs within the formalism.

此为试读，需要完整PDF请访问：www.ertongbook.com