CONNECTIONIST-SYMBOLIC INTEGRATION

From Unified to Hybrid Approaches

EDITED BY

Ron Sun

Frederic Alexandre

CONNECTIONIST-SYMBOLIC INTEGRATION

From Unified to Hybrid Approaches

CONNECTIONIST-SYMBOLIC INTEGRATION

From Unified to Hybrid Approaches

EDITED BY

Ron Sun
The University of Alabama
Tuscaloosa, AL, USA

Frederic Alexandre
CRIN-CNRS/Inria-Lorraine
Vandoeuvre-les-Nancy, France

LEA
1997
LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS
MAHWAH, NEW JERSEY LONDON

Lawrence Erlbaum Associates, Inc., Publishers
10 Industrial Avenue
Mahwah, New Jersey 07430

Library of Congress Cataloging-in-Publication Data

Connectionist-symbolic integration : from unified to hybrid approaches / edited by Ron Sun, Frederic Alexandre.
p. cm.
Includes bibliographical references and index.
ISBN 0-8058-2348-4 (cloth : alk. paper). -- ISBN 0-8058-2349-2 (pbk. : alk. paper)
1. Hybrid computers. 2. Systems engineering. 3. Neural networks (Computer science) I. Sun, Ron, 1960- . II. Alexandre, Frederic.
QA76.38.C66 1997
006.3--dc21 97-30217
CIP

Books published by Lawrence Erlbaum Associates are printed on acid-free paper, and their bindings are chosen for strength and durability.

10 9 8 7 6 5 4 3 2 1

CONTENTS

PREFACE

This book is concerned with the development, analysis, and application of hybrid connectionist-symbolic models in artificial intelligence and cognitive science, drawing contributions from an international group of leading experts. It describes and compares a variety of models in this area. The types of models described in this book cover a wide range of the evolving spectrum of hybrid models. Thus, it serves as a well-balanced progress report on the state of the art in this area. We hope that it will also stimulate its future development.

This book is the outgrowth of *The IJCAI Workshop on Connectionist-Symbolic Integration: From Unified to Hybrid Approaches*, which was held for two days during August 19-20 in Montreal, Canada, in conjunction with the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95). The workshop was co-chaired by Ron Sun and Frederic Alexandre. It featured 23 presentations, including two invited talks, and two panel discussions. During the two days of the workshop, various presentations and discussions brought to light many new ideas, controversies, and syntheses, which lead to the present volume.

We hereby wish to thank all the participants of the workshop for their contributions that lead to the present book. We expecially would like to thank the members of the program and organization committees who reviewed papers or in other ways helped the organization of the workshop: John Barnden, Steve Gallant, Larry Medsker, Christian Pellegrini, Noel Sharkey, Lawrence Bookman, Michael Dyer, Wolfgang Ertel, LiMin Fu, Jose Gonzalez-Cristobal, Ruben Gonzalez-Rubio, Jean-Paul Haton, Melanie Hilario, Abderrahim Labbi, and Ronald Yager. We thank the two invited speakers at the workshop: Jim Hendler and Noel Sharkey, as well as each of the panelists. We also thank the editors at Lawrence Erlbaum Associates for their part in producing this book.

CONTRIBUTORS

Frederic Alexandre
Crin-Cnrs/Inria-Lorraine
BP 239
54500 Vandoeuvre-les-Nancy, France
falex@loria.fr

Bernard Amy
LEIBNIZ
46 ave Felix Viallet
38031 Grenoble
France

Jim Austin
Dept. of CS
University of York
York, YO1 5 DD, UK
austin@minster.york.ac.uk

Haihong Dai
Department of Computer Science
Queen's University of Belfast
Belfast BT7 1NN, UK
h.dai@qub.ac.uk

Jose C. Gonzalez
Dep. Ingenieria de Sistemas Telematicos
E.T.S.I. Telecomunicacion
Universidad Politecnica de Madrid
E-28040 Madrid, Spain
jcg@gsi.dit.upm.es

Melanie Hilario
Centre Universitaire d'Informatique
University of Geneva
24, rue du General Dufour
CH-1211 Geneva 4, Switzerland
hilario@cui.unige.ch

Carlos A. Iglesias
Dep. Teoria de la Senal, Comunicaciones e Ing. Telematica
E.T.S.I. Telecomunicacion
Universidad de Valladolid
E–47011 Valladolid, Spain
cif@tel.uva.es

Todd Johnson
Ohio State Univ.
Medical Informatics
Columbus, OH 43210
tj@medinfo.ohio-state.edu

Rajiv Khosla
Expert and Intelligent Systems Laboratory
Department of Computer Science and Computer Engineering
La Trobe University
Melbourne, Victoria 3083, Australia
khosla@latcs1.lat.oz.au

Boicho Kokinov
Institute of Mathematics
Bulgarian Academy of Sciences
Bl.8, Acad. G. Bonchev Str.
Sofia 1113, Bulgaria
kokinov@bgearn.bitnet

Ramon Krosley
Dept. MCS
Colarado School of Mines
Golden, CO 80401
rkrosley@mines.colorado.edu

Abderrahim Labbi
Department of Computer Science
University of Geneva
24, Rue du General Dufour CH-1211
Geneve 4
Switzerland
labbi@cui.unige.ch

Yannick Lallement
Crin-Cnrs/Inria-Lorraine
BP 239
54500 Vandoeuvre-les-Nancy, France
lallement@loria.fr

Luis Magdalena
Dep. Matematica Aplicada
E.T.S.I. Telecomunicacion
Universidad Politecnica de Madrid
E-28040 Madrid, Spain
llayos@mat.upm.es

Maria Malek
LEIBNIZ
46, ave Felix Viallet
38031 Grenoble
France
maria.malek@imag.fr

Michael McTear
School of Information & Software Engineering
University of Ulster at Jordanstown
Newtownabbey
County Antrim BT37 0QB, UK
m.mctear@ulster.ac.uk

Manavendra Misra
Dept of Mathematical and Computer Sciences
Colorado School of Mines
Golden, CO 80401
mmisra@mines.edu

Eric Mjolsness
Machine Learning Systems Group
Jet Propulsion Laboratory, Caltech
4800 Oak Grove Drive
Pasadena, CA 91109

Piyush Ojha
School of Information & Software Engineering
University of Ulster at Jordanstown
Newtownabbey
County Antrim BT37 0QB, UK
p.ojha@ulster.ac.uk

Bruno Orsier
Lab for Artificial Brain Systems
The Institute for Physical and Chemical Research (RIKEN)
Hirosawa 2-1, Wako-shi, Saitama 351-01,
Japan
orsier@zoo.riken.go.jp

Nam Seog Park
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh, EH1 1HN UK.
namseog@aisb.ed.ac.uk

Todd Peterson
Department of Computer Science
The University of Alabama
Tuscaloosa, AL 35487
todd@cs.ua.edu

Tony Plate
School of Mathematical and Computing Sciences
Victoria University of Wellington
PO Box 600, Wellington, New Zealand
TonyPlate@vuw.ac.nz

Dave Robertson
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh, EH1 1HN UK.
dr@dai.ed.ac.uk

Jose C. Gonzalez
Universidad Politecnica de Madrid
Dept Ingenieria de Sistemas Telematicos
ETSI Telecomunicacion
Cuidad Universitaria
E-28040 Madrid, Spain
jgonzalez@dit.upm.es

Alessandro Sperduti
University of Pisa
Dipartimento di Informatica
Corso Italia 40
56125 Pisa, Italy
perso@di.unipi.it

Antonio Starita
University of Pisa
Dipartimento di Informatica
Corso Italia 40
56125 Pisa, Italy
starita@di.unipi.it

Suzanne Stevenson
Department of Computer Science
Center for Cognitive Science
CORE Bldg, Busch Campus
Rutgers University
New Brunswick, NJ 08903
suzanne@cs.rutgers.edu

Ron Sun
Department of Computer Science
Department of Psychology
The University of Alabama
Tuscaloosa, AL 35487
rsun@cs.ua.edu

Juan R. Velasco
Dep. Ingenieria de Sistemas Telematicos
E.T.S.I. Telecomunicacion
Universidad Politecnica de Madrid
E-28040 Madrid, Spain
juanra@gsi.dit.upm.es

Xinyu Wu
School of Information & Software Engineering
University of Ulster at Jordanstown
Newtownabbey
County Antrim BT37 0QB, UK
w.xinyu@ulster.ac.uk

Jiajie Zhang
Ohio State University
Dept of Psychology
Columbus, Ohio 43210
zhang@canyon.psy.ohio-state.edu

1

AN INTRODUCTION TO HYBRID CONNECTIONIST-SYMBOLIC MODELS

Ron Sun

Department of Computer Science
The University of Alabama

1 MOTIVATIONS

There has been a considerable amount of research in integrating connectionist and symbolic processing. While such an approach has clear advantages, it also encounters serious difficulties and challenges. Consequently, various ideas and models have been proposed to address different problems and different aspects in this integration. The need for such models has been slowly but steadily growing over the past five years, from many segments of the artificial intelligence and cognitive science communities, ranging from expert systems to cognitive modeling and to logical reasoning. Some interesting and important approaches have been developed. There has been a general consensus that hybrid connectionist-symbolic models constitute a promising avenue toward developing more robust, more powerful, and more versatile architectures, both for cognitive modeling and for intelligent systems. It is definitely worthwhile pursuing research in this area further still, which might generate important new ideas and significant new applications.

The basic motivations for research in hybrid connectionist-symbolic models can be briefly summarized as follows:

- Cognitive processes are not homogeneous; a wide variety of representations and mechanisms are employed. Some parts of cognitive processes are best captured by symbolic models, while others by connectionist models (Smolensky 1988, Sun 1995). Therefore, a need for "pluralism" exists in cognitive modeling, which leads to the development of hybrid models as tools and frameworks.

- The development of intelligent systems for practical applications can benefit greatly from a proper combination of different techniques, since no one single technique can do everything, as is the case in many application domains, ranging from bank loan approval to industrial process control (Medsker 1994). By combining different techniques, intelligent systems can explore the synergy of these techniques.
- To develop a full range of capabilities in autonomous agents, an autonomous agent architecture needs to incorporate both symbolic and subsymbolic processing for handling declarative and procedural knowledge, respectively, in order to effectively deal with a variety of environments in which an agent finds itself (Sun and Peterson 1995). Such an agent architecture, incorporating both conceptual and subconceptual processes, leads naturally to a combination of symbolic models (which capture conceptual processes) and connectionist models (which capture subconceptual processes).

The book tries to bring to light many new ideas, controversies, and syntheses in this broad area. The focus is on learning and architectures that feature hybrid representations and support hybrid learning.

2 IMPORTANT ISSUES

There have been many important and/or crucial issues that have been raised with regard to hybrid connectionist-symbolic models. These issues concern architectures of these models, learning in these models, and various other aspects.

Hybrid models involve a variety of different types of processes and representations, in both learning and performance. Therefore, multiple mechanisms interact in complex ways in most of these models. We need to consider seriously ways of structuring these different components; in other words, we need to consider *architectures*, which thus occupy a clearly more prominent place in this area of research compared with other areas in AI. Some architecture-related issues are as follows:

- What type of architecture facilitates what type of process?
- Should hybrid architectures be modular or monolithic?

- For modular architectures, should we use different representations in different modules of an architecture or should we use the same representation throughout?
- How do we decide if a particular part of an architecture should be symbolic, localist, or distributed in its representation?
- How do we structure different representations in different parts to achieve optimal results?
- How do we incorporate prior knowledge into hybrid architectures?

Although purely connectionist models, which constitute a part of any hybrid model, are known to excel in their learning abilities, hybridization makes it more difficult to do learning. Most symbolic models and architectures are not specifically designed to perform learning, especially not in a fully autonomous and bottom-up fashion, and most of them have difficulties with learning in some ways. Therefore, the hybridization of connectionist and symbolic models inherits the difficulty of learning from the symbolic side and mitigates to some large extent the advantage that the purely connectionist models have in their learning abilities. Considering the importance of learning in both modeling cognition and building intelligent systems, it is crucial for researchers in this area to pay more attention to ways of enhancing hybrid models in this regard and to putting learning back into hybrid models. Some of the learning-related issues that need to be addressed include:

- How can learning be incorporated and utilized in each type of architecture?
- What kinds of learning can be done in each type of architecture, respectively?
- How do learning and representation interact along the developmental line?
- What is the relationship between symbolic machine learning methods, knowledge acquisition methods, and connectionist (neural network) learning algorithms, especially in the context of hybrid models?
- How can each type of architecture be developed with various combinations of the above-mentioned methods?
- How can learning algorithms be developed for (usually knowledge-based) localist connectionist networks?

1. single-module	
* representation	symbolic, localist, distributed
* mapping	direct translational, transformational
2. heterogeneous multi-module	
* components	localist+distributed, symbolic+connectionist
* coupling	loosely coupled, tightly coupled
* granularity	coarse-grained, fine-grained
3. homogeneous multi-module	
* granularity	coarse-grained, fine-grained

Figure 1 Classifications of Hybrid Models

- How can rules be extracted from, and refined by, (hybrid) connectionist models?
- How can complex symbolic structures besides rules, such as frames and semantic networks, be learned in hybrid connectionist models?

3 ARCHITECTURES

In terms of architectures of hybrid models, various distinctions, divisions, and classifications have been proposed and discussed. (see chapter 2). As a first cut, we can divide these models up into two broad categories: *single-module* architectures and *multi-module* architectures (including both homogeneous and heterogeneous multi-module architectures). See Figure 1.

For single-module architectures, along the *representation* dimension, there can be the following types of representations (see Sun and Bookman 1994): symbolic (as in conventional symbolic models, in which case, the model is no longer a hybrid model), localist (with one distinct node for representing each concept; for example, Lange and Dyer 1989, Sun 1992, Shastri and Ajjanagadde 1993, Barnden 1994), and distributed (with a set of non-exclusive, overlapping nodes for representing each concept; for example, Pollack 1990, Sharkey 1991). Usually, it is easier to incorporate prior knowledge into localist models since their structures can be made to directly correspond to that of symbolic knowledge (Fu 1991). On the other hand, connectionist learning usually leads to distributed representation, such as in the case of backpropagation learning. Along

a different dimension, in terms of *mappings* between symbolic and connectionist structures (Hilario 1995, Medsker 1994), we see that there are the direct translational approach, which creates a network structure that directly corresponds to the symbolic structure to be implemented (usually in a localist network), such as in the implementation of rules in a backpropagation network by Fu (1991) and Towell and Shavlik (1993), and the transformational approach, which creates the equivalent of symbolic structures in connectionist networks without actually embedding the structures directly in networks, such as the encoding of trees in RAAM (Pollack 1990). The relative advantage of each is a still unsettled issue (which is related to the compositionality issue as being debated in the theoretical community). Another possible dimension is in terms of the *dynamics* of the models, rather than in terms of the static topology (i.e., the static mapping) of the networks used; that is, we can classsify models based on whether their internal dynamics is translational or transformational, which can be highly correlated with but not necessarily identical to the static topology of networks.

For multi-module models, we can distinguish between *homogeneous* models and *heterogeneous* models. Homogeneous models may be very much like a single-module model discussed above, except they contain several replicated copies of the same underlying structure, each of which can be used for processing the same set of inputs, to provide redundancy for various reasons. For example, we can have competing experts (of the same domain), each of which may vote for a particular solution. Or, each module (of the same makeup) can be specialized (content-wise) for processing a particular type of input or another; for example, we can have different experts with the same structure and representation but different content/knowledge for dealing with different situations.

For heterogeneous multi-module models, a variety of distinctions can be made. First of all, a distinction can be made in terms of *representations* of constituent modules. In multi-module models, there can be different combinations of different types of constituent modules: for example, a model can be a combination of localist and distributed modules (for example, CONSYDERR as described in Sun 1995, for cognitive modeling of commonsense reasoning and decision making), or it can be a combination of symbolic modules and connectionist modules (either localist or distributed; for example, SCRUFFY as described in Hendler 1991, mainly for practical applications).

Another distinction that can be made is in terms of the *coupling* of modules: a set of modules can be either loosely coupled or tightly coupled (Medsker 1994). In loosely coupled situations, modules communicate with each other, primarily through some interfaces as in, for example, SCRUFFY (Hendler 1991). Such

loose coupling enables some loose forms of cooperation among modules. One form of cooperation is in terms of pre/postprocessing vs. main processing: while one or more modules take care of pre/postprocessing, such as transforming input data or rectifying output data, a main module focuses on the main part of the processing task. This is probably the simplest and earliest form for hybrid systems, in which, commonly, pre/post processing is done using a connectionist network and the main task is accomplished through the use of symbolic methods (as in conventional expert systems). Another form of cooperation is through master-slave relationships: while one module maintains control of the task at hand, it can call upon other modules to handle some specific aspects of the task. For example, a symbolic expert system, as part of a rule, may invoke a neural network to perform a specific classification or decision making or some other processing. A variation of this form is the processor-monitor (meta-processor) combination, in which a processing module does the work while a monitor module waits for certain events to occur in which case the monitor will inform and/or alter the working of the processing module. Yet another form of cooperation is the equal partnership of multiple modules. In this form, the modules (the equal partners) can consist of (1) complementary processes, such as in the SOAR/ECHO combination (see chapter 6), or (2) multiple functionally equivalent but representationally different processes, such as in the CLARION architecture (chapter 7), or (3) they may consist of multiple differentially specialized and heterogeneously represented experts each of which constitutes an equal partner in accomplishing the task. [1]

In tightly coupled systems, on the other hand, the constituent modules interact through multiple channels or may even have node-to-node connections across two modules, such as CONSYDERR (Sun 1995) in which each node in one module is connected to a corresponding node in the other module. For tightly coupled multi-module systems, there are also a variety of different forms of cooperation among modules, in ways quite similar to loosely coupled systems. Such forms include master-slave, processor-monitor, and equal partnership, each of which is basically the same as in loosely coupled systems, except in this case a larger number of connections exist and a lot more interactions are occuring among modules. However, another possibility in loosely coupled systems, i.e., pre/post-processing, is not one of the possibilities with tightly coupled systems, since it entails loose connections between the pre/post-processing module and the main processing modules.

[1] These forms have been referred to as *subprocessing*, *metaprocessing*, and *coprocessing* in chapter 2.

Another distinction that can be made of all multi-module systems is with regard to the *granularity* of modules in such systems: they can be coarse-grained or fine-grained. On one end of the spectrum, a multi-module system can be very coarse-grained so that it contains only two or three modules. On the other end of the spectrum, a system can be so fine-grained that it can contain numerous modules, such as the case in DUAL (see Kokinov 1995; see also chapters 11 and 12). Sometimes, in an extremely fine-grained system, each tiny module may contain both a (simple and tiny) symbolic component and a (simple and tiny) connectionist component. Such a form is termed "micro-level" integration of symbolic and connectionist models (by Kokinov 1995), as opposed to "macro-level" integration in which each module is much more powerful and complete and contains one type of model only. The advantage of such "micro-level" integration, computationally speaking, is that we can have a vast number of simple "processors" (i.e., fine-grained integrated modules) that constitute a uniform and massively parallel system that combines the power of connectionist as well as symbolic models. Such a system, in a way, is a homogeneous system in the sense discussed earlier.

4 LEARNING

One fundamental issue that clearly requires more attention from researchers in this area has been highlighted in this book: the issue of learning, which includes both learning the content/knowledge of an architecture as well as learning and developing the architectures themselves. Learning is necessary, both because it is a fundamental process of intelligence/cognition, and because it is practically indispensable in scaling up to large-scale systems.

Looking back to the proceedings of earlier meetings, earlier collections of papers, and/or earlier special issues of journals dealing with hybrid models, such as Hinton (1991), Sun et al. (1992), and Sun and Bookman (1994), the treatment of learning has been sparse. Many models were presented as simply representational ones: that is, a framework in which both symbolic and connectionist knowledge can coexist and can be represented in some ways, but not necessarily acquired automatically. The earlier workshop on this topic, as reported in Sun et al. (1992), was almost exclusively focused on representational issues. Such a focus might be justified at the early stage of development of this research area, since before we can learn complex symbolic representation in connectionist and hybrid connectionist models, we need to figure out ways of representing complex symbolic structures in the first place.

In Sun and Bookman (1994), the following question was raised:

> How can more powerful learning algorithms be developed that can acquire complex symbolic representation in connectionist systems, including localist systems? This is a difficult issue, in that simple learning algorithms that build up functional mappings in typical neural network models are insufficient for symbolic processing connectionist networks, because of the discrete and discontinuous nature of symbolic processes and because of the systematicity of such processes. Newer and more powerful learning algorithms are needed that can extract symbolic structures from data and/or through interaction with environments. ..... Such algorithms should somehow incorporate some symbolic methods, as more powerful learning algorithms will result from such incorporation. (see p.9, Sun and Bookman 1994.)

Now, after a number of years of maturation, the hybrid model area is ready to take on the real challenge of learning: not only of simple procedural skills, but also of complex symbolic structures, and even of architectures themselves. Such learned representations should be linked closely to the context of their use, not as a stand-alone showcase of the "power" of a particular learning method.

A number of chapters in this book deal with the issue of learning, each to a different extent. Chapter 6 presents a model for abductive reasoning that learns its internal representation through a combination of symbolic and connectionist methods, aimed at cognitive modeling. Chapter 18 shows how RAAM (Pollack 1990) can be extended to deal with the learning of logical term classification in symbolic reasoning. Chapter 7 (also Sun and Peterson 1995) presents a model for learning sequential decision making in which symbolic declarative knowledge is extracted online from a reinforcement learning connectionist network and is used in turn to speed up learning and to facilitate transfer. Thus, it demonstrates not only the synergy between connectionist and symbolic learning but also the point that symbolic knowledge can be learned autonomously in a bottom-up fashion, which is very useful in developing autonomous agents.

The future advance in this area is dependent on the progress in the development of new learning methods in hybrid systems and the integration of learning with complex symbolic representations. As was suggested in some chapters here, symbolic representation and reasoning may well emerge from subsymbolic processes, and a synergistic combination of symbolic and subsymbolic processes is thus possible (see chapter 7 and also Giles et al. 1995).

5 SUMMARY

In summary, a variety of ideas, approaches, and techniques exist, in terms of both architectures and learning, and this abundance seems to lead to many exciting possibilities in theoretical advances (for example, in learning and knowledge acquisition) and in application potentials. We need to extend more effort to exploit the possibilities and opportunities in this area.

Despite the apparant diversity, there is clearly an underlying unifying theme: architectures that bring together symbolic and connectionist models to achieve a synthesis and the synergy of the two different paradigms (and the learning and knowledge acquisition methods for developing such architectures). With this book, we hope to provide an information clearinghouse for various proposed approaches and models that share the common belief that connectionist and symbolic models can be usefully combined and integrated, and such integration may lead to significant advances in our understanding of intelligence.

REFERENCES

[1] J. Barnden, (1994). Complex symboli-processing in CONPOSIT. In: R. Sun and L.Bookman, (eds.) *Computational Architectures Integrating Neural and Symbolic Processes.* Kluwer. Boston, MA.

[2] L.M. Fu, (1991). Rule learning by searching on adapted nets, *Proc. of AAAI'91*, pp. 590-595.

[3] C.L. Giles, B.G. Horne, and T. Lin, (1995). Learning a class of large finite state machines with a recurrent neural network. *Neural Networks*, vol.8, no.9, pp. 1359-1365.

[4] J. Hendler, (1991). Developing hybrid symbolic/connectionist models. In: J. Barnden and J. Pollack (eds.), *Advances in Connectionist and Neural Computation Theory*, pp. 165–179. Lawrence Erlbaum Assocciates. Hillsdale, NJ.

[5] M. Hilario, (1995). An overview of strategies for neurosymbolic integration. In: R. Sun and F. Alexandre, (eds.) *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches.* IJCAI, Montreal, Canada.

[6] J. Hinton, (1991). *Connectionist Symbolic Processing: A Special Issue Of Artificial Intelligence.* MIT Press. Cambridge, MA.

[7] T. Johnson and J. Zhang, (1995). A hybrid learning model of abductive reasoning. In: R. Sun and F. Alexandre, (eds.) *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches.* IJCAI, Montreal, Canada.

[8] B. Kokinov, (1995). Micro-level hybridization in DUAL. In: R. Sun and F. Alexandre, (eds.) *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches.* IJCAI, Montreal, Canada.

[9] T. Lange and M. Dyer, (1989). High-level inferencing in a connectionist network. *Connection Science*, 1, pp. 181–217.

[10] L. Medsker, (1994). *Hybrid Neural Networks and Expert Systems.* Kluwer. Boston, MA.

[11] J. Pollack, (1990). Recursive distributed representation. *Artificial Intelligence.* 46 (1-2). pp. 77-106.

[12] N. Sharkey, (1991). Connectionist representation techniques. *AI Review*, 5. pp. 142-167.

[13] L. Shastri and V. Ajjanagadde, (1993). From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings. *Behavioral and Brain Sciences*, 16(3). pp. 417-494.

[14] P. Smolensky, (1988). On the proper treatment of connectionism. *Behavorial and Brain Sciences*, 11(1). pp. 1-74.

[15] R. Sun, (1992). On Variable Binding in Connectionist Networks, *Connection Science*, Vol.4, No.2, pp. 93-124.

[16] R. Sun, (1995). Robust reasoning: integrating rule-based and similarity-based reasoning. *Artificial Intelligence.* 75. pp. 241-295.

[17] R. Sun and T. Peterson, (1995). A hybrid learning model for sequential decision making. *The IJCAI Workshop on Connectionist-Symbolic Integration: From Unified to Hybrid Approaches.* IJCAI, Montreal, Canada.

[18] R. Sun and L. Bookman, (eds.) (1994). *Computational Architectures Integrating Neural and Symbolic Processes.* Kluwer Academic Publishers. Boston, MA.

[19] R. Sun, L Bookman, and S. Shekhar, (1992). *The Working Notes of the AAAI Workshop on Integrating Neural and Symbolic Processes: The Cognitive Dimension.* San Jose, CA.

[20] G. Towell and J. Shavlik, (1993). Extracting Refined Rules from Knowledge-Based Neural Networks, *Machine Learning.*

PART I

REVIEWS AND OVERVIEWS

2

AN OVERVIEW OF STRATEGIES FOR NEUROSYMBOLIC INTEGRATION

Mélanie Hilario

Computer Science Center
University of Geneva

1 INTRODUCTION

Throughout its brief history, the field of artificial intelligence (AI) has been the arena of jousts between two *frères ennemis*, symbolicism and connectionism. No sooner had connectionism recovered from Minsky and Papert's (1988) devastating blows than Fodor and Pylyshyn (1988) charged to the fore in the name of symbolic AI. They argued that connectionism cannot be a valid theory of cognition since it fails to account for the combinatorial syntactic and semantic structure of mental representations; at best, connectionism is just another implementation technology, an alternative means of implementing classical symbolic structures and processes. This implementationalist viewpoint has since been the traditional defense of symbolic AI against connectionism's cognitive claims. At the other extreme, according to Pinker and Prince's (1988) classification, eliminativism rejects the symbol level as a valid level of description of cognitive phenomena; symbolic theories are no more than crude approximations of what really takes place in the brain and must give way to connectionist or neural theories.

Between these two radical stances, a number of more subtle philosophies have emerged at the interface of connectionist and symbolic AI. Their origins have been inextricably linked with the proliferation of attempts at integrating neural and symbolic processing. This paper will give an overview of the various approaches to neurosymbolic integration. Roughly, these can be divided into two strategies: *unified* strategies aim at attaining neural and symbolic capabilities using neural networks alone, while *hybrid* strategies combine neural networks with symbolic models such as expert systems, case-based reasoning systems,

and decision trees. These two approaches form the main subtrees of the classification hierarchy depicted in Figure 1.

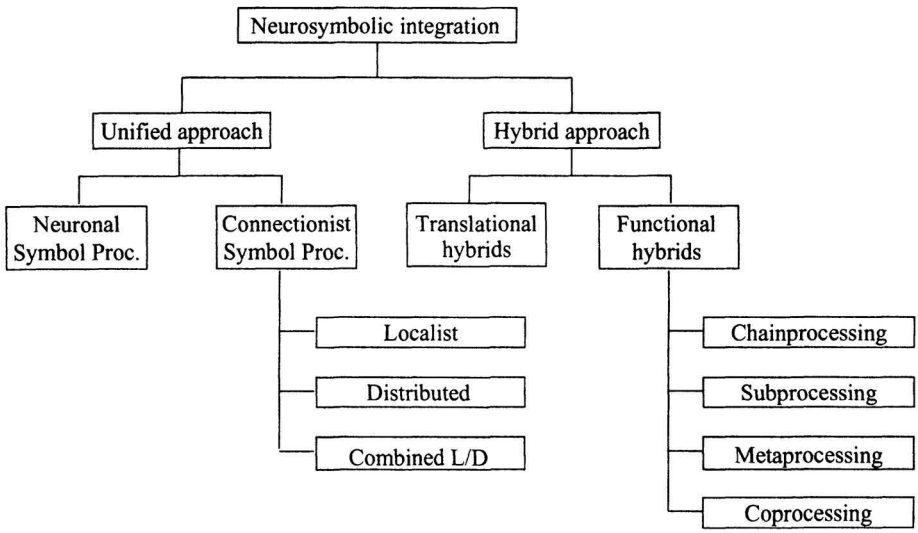

Figure 1 Classification of integrated neurosymbolic systems.

This chapter is organized as follows. Sections 2 and 3 discuss unified and hybrid strategies respectively; [1]. Section 4 relates these strategies to the various philosophical stances that have been observed in the literature vis-a-vis the relationship between connectionist and symbolic processing. Section 5 explores the major computational issues involved in neurosymbolic integration and Section 6 concludes.

2 UNIFIED STRATEGIES

Unified strategies are premised on the claim that there is no need for symbolic structures and processes as such; full symbol processing functionalities emerge from neural structures and processes alone. They can be subdivided into two distinct trends: neuronal symbol processing and connectionist (or neural) symbol processing. This distinction is based on Reeke and Edelman's (1988) ter-

[1] This overview does not include fuzzy-neural integration strategies, which are discussed in chapter 5 of this volume

minological convention, according to which the term *neuronal* denotes a close identification with the properties of actual (biological) neurons whereas *neural* implies only a general similarity to actual neurons.

2.1 Neuronal Symbol Processing

The neuronal approach aims at grounding all cognitive processes in biological reality. One particular case of this approach is neuronal symbol processing (NSP), whose specific objective is to model the brain's high-level functions. The neuronal approach is a bottom-up approach: its mandatory starting point is the biological neuron. Perhaps the most brilliant example of the neuronal approach is the theory of neuronal group selection (TNGS), better known as neural Darwinism (Edelman 1987). Built on three fundamental tenets—developmental selection, experiential selection and reentrant mapping—this theory attempts to provide a biological account of the full range of cognitive phenomena, from sensorimotor responses all the way up to concept formation, language, and higher order consciousness. The consistency of the TNGS has been demonstrated in a series of automata which avoid the preestablished categories and programming of standard AI. Constructed as networks of neuronlike units undergoing a process of natural selection, these automata carry out categorization and association tasks in a dynamic environment. In Darwin III (Edelman 1992), for example, recognition and categorization networks are combined with motor circuits and effectors that act on the environment. Objects are categorized on the basis of internal values like "light is better than no light"; the result of the automaton's neuronal activity becomes apparent as motor responses to categorized objects. The processes demonstrated in these automata—perceptual categorization, memory and learning—are precisely the fundamental triad of higher order brain functions, according to the TNGS. However, effective emergence of these higher level functions remains to be demonstrated in the Darwin or its descendant series. Neuronal symbol processing may yet be the ultimate proof-of-concept of the neuronal approach; however, the field is as immature as its ambitions are high, and it may take some time before real-world applications can even be envisaged.

2.2 Connectionist Symbol Processing

Connectionist symbol processing (CSP) or neural symbol processing lays no claim to neurobiological plausibility; the neuron in question here is generally a formal neuron. Artificial neural networks are used as building blocks to

create a cognitive architecture capable of complex symbol processing. Typically, model construction starts with an idea of some high-level symbolic function to be performed and proceeds with the design of the appropriate connectionist infrastructure. In this sense, the neural approach can be thought of as a top-down strategy, despite the opposite thrust of its claim that complex symbolic functions emerge from neural structures and processes. However, CSP is not inherently top-down: in principle, nothing precludes it from actually starting out with neural networks from which non-predetermined symbolic structures and processes can emerge in unforeseen ways.

Historically, Fodor and Pylyshyn's (1988) critique has been a significant if negative driving force behind CSP: one of its persistent motivations has been to show that neural networks exhibit a combinatorial constituent structure—precisely what Fodor and Pylyshyn declared wanting in connectionist architectures. For instance, BoltzCONS is a connectionist model that dynamically creates and manipulates linked lists; according to its author, its aim is not just to implement complex symbol structures using neural networks, but rather to show how neural networks can exhibit compositionality and distal access, two distinguishing properties of high-level symbol processing (Touretzky 1990).

Work in connectionist symbol processing can be classified along two dimensions. From the point of view of the underlying representation scheme, CSP architectures can be localist, distributed, or combined localist/distributed. Localist architectures use a one-to-one mapping between individual units and symbolic structures. Each node in a neural network represents a concept or a combination of concepts, e.g., a two-place predicate, a relation, a rule (Shastri 1988, Ajjanagadde and Shastri 1991, Sohn and Gaudiot 1991, Lange 1992, Feldman and Ballard 1992). The principal disadvantage of localist architectures is that they quickly succumb to combinatorial explosion as the number of individual concepts increases. This has motivated the development of distributed architectures, where the most elementary concepts emerge from the interaction of several different nodes. Each knowledge item (e.g., concept, fact or rule) is represented using a combination of several units, and each unit can be used in the representation of several items. DCPS (Touretzky and Hinton, 1988), for example, is a distributed connectionist production system that uses coarse coding to store all entities manipulated by the rule interpreter. Each unit has a receptive field that is the cross-product of the six symbols in each of its three colums, given 216 triples per field. As a result, this distributed coding scheme can be used to construct a working memory that requires far fewer units than the number of facts that can potentially be stored. Coarse coding is also used in BoltzCONS (Touretzky 1990) to represent symbolic structures such as lists and stacks, whereas recursive distributed representations introduced in

Pollack's (1990) RAAM have been used to implement tree-matching (Stolcke and Wu, 1992). Finally, local/distributed architectures combine systems using these two representations as separate modules. The main justification of these systems is that they combine the efficiency of distributed representations with the power of localist representations. For instance, whereas DCPS is a highly constrained system that can represent only two triples in a rule's left-hand side, RUBICON (Samad 1992), a connectionist rule-based system that uses a combined localist/distributed rule-based system, allows for a variable number of expressions in the left and right hand sides of each rule. It also supports chain inferencing as well as addition and deletion of working memory elements.

From the point of view of system tasks, the CSP approach has been actively investigated in a variety of task domains, particularly in automated reasoning and natural language understanding. CONSYDERR (Sun 1991) implements commonsense reasoning using a localist network which performs rule-based reasoning and a distributed network which encodes feature similarities. The CHCL system (Hölldobler and Kurfess 1991) embodies a connectionist inference mechanism which uses W. Bibel's well-known connection method (Bibel 1987) to perform inferencing on Horn clauses using a matrix representation. An important subarea of work on logic and reasoning concerns variable binding (Sun 1992, Ajjanagadde and Shastri 1991, Touretzky and Hinton 1988, Chen 1992, Smolensky 1990, Pinkas 1994, Park and Robertson 1995). Indeed, this crucial problem needs to be solved if neural networks are to equal the expressive power of symbolic systems, which perform first-order predicate logic tasks as a matter of routine. Techniques experimented in the field of automated reasoning have been applied in connectionist expert systems such as MACIE (Gallant 1988, Gallant 1993), TheoNet (Bradshaw et al. 1989) and others (Saito and Nakano 1988, Hayashi 1991). Examples of studies in natural language processing via connectionist symbol processing can be found in (Bookman 1987, Dyer 1991, McClelland and Kawamoto 1986, Gasser 1988).

3 HYBRID STRATEGIES

The hybrid approach rests on the assumption that only the synergistic combination of neural and symbolic models can attain the full range of cognitive and computational powers. Hybrid neurosymbolic models can be either translational or functional hybrids.

3.1 Translational Hybrids

Translational hybrids—also called transformational models (Medsker 1994)—can be viewed as an intermediate class between unified models and functional hybrids. Like unified models, they rely only on neural networks as processors, but they can start from or end with symbolic structures. Typically, their objective is to translate or transform symbolic structures into neural networks before processing, or extract symbolic structures from neural networks after processing. Most often, the symbolic structures used are rules—classical propositional rules (Towell and Shavlik 1994, Dillon et al. 1994), fuzzy rules (Romaniuk and Hall 1991, Hayashi 1991, Magrez and Rousseau 1992), rules with certainty factors or probability ratings (Lacher et al. 1992, Fu 1989, Fu and Fu 1990, Mahoney and Mooney 1994, Tresp et al. 1993). Attempts have also been made to compile differential equations (Cozzio 1995) or deterministic finite state automata (Giles and Omlin 1993) into neural networks, and to extract hierarchies of concepts or schemata from them (Crucianu and Memmi 1992, Dillon et al. 1994). However, the key point is that symbolic structures are not processed as such in translational systems; for instance, rules are not applied by an inference engine within the system but only serve as source or target representations of knowledge built into neural nets. The implicit assumption seems to be that, whether purely connectionist systems are capable of full symbol processing or not, interaction with other (human or symbolic) systems imposes the need for a two-way transformation between neural network structures and high-level symbolic representations.

3.2 Functional Hybrids

Functional hybrids incorporate *complete* symbolic and connectionist components: in addition to neural networks, they comprise both symbolic structures and their corresponding processors—e.g., rule interpreters, parsers, case-based reasoners and theorem provers. Functional hybrids are so-called because, contrary to translational hybrids, they achieve effective functional interaction and synergy among the combined components. In a sense, translational hybrids are a degenerate case of functional hybrids; in the rest of this chapter, we shall therefore use the term *hybrid* to designate complete or functional hybrids, unless indicated otherwise.

Hybrid systems can be distinguished along different dimensions such as their target problem or task domain, the symbolic (e.g., rule-based reasoning, case-based reasoning) and neural (e.g., multilayer perceptrons, Kohonen networks)

models used, or the role played by the neural (N) and symbolic (S) components in relation to each other and to the overall system. Although such dimensions allow for more or less clear distinctions between individual systems, they have little bearing on the central issues of neurosymbolic integration. We therefore propose a taxonomy of hybrid systems based on the degree and the mode of integration of the N and S components.

The **degree of integration** is a quantitative criterion: one can imagine a spectrum going from the simple juxtaposition of symbolic and neural components under a common supervisor to systems characterized by strong and repeated, if not continuous, interaction between the two components. Although this spectrum can be graded numerically to represent progression from one extreme to another, we shall simplify by distinguishing two main degrees of integration—loose and tight coupling. In *loosely coupled* systems, interaction between the two components is clearly localized in space and time: control and data can be transferred directly between N and S components (e.g., by function or procedure calls), or via some intermediate structure (e.g., domain or control blackboards accessible to both components) or agent (e.g. a supervisor), but interaction is always explicitly initiated by one of the components or by an external agent. In *tightly coupled* systems, knowledge and data are not only transferred, they can be shared by the N and S components via common *internal* structures. Thus a change in one of the components which affects these common internal structures has immediate repercussions on the other component without need for explicit interaction initiatives. Within this category, too, coupling is not uniformly tight from one system to another: whereas the shared structures are often simple links or pointers between the N and S components as in SYNHESYS (Giacometti 1992), they can be significantly more important in number and function, e.g., nodes shared by a semantic marker-passing network and a distributed neural network, as in (Hendler 1989).

Along the qualitative dimension, the **integration mode** or scheme refers to the way in which the neural and symbolic components are configured in relation to each other and to the overall system. Four integration schemes have been identified: chainprocessing, subprocessing, metaprocessing and coprocessing (Figure 2). To define them, we suppose a system comprising one neural and one symbolic module, with the understanding that for more complex systems, there can be as many integration schemes as pairs of neural and symbolic components. In chainprocessing mode, one of the (N or S) modules is the main processor whereas the other takes charge of pre and/or postprocessing tasks. In subprocessing mode, one of the two modules is embedded in and subordinated to the other, which acts as the main problem solver. In metaprocessing mode, one module is the base-level problem solver and the other plays a metalevel

role (such as monitoring, control, or performance improvement) vis-à-vis the first. In coprocessing, the N and the S modules are equal partners in the problem solving process: each can interact directly with the environment, each can transmit information to and receive information from the other. The two modules can compete under the supervision of a metaprocessor, or they can cooperate in various ways, e.g., by performing different subtasks or by doing the same task in different ways and/or under different conditions. These four integration modes are described in detail below.

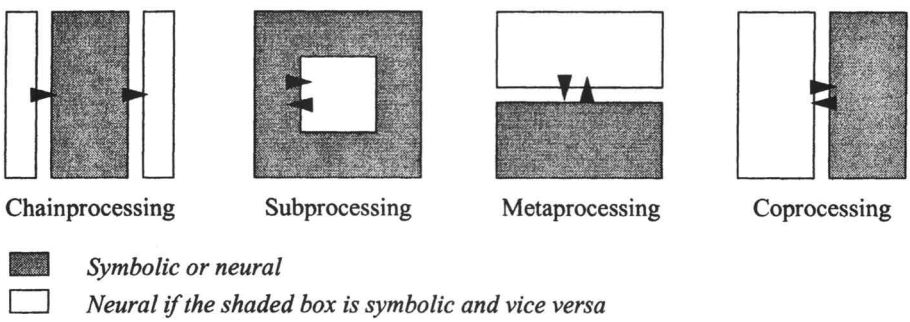

Figure 2 Hybrid neurosymbolic integration modes.

Chainprocessing

Two main configurations can be distinguished in chainprocessing mode: (1) the symbolic module acts as the main problem solver and is assisted by a neural preprocessor and/or postprocessor; (2) the neural module is the main processor and is assisted by a symbolic preprocessor and/or postprocessor.

The first configuration is illustrated by a respiratory monitoring system where a rule-based expert system is assisted by a connectionist preprocessor (Ciesielski et al. 1992, Hayes et al. 1992). The system receives data from a ventilator which records a patient's airway pressure and lung pressure every 15 seconds. At the outset, the system consisted of a simple PC-Expert rulebase whose task was to recommend actions to be taken to avoid breathing complications. Top-level rules were typically: "If *qualitative-state* then *action*," where *qualitative-state* is a symbolic representation of a change in a pressure parameter over time, e.g., "pressure is constant," "pressure is rising gradually," or "pressure is rising rapidly." Determination of these qualitative states on the basis of ventilator output turned out to be the knowledge acquisition bottleneck: while

domain experts found it relatively straightforward to express top-level rules recommending actions on the basis of qualitative states, they had a hard time formulating criteria to determine these qualitative states. Hence the idea of using a neural preprocessor to accomplish this task (see Figure 3).

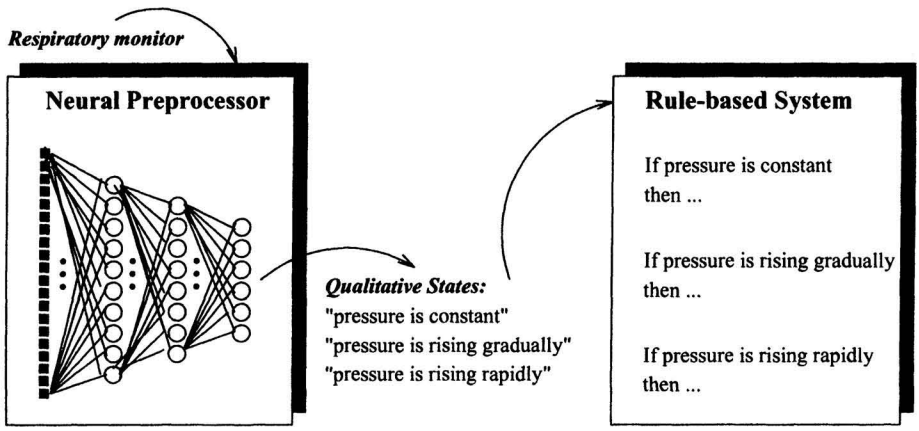

Figure 3 An example of neural preprocessing.

A feedforward neural network was trained using backpropagation. It had 20 inputs (patient data produced by the ventilator), two hidden layers with 10 and 8 units respectively, and 6 output units, each representing a specific interpretation of pressure variations. Its result, the qualitative state corresponding to the output unit with the highest activation, was then stored in the working memory of the expert system for use in the firing of action rules. After training on a set of 54 examples, the accuracy of the systems was measured on a test set of 78 cases. With neural preprocessing, development time was cut down from 3 to 2 months. Moreover, the hybrid system was considerably more accurate (97.5 %) than the rule-only version (74.5 %). It also turned out to be more sensitive, i.e., it detected a respiratory problem after examining fewer data samples than the rule-only system. Understandably, execution time was slightly higher for the hybrid system (1.7 vs 0.5 seconds), though it remained well within the 15-second realtime constraint.

The reverse setup is that of a symbolic preprocessor assisting a connectionist main processor. An example is the combination of ITRULE and neural networks (Goodman 1989, Greenspan et al. 1992). The preprocessor (ITRULE) is a rule induction algorithm which automatically generates probabilistic rules from databases. Learned rules are 0-order rules of the form *If attribute-1 value-1 then attribute-2 value-2 with probability p.* All attributes are binary-valued

so they can be mapped directly to binary units. These rules are then compiled into a neural net: attributes of rule conditions are mapped onto input units, attributes of rule conclusions onto output units, and information metrics (such as p) onto connection weights. The result can then be loaded into the main processor, a neural network simulator. The originality of this approach lies in the fact that contrary to the vast majority of hybrid neurosymbolic systems, learning is done in the symbolic preprocessing phase; the neural network is used for inferential, not for generalization purposes.

Symbolic preprocessing may also be used as a means of alleviating well-known problems related to neural processing. One such problem is the learning time taken by feedforward networks before achieving acceptable error rates. Among the factors that slow down the generalization process is the presence of noise, not only in the feature values, but in the features themselves; more precisely, certain features have no significant effect on the network output and their elimination should result in faster convergence. To limit the number of input nodes in feedforward neural networks, Piramuthu and Shaw (1994) thought of using decision trees as feature selectors. The decision tree algorithm C4.5 (Quinlan 1993) uses information-theoretic measures to select the most important features from a given dataset. The selected features are used as input to a feedforward network which is then trained by backpropagation. The impact of this symbolic preprocessing method on training time was tested on the PROMOTER database, which consists of 106 examples belonging to either class 1 or 0. Each example consisted of 57 attributes and had no missing values. Without preprocessing, a 57–29–1 feedforward network took 84 seconds to converge (total sum of squares of error = 2); its average rate of accuracy over 10 trials was 77.7% on the training set and 55% on the test set. With symbolic preprocessing, the C4.5 algorithm generated a decision tree in 2 seconds and reduced the number of relevant attributes from 57 to 3. The resulting 3–2–1 network converged in 22 seconds, attaining a rate of accuracy of 96.4% for the training set and 66% for the test set. In short, 2 seconds of symbolic preprocessing accelerated convergence of the backpropagation learning process by a factor of 4 while improving accuracy on both the training and the test sets.

Whereas the preceding system reduces training time, WATTS (Wastewater Treatment System) (Krovvidy and Wee 1992) attempts to reduce the problem-solving time taken by a Hopfield network to converge to an optimal solution. WATTS' application task is to determine a treatment train, i.e., a sequence of processes aimed at eliminating wastewater contaminants before discharge to the environment. Since each contaminant can be eliminated by a variety of treatment processes and technologies, selection of the optimal treatment train requires combinatorial search. One approach adopted in WATTS is the use of a Hopfield

network whose outputs can be decoded into a solution (a treatment train). The network uses an energy function which was derived taking into account expert knowledge about interactions among these different treatment processes. In the simple NN approach, a random initial solution was generated and used to search for an optimal solution. In the hybrid CBR/NN approach, a case-based reasoner maintains a base of previous solutions and retrieves a relevant solution given a new problem. This solution is then used to initialize the Hopfield network in lieu of a random initial state. In general, the CBR/NN approach was found to improve performance both in terms of convergence time and—in a few cases—the quality of the solution.

Subprocessing

In subprocessing, one of the two components is embedded in and subordinated to the other, which acts as the main problem solver. Typically, the S component is the main processor and the N component the subprocessor. It is an open question whether the reverse setup is at all possible.

Neural subprocessing is used in two distinct conditions. In the first case, the symbolic main processor delegates to neural networks certain phases or subtasks of the application task that it presumably cannot do or does less well than NNs. This is illustrated in INNATE/QUALMS: the main processor, an expert system for fault diagnosis, calls on a set of multilayered perceptrons to generate a candidate fault, then either confirms their diagnosis or offers an alternative solution (Becraft et al. 1991). Another example is LAM™, a system for window glazing design (Medsker 1994). Its principal role is to serve as a design assistant: relying mainly on rule-based modules, it classifies glass types, checks for errors or improbable glass designs, and helps a user take design decisions such as choosing the glass type and then determining such properties as glass strength, solar specifications or sound control class. Design rules for properties such as structural strength of glass were easily constructed from architectural manufacturer specifications; however, two design tasks—selecting solar control and sound control properties of glass—are difficult to express in the form of situation/action rules since they involve estimating complex correlations between input and output parameters. At the same time, there exists a sizeable set of training/test data which accurately correlates known inputs with output predictions. Two three-layer feedforward networks were therefore trained to determine the appropriate solar and sound properties on the basis of other glass properties; during an interactive consultation, these two specific tasks are subcontracted to the neural network modules while a knowledge base of 578 *if-then* rules takes charge of all the other design phases.

The second case is where the symbolic processor calls on neural networks to perform specific internal functions, independently of the application task at hand. In Giambasi et al.'s (1989) system, for example, the connectionist component is not called explicitly by domain-level rules; it comes into action as an automatic subroutine of rule interpretation. When a rule is selected for execution, its associated neural net is activated to compute uncertainty factors for each of the facts added by the rule. Neural nets play a similar role in a hybrid system built on the SETHEO (Letz 1992) theorem prover. The system's default depth-first search strategy can be inefficient due to combinatorial explosion, thus the need for a heuristic means of detecting the most promising direction at each choice point. The prover therefore calls on a backpropagation neural network to estimate the probability that a candidate branch will lead to a proof. The input of the network is a set of features concerning the current formula (e.g., the number of literals or variables) as well as current state of the proof (e.g., current inference depth). Training data are pairs of feature vectors and desired estimations obtained from theorems proven by SETHEO using exhaustive search. The theorem prover selects the branch to follow on the basis of estimation results returned by the connectionist component. Tests have shown that the learned evaluation function can decrease search time by an order of magnitude in certain cases (Suttner and Ertel 1990).

Metaprocessing

Contrary to the widespread belief that metalevel capabilities are a prerogative of symbolic systems, both symbolic and connectionist modules can assume a metalevel role in hybrid systems. As in pure symbolic systems, a metalevel architecture is said to exist only if metaknowledge is explicitly represented (i.e., expressed in the representation language of the symbolic or neural component and not in their underlying implementation languages). According to this definition, a neural network can be considered a metaprocessor if it represents both domain and metaknowledge and if its output plays a metalevel role in the combined system. We thus distinguish between symbolic and neural metaprocessing systems, depending on whether the metaprocessor is symbolic or connectionist.

Symbolic metaprocessing is illustrated in ALVINN, Carnegie-Mellon's system for guiding autonomous vehicles (Pomerleau et al. 1991). Multiple networks are trained to become experts in specialized aspects of the autonomous vehicle control task (e.g., single-lane road driving, highway driving, collision avoidance). Once these networks are trained, their exploitation in real-world situations requires the ability to integrate their responses in order to ensure effective control

in a variety of circumstances. Furthermore, a truly autonomous system needs to be capable of planning its itinerary to reach a goal. ALVINN uses a rule-based arbitrator for this task: the decisions of the driving neural nets are sent to the arbitrator, which decides which network to follow and therefore how to steer. To take a decision, the arbitrator relies on an annotated map which stores geometric information such as the location of roads and landmarks, what type of road the vehicle is in, whether there is a dangerous permanent obstacle ahead, and so on. The map also contains control information relevant to the current driving situation; for instance, a point where a road changes from one lane to two is indicated on the map so the arbitrator knows when to start following the decision of the two-lane driving neural network.

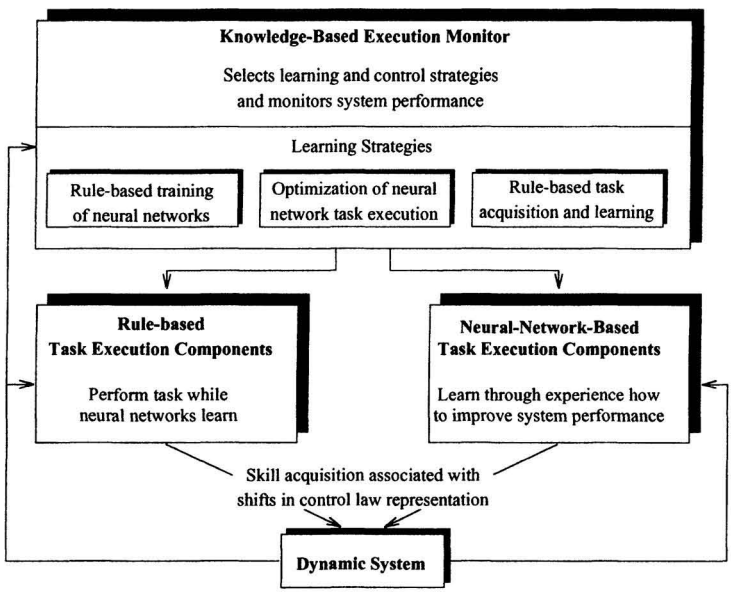

Figure 4 Symbolic metaprocessing for Robotic Skill Acquisition.

In the Robotic Skill Acquisition Architecture (RSA^2) (Handelman et al. 1989, Handelman et al. 1992), the symbolic metaprocessor supervises both symbolic and neural baselevel components. The system's goal is to develop robots which perform complex tasks using designer-supplied instructions and self-induced practice. The approach is patterned after human motor learning, which shifts from an explicit to an implicit representation and from controlled, verbally oriented to automatic, reflexive execution. At the base level, a rule-based system provides a declarative representation of human expert knowledge, whereas neu-

ral networks embody reflexive procedural knowledge that comes with practice. The symbolic metaprocessor is a rule-based execution monitor which supervises the training of the neural network and controls the operation of the system during the learning process.

Robotic skill acquisition can be divided into three distinct phases (Figure 4). During the declarative phase, the system executes a given task using explicit instructions stored in the knowledge base. During the hybrid phase, neural network components observe and try to duplicate rule-based maneuver commands, thus learning by example. Since initial net performance is poor, the knowledge base continues to ensure task execution; however, as the networks develop robust patterns of learned behavior, task execution is increasingly shared by the symbolic and connectionist components. Finally, during the reflexive phase, network-based control is optimized via reinforcement learning. Transitions between these three phases are managed by rules of the execution monitor.

In instances of *neural metaprocessing* that we are aware of, the connectionist component enforces search control over the symbolic baselevel processor. One example is a system which solves high school physics problems (Gutknecht and Pfeifer 1990). Here, object-level rules encode kinematics equations: for instance, $v = v0 + at$ is mapped roughly into the rule: "If the goal variable is v (final-velocity) and the known variables are $v0$ (initial velocity), a (acceleration) and t (time), then the final velocity is known." However, if several of the needed variables are unknown, the rules say nothing about a crucial control problem: which subgoal variable to solve for next. This is the task of the metalevel connectionist module. A neural network, previously trained by backpropagation, receives as input the goal variable and the known variables at a given stage; it outputs the next variable to solve for in view of finding the value of the goal variable. The sequence of subgoal variables selected successively by the network in effect serves as a control plan for the problem-solving process.

Another example of neural metaprocessing is Kwasny and Faisal's (1992) natural language parser. Classical parsing systems map an input sentence into a parse tree by maintaining a lookahead buffer and an internal stack. Tree-building actions (e.g., creation of a new node) are taken by a set of rules after examining both buffer and stack. The main drawback of these systems is that they can only parse sentences whose form has been anticipated in the ruleset; but natural language is too rich to be encoded in such a set. In the proposed hybrid parser, symbolic parsing rules are replaced by a feedforward network which has been trained on a set of sentences generated by a deterministic grammar. The symbolic module manages the buffer and stack and codes their state as inputs for the network, which determines the action to be taken on a best-match

basis. With this neural control component, the hybrid system is able to recognize grammatical as well as non-grammatical sentences, whether encountered previously or not.

Coprocessing

In coprocessing, the N and S components are equal partners in the problem-solving process: each can interact directly with environment, each can transmit information to and receive information from the other. They can compete under the supervision of a metaprocessor, or they can cooperate in various ways, e.g., by performing different subtasks, or by doing the same task in different ways and/or under different conditions. In SYNHESYS (Giacometti 1992), the same diagnostic task is executed by a rule-based system and a prototype-based neural network that learns incrementally. The neural component is tried first; if it comes up with a diagnostic, this output is validated by the rulebase in backward chaining mode; otherwise, the rulebase is activated in forward chaining mode and its diagnostic is used to train the neural network.

An example of the alternative setup—cooperative coprocessing by distribution of specialized subtasks—is a system where a neural network and a decision tree work together to detect arrythmia in heart patients (Jabri et al. 1992). Intra-cardial defibrillators (ICDs) are devices implanted in people with heart disorders: they sense the electrical activity of the heart and identify abnormal rhythms or "arrhythmias." These dysfunctions can be clustered into three main groups depending on the type of action they call for: continue monitoring, pace the heart, or apply high-voltage electric shock. Classification accuracy is, of course, crucial in this application. For so-called dual chamber classification (based on ventricular and atrial sensing), three architectures were tested: a single large multilayer perceptron (MLP), a multmodule NN consisting of three MLPs and two gates, and a hybrid decision tree/MLP model. In the hybrid model, the input data—sample signals sent by the ventricular and atrial probes—are sent to a decision tree which acts as a timing classifier and to a multilayer perceptron which performs morphology-based classification. The outputs of both modules are fed into an arbitrator which determines the class of the arrhythmia; this classification is then smoothed out by an "X out of Y" classifier to yield an "averaged" final classification. The decision tree/MLP hybrid attained an accuracy rate of 99% on a multi-patient database, whereas the best performance of the multimodule neural net was 96.2% and that of the simple MLP 89.3%.