1

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010.
(ANSI and ISO TEST CHART No. 2)

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC, 1970, c. C-30, et ses amendements subséquents.

**A FUNCTIONAL ARCHITECTURE
TO SUPPORT KNOWLEDGE SHARING
AMONG KNOWLEDGE BASED SYSTEMS**

by

Oleh Kandyba, B.Sc. (Eng)

**A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Engineering**

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario

June 1989

© copyright

1989, Oleh Kandyba

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-59042-4

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis,

**"A FUNCTIONAL ARCHITECTURE TO SUPPORT KNOWLEDGE
SHARING AMONG KNOWLEDGE BASED SYSTEMS"**

submitted by

Oleh Kandyba, B.Sc. (Eng)

in partial fulfillment of the requirements
for the degree of Master of Engineering.

Chair, Department of Systems
and Computer Engineering

Thesis Supervisor

Carleton University

December 27, 1989

ABSTRACT

This thesis defines a functional architecture for a new type of Problem Solving Network (PSN) which we have called a Consulting Network (CN). The prime objective of Consulting Networks is to facilitate the sharing of knowledge between intelligent agents, thereby providing users with easy access to judgement or advisory services on a wide variety of subjects.

Consulting Networks are based on hierarchical problem solving plus the idea of limited co-operation. We have defined suitable mechanisms to bridge agent heterogeneity, solve a broad range of general problems, work on multiple problems simultaneously, and to interact with users during the problem solving process. All of these features are implementable without unusual system complexity, albeit at the expense of some extra processing. We believe that this penalty is acceptable because the ability to functionally fuse otherwise isolated pools of knowledge for purposes of general problem solving carries an intrinsically high value to society.

ACKNOWLEDGEMENTS

I would like to express my thanks to my thesis supervisor, Dr. Morteza Nik-tash for his continuous support, guidance and encouragement during the exploration of this very new and exciting field. His patience and willingness to discuss the numerous technical alternatives were instrumental in helping to coalesce the sparks of ideas into a thesis.

Second, I am greatly indebted to Dr. Archibald Bowen for taking the trouble to review and understand this thesis despite extremely heavy personal commitments, and make numerous valuable suggestions which led to its completion.

I would also like to thank LCol. Barry Brown, P.Eng., for also reviewing the contents of this thesis and for helping to keep in focus the objective that the ideas presented herein must be usable in the real world.

The greatest thanks of all goes to my wife Tamara, who stood by me throughout the years of studying, never questioned my commitment and gave me the strong spiritual support which I needed to reach my goal. This Master of Engineering degree is as much hers as it is mine.

TABLE OF CONTENTS

	PAGE
Acceptance Sheet	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Figures	ix
Glossary of Acronyms	x

CHAPTER 1:

INTRODUCTION	1-1
--------------------	-----

CHAPTER 2:

CURRENT APPROACHES TO DISTRIBUTED PROBLEM SOLVING	2-1
2.1 KEY ISSUES IN THE DESIGN OF PSNs	2-2
2.2 AGENT INTERACTION MODE	2-4
2.3 MECHANISMS FOR AGENT INTERACTION	2-6
2.3.1 Centralized Planning & Problem Solving	2-7
2.3.2 Approaches To Distributed Planning & Problem Solving	2-9
2.3.2.1 Multi-Agent Planning	2-10
2.3.2.2 Negotiation	2-11
2.3.2.3 Functionally Accurate/Co-Operative Approach	2-13
2.3.3 Communications Policies & Network Coherence	2-14
2.3.3.1 Send-All	2-17
2.3.3.2 Locally-Complete	2-17
2.3.3.3 First-and-Last	2-18
2.3.4 Hard-Wired Interaction	2-19

	PAGE
2.4 SHORTCOMINGS OF CURRENT APPROACHES	2-20
2.4.1 Nodal Resource Penalties	2-20
2.4.2 The Problem of Heterogeneous Agents	2-21
2.4.3 Overspecialized Problem Solving Capabilities	2-22
2.4.4 Interaction With Users	2-22

CHAPTER 3:

THE BUILDING BLOCKS OF CONSULTING NETWORKS	3-1
3.1 FUNCTIONAL EXTENSIONS TO THE PSN CONCEPT	3-2
3.2 SYSTEM CONSIDERATIONS	3-6
3.2.1 Agent Connectivity	3-6
3.2.2 Problem-Solving Agent Requirements	3-8
3.2.3 Man-Machine Interaction	3-9
3.2.4 Bridging Heterogeneity	3-13
3.2.4.1 Disparate Computing Environments	3-13
3.2.4.2 Disparate Specializations	3-14
3.2.4.3 Disparate Knowledge Representation	3-15
3.2.4.4 Linguistic Issues in Distributed Problem Solving ...	3-16
3.2.4.5 KDL: The KBS Discourse Language	3-18

CHAPTER 4:

CONSULTING NETWORK PROBLEM SOLVING PROCESS	4-1
4.1 INTRODUCTION	4-2
4.2 PROBLEM DECOMPOSITION	4-3
4.2.1 Problem Solving Through Limited Co-Operation	4-4
4.3 SUB-PROBLEM DISTRIBUTION	4-6
4.3.1 Determination of Knowledge Domain Limits	4-7
4.3.2 Representation of Organizational Knowledge	4-8
4.3.3 Self-Knowledge and Agent Capability Assessment	4-13

	PAGE
4.3.4 Selection of an Optimum Task Target	4-14
4.3.4.1 Task Assignment Through Network Organizational Knowledge	4-14
4.3.4.2 Task Assignment Through Self-knowledge	4-19
4.3.5 Relationship Between the KDL and the Task Distribution Process	4-21
4.3.6 Generation of Query Topical Chains (QTCs)	4-21
4.4 SUB-PROBLEM SOLUTION	4-25
4.5 ANSWER SYNTHESIS	4-26

CHAPTER 5:

ANALYSIS OF THE LIL DERIVATION PROCESS	5-1
5.1 INTRODUCTION	5-2
5.2 THE TSN SEARCH AND SCORING ALGORITHMS	5-3
5.2.1 Effects of Topical Classification	5-3
5.2.2 Effects of Knowledge Granularity	5-5
5.2.3 Proposed TSN Search Algorithm	5-7
5.2.4 Proposed Scoring Algorithm	5-9
5.2.5 Performance of the TSN Search Process	5-11
5.3 ANALYSIS OF TSN SEARCH STRATEGIES	5-12
5.3.1 The Network Organizational Knowledge Approach	5-13
5.3.2 The Self-Knowledge Approach	5-15
5.3.3 Discussion of Relative Merits	5-18

CHAPTER 6:

A FUNCTIONAL MODEL OF CONSULTING NETWORKS

- A FRAMEWORK FOR KNOWLEDGE DOMAIN FUSION	6-1
6.1 INTRODUCTION	6-2
6.2 NETWORK FUNCTIONAL DECOMPOSITION	6-4

	PAGE
6.2.1 The System	6-4
6.2.2 Level 0 - The Consulting Network Function	6-6
6.2.3 Level B - The Problem Solving Agent Function	6-10
6.2.4 Level BB - The Manage Problem Solving Node Function	6-13
6.2.5 Level BD - The Solve Problem Function	6-16
6.2.6 Level BDD - The Solve Locally Unsolvable Sub-Problems Function	6-21
6.3 FUNCTIONAL DECOMPOSITION SUMMARY	6-24
 CHAPTER 7:	
THESIS SUMMARY	7-1
7.1 CONTRIBUTIONS TO DAI RESEARCH	7-2
7.2 LIMITATIONS OF CONSULTING NETWORKS	7-6
7.3 CONCLUSIONS	7-8
7.4 SUGGESTIONS FOR FURTHER RESEARCH	7-11
 APPENDIX A: LISTINGS OF THE TSN SEARCH & SCORE ALGORITHM	
	A-1
 REFERENCES	 R-1

LIST OF FIGURES

FIG.	DESCRIPTION	PAGE
3-1	Conceptual KDL Translation Process	3-22
4-1	Storage of Topical Data as a Tree Structure	4-11
4-2	LIL Derivation and the Search for a Task Target	4-16
4-3	Role of the KDL in Task Vectoring	4-22
4-4	Formation of QTCs From Task Descriptions	4-24
4-5	The Problem Solving Hierarchy and the Tracking of Sub-Tasks	4-27
5-1	TSN Search Algorithm Structure Chart	5-8
5-2	Network Event Timing Diagram for RFC Message Emission	5-16
5-3	LIL Derivation Time vs Number of Network Agents	5-19
5-4	LIL Derivation Load vs Number of Network Agents	5-19
5-5	Total Computation Load vs Number of Network Agents	5-20
5-6	LIL Derivation Time vs Load	5-21
6-1	The System Level	6-5
6-2	Level 0 - The Consulting Network Function	6-7
6-3	Level B - The Problem Solving Agent (PSA) Function	6-11
6-4	Level BB - The Manage Problem Solving Node (MPSN) Function ...	6-14
6-5	MPSN Function State Transition Diagram	6-17
6-6	Level BD - The Solve Problem (SP) Function	6-18
6-7	Level BDD - The Solve Locally Unsolvable Sub-Problems (SLUSP) Function	6-22

GLOSSARY OF ACRONYMS

ASPT	Awaken SP Task (Function)
CP	Create Plan (Function)
CPU	Central Processing Unit
CS	Combine Solutions (Function)
DAC	Determine Agent Capability (Function)
DAI	Distributed Artificial Intelligence
DBMS	Database Management System
DKT	Domain - KDL Translation (Function)
DRM	Decode & Route Message (Function)
ES	Expert System
FA/C	Functionally Accurate/Co-Operative
GRM	Generate RFC Message (Function)
ISPT	Instantiate SP Task (Function)
KB	Knowledge Base
KBS	Knowledge Based System
KDL	Knowledge based system Discourse Language
KDTMR	KDL - Domain Translation & Message Routing (Function)
KS	Knowledge Source
LAN	Local Area Network
LIL	Likelihood List

MPSN	Manage Problem Solving Node (Function)
MSA	Manage Subordinate Agents (Function)
NAN	Network Access Node
NLI	Natural Language Interface
NLP	Natural Language Processing
NLU	Natural Language Understanding
OSI	Open Systems Interconnection
PL	Produce LIL (Function)
PSAN	Problem Solving Agent Node
PSN	Problem Solving Network
QTC	Query Topical Chain
RFC	Request For Capability
RIM	Receive Incoming Messages (Function)
SADT	Structured Analysis Design Technique
SLSSP	Solve Locally Solvable Sub-Problems (Function)
SLUSP	Solve Locally Unsolvable Sub-Problems (Function)
SOM	Send Outgoing Messages (Function)
SP	Solve Problem (Function)
SSPT	Suspend SP Task (Function)
TSN	Topically Oriented Semantic Net
TSPT	Terminate SP Task (Function)

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

One of the most important attributes of the human species is its ability and willingness to work in groups for the achievement of common goals. Many group problems are often of such complexity that they transcend the capabilities of a single human being to solve. Therefore, in order to deal with them, it becomes necessary to bring to bear the collective body of knowledge and experience of the group. Implicit in this approach is the co-operative sharing of knowledge among group members. Often the outcome of such problem solving activities far exceeds the reasoning effectiveness of any one individual. Perhaps because of their focused intellectual power, group problem solving teams have often been called "think tanks" and have been tasked with studying some of the most complex issues facing mankind.

The maturing of AI technologies over the last decade has led to the development and proliferation of Knowledge Based Systems (KBSs) which contain knowledge of specific domains. These KBSs can provide access to expert level judgement in their respective areas of expertise. KBSs are usually stand-alone systems which are isolated from other KBSs. When a given problem is beyond their capabilities, they often exhibit brittle behaviour and the quality of their rendered judgments deteriorates quickly. Recognising the intellectual synergy of humans teams working on common problems, it is extremely attractive to be able to duplicate the same type of process among distributed intelligent machines such as KBSs. The benefits of distributed problem solving are well recognized [Durfee 87a][Huhns 87a]. There is, however, an

important obstacle. Whereas, co-operating humans have an understanding of language and of group norms which provide a framework for interactions among them, the current generation of KBSs have not been designed to co-operate with other KBSs in solving joint problems. They have little or no framework for co-operative interaction and can co-operate only in very limited ways [Durfee 87a].

There is, however, a rising interest among AI researchers to study the phenomenology of distributed problem solving [Davis 83] [Retelle 86] [Sridharan 87] [Durfee 87a] [Huhns 87a] [Shastri 87b]. Much of the research centers around Problem Solving Networks (PSNs) which is an architecture of distributed, loosely coupled, intelligent problem solver agents [Davis 83]. Many of the agent interaction processes developed to-date have been evolved in the context of PSN testbeds by the researchers mentioned above. In their present form, PSNs are composed of homogeneous agents, and are being used to solve single, carefully selected problems. They are currently unsuitable for general problem solving.

The objective of this thesis is to establish a functional system architecture and to define some of the fundamental underlying processes needed to support a more general and powerful mode of co-operative interaction between KBSs leading to general problem solving capabilities. We realize that this entails applying several technologies which are the subject of considerable investigation among AI researchers. In order to accommodate a solution to our problem, we have made several simplifying assumptions which are briefly discussed below.

The issue of knowledge portability is central to the co-operative problem solving process. In general, this is difficult because the formalisms used to codify knowledge are often specific to the knowledge type and to the implementation of

agents. Not all formalisms may be supported by all agents, which requires the remapping knowledge sources between representation forms. It may not be possible to correctly translate all arbitrary concepts or knowledge sources. However, we have assumed that knowledge sharing can be made to function not through the exchange of knowledge sources, but through the exchange of facts and findings, thus avoiding the problem.

Implicit in knowledge sharing is an understanding of the semantics of the knowledge. This raises some very important linguistic issues which appear on two distinct levels in a system of co-operating agents. The power and expressiveness of the man-machine interface is crucial to ensure that the system will solve the required problem. Equally important is the need for the user to interact with the system at run-time and to participate in key segments of the problem solving process. Both of these processes require Natural Language Processing (NLP) and Natural Language Understanding (NLU) which are particularly difficult areas of research because of the syntactic complexities, context dependencies, and semantic ambiguities found in natural languages. We have assumed that NLP and NLU currently are, or soon will be sufficiently mature, to allow a useful level of general problem solving. Difficult linguistic problems also exist on the level of inter-agent interfaces. Co-operative reasoning processes will entail the exchange among agents of various concepts such as problem statements, constraints, facts and findings, etc. This raises the requirement for a highly expressive, yet unambiguous common language of discourse through which this interaction can be accomplished. The design of such an artificial language poses a problem of the same order of difficulty as the design of a natural language. On the basis

of recent work, we are hopeful that a suitable language can indeed be developed [Jakobson 86][Hillenbrand 89].

Planning for a distributed problem solving environment is another difficult question which must be solved in order to achieve coherence in the network of co-operative agents. Robust universal planning algorithms for general problem solving in broad knowledge domains have not yet been found. However, general planning within narrow domains has been shown to be viable [Hayes-Roth 83][Waterman 86]. We have assumed that hierarchical reduction of problems will allow co-operating agents to progressively decompose a general problem statement to the point where specific problems in narrow domains can be identified and solved. A second related assumption presupposes that a sufficiently wide variety of human problems are hierarchically structured, justifying the hierarchical problem reduction approach in the first place.

It is recognized that non-monotonic reasoning may be required to solve some problems. We suggest that functional non-monotonicity in a network of agents should appear on two levels. Individual agents may certainly employ local non-monotonic reasoning as required to derive solutions to local problems. However, on the network level, support for non-monotonic reasoning spanning multiple agents is a difficult problem which has not been specifically addressed in this thesis. It requires specialized knowledge sources residing on several layers [Wielinga 86] of agents' knowledge bases. These will ensure that specific sub-problems are reworked because of changes occurring elsewhere in the problem solving process.

In Chapter 2, focusing on the work done in PSNs, we review the current approaches to distributed problem solving and mechanisms for agent interaction. We review several progressively more complex approaches to network planning. These at-

tempt to circumvent the fundamental conflict of distributed problem solving, namely that a complete knowledge of network events is needed to ensure complete coherence among agents, but incomplete knowledge is a characteristic of partitioning the problem solving process [Davis 83]. We describe how network coherence can be improved through appropriate communications policies which provide agents with predictive information about network events. As an example of a highly efficient, but inflexible interaction mode we examine Takenouchi's hard-wired approach [Takenouchi 87]. Finally we show that if our objective is general problem solving, then current approaches such as PSNs are incapable of achieving this because of overspecialization, nodal resource penalties caused by very complex software processes, inability to bridge agent heterogeneity and to interact with users during the problem solving process.

In Chapter 3 we propose to change the focus of distributed problem solving from specialization to general problem solving. We propose functional extensions to PSNs which are intended to rectify the problem areas identified above. To distinguish general problem solving networks from the current generation of PSNs, we coined the name Consulting Network (CN), whose most important attribute is the cooperative sharing of knowledge among agents to solve user problems. We then examine several key high level system issues related to CNs. We conclude that to maximize agent connectivity and foster an open architecture of distributed knowledge sources requires an active interaction mode. We recognize that problem solving agents will require powerful computing engines, memory and disk storage resources which are appropriate to the size of the knowledge bases which they host. Because of the expressive power and flexibility of human language, it is highly desirable that the man-machine interface (MMI) with CNs be based on natural language. However, due to

the syntactic complexities, context dependencies and semantic ambiguities found in most natural languages, Natural Language Processing (NLP) and Natural Language Understanding (NLU) technologies are currently incapable of providing a sufficiently robust MMI. We also show that in a CN, problem solving must be session oriented for reasons of processor efficiency, to support the capability for follow-up questions from users, and for generating explanations on request.

We show that heterogeneity of knowledge representation is a fundamental and necessary characteristic of CNs. Heterogeneity also encompasses issues such as disparate computing environments and agent specializations. As a result, the sharing of knowledge among problem solving agents is predicated on the availability of suitable mechanisms for bridging this heterogeneity. To this end, we propose a KBS Discourse Language (KDL) to support problem solving by heterogeneous agents within a CN, and then discuss some of its key attributes. From this discussion we conclude that the design of an artificial language capable of correctly conveying the subtleties of user problems is an extremely difficult problem. Consequently, in our view, various linguistics issues constitute a major hurdle in the implementation of Consulting Networks.

Chapter 4 examines the four phases of the problem solving process: problem decomposition, sub-problem distribution, sub-problem solution and answer synthesis. We discuss various relevant implementation techniques applicable to each phase, and if appropriate, we propose modifications or completely new approaches. In discussing Problem Decomposition, we suggest that a hierarchical approach to planning corresponds to the structure of many human problems and should be used in Consulting Networks. We propose to distribute the planning process by allowing all Network agents to plan for themselves and for a limited number of subordinates. We

propose to simplify the interactions between Network agents by limiting their duration and scope. The result is a distributed hierarchical problem decomposition conforming to a multi-tiered client-server model. We show that this approach yields several important advantages over previous ones, such as decentralized planning and control processes, as well as flexible and simplified interactions between agents. For the Sub-Problem Distribution phase, we propose to apply topically oriented semantic nets (TSNs) to catalogue the domain metaknowledge of agents [de Haan 86], which is often referred to as "organizational knowledge" [Durfee 87a]. Given a list of sub-problem attributes, this enables agents to assess the capabilities of other agents in the Network and determine to whom to delegate reasoning tasks which they themselves cannot handle. We discuss two approaches for task assignment. The Network Organizational Knowledge (NOK) approach is based on agents knowing about the capabilities of all other agents in the Network, while the Self Knowledge (SK) approach is based on agents only having knowledge of their local capabilities. We show that several of the mechanisms needed for the distribution of reasoning tasks are dependent on the power and expressiveness of the KBS Discourse Language discussed earlier. Assuming the presence of an effective KDL, a technique is proposed for generating agent qualifications called Query Topical Chains (QTCs) directly from sub-problems descriptions. These will be pattern-matched against branches of the TSNs described above and used to compute the degree to which agents are qualified to perform a specific reasoning task. In discussing the Sub-Problem Solution phase, we discuss how the multi-tasking capabilities of agents allow them to efficiently interface with users and obtain supplementary information needed for reasoning, while not wasting processing cycles by waiting for the characteristically slow human responses. For the Answer Synthesis

phase we propose a very simple but effective technique for tracking sub-tasks throughout the Network belonging to multiple problem-solving sessions.

The speed and effectiveness of the task distribution process is seen to have a major effect on the performance and problem solving abilities of Consulting Networks. Therefore, in Chapter 5 we discuss issues related to the generation of lists of candidate agents. We first examine issues such as the effects of topical classification strategies used in the creation of TSNs and QTCs, and the effects of knowledge granularity on the ability of agents to identify suitable agents to task. We then present a TSN search algorithm written in Pascal on a PC/AT, which incorporates some key heuristics which shorten the candidate search process, and present estimates of the algorithm's performance.

We present the results of a formal mathematical analysis of the candidate agent identification strategies discussed in Chapter 4 the Network Organizational Knowledge (NOK) and Self-Knowledge (SK) approaches. It is recognized that this analysis is at a high level, but this is understandable considering the early state of development of the CN concept. We also model the performance of both strategies on Lotus 123 using representative parameter values. Although this modelling does not relate to any specific implementation of a CN, it allows one to get a qualitative appreciation of the relative performance of the two techniques. With the help of these modelling results, we see that the processing load on agents using the SK approach is virtually identical to the NOK approach. However, the SK approach generates lists of candidate agents in very much less time because of the parallel search procedure. In addition, the SK approach alleviates the problem of software configuration control across the

network and helps to maintain an open architecture of distributed knowledge sources. It is therefore judged to be superior to the NOK approach.

In Chapter 6 we present a functional decomposition of a Consulting Network. The description is based on Ross' graphics-oriented SADT methodology. We show how the various proposed processes and mechanisms interact in the simultaneous solution of multiple user problems. The decomposition is based on the assumption that all of the proposed mechanisms described above can be made to work. We conclude that the functionality is coherent, complete and comprises a good framework for the planning and integration of future research efforts.

Finally in Chapter 7 we summarize the findings of this thesis, describing the contributions to distributed AI research. We discuss some limitations of the Consulting Network concept in its present form. We conclude that CNs are potentially a very rich area of research with many areas requiring further investigation. Perhaps one of the most difficult problems is that of Natural Language Processing and Natural Language Understanding as it relates to the areas of the man-machine interface and in the exchange of knowledge between co-operating heterogeneous agents. However, we believe that the promise of Consulting Networks is such, that further investigations into the concept are warranted.

CHAPTER 2

CURRENT APPROACHES TO DISTRIBUTED PROBLEM SOLVING

2.1 KEY ISSUES IN THE DESIGN OF PSNs

The design of PSNs should strive for maximum network throughput and highly efficient use of network resources such as communications bandwidth, processing and storage. To achieve these objectives, network agents must work coherently. Coherence means that network agents must not work at cross-purposes to each other and that their actions must be supportive of each other in achieving the network goal [Durfee 87a]. Global coherence means that the activities of the nodes must be relevant to the achievement of the overall network goals. Nodes should avoid unnecessary duplication of the work of other nodes, remaining idle when other parts of the network are overloaded, and transmitting irrelevant or distracting information to other nodes [Durfee 87b]. That is to say, network agents must co-operate with each other in problem solving. Because of its importance, global coherence has been called the holy grail of distributed problem solving research [Durfee 85].

Full global coherence is a difficult and elusive goal. According to Davis & Smith, a fundamental conflict exists between the complete knowledge needed to ensure network coherence and the incomplete knowledge which is a characteristic of distributed problem solving [Davis 83]. Full global coherence can only be guaranteed if each agent has a complete and accurate perception of problem solving activities across the network as well as the intentions of all network agents. Because all real-world communication channels have limited bandwidth, are subject to errors and transmission delays, complete global coherence cannot be achieved [Durfee 87b]. Therefore, PSNs

require specific interaction mechanisms to resolve these shortcomings. All that one can hope for at present is a compromise yielding maximized network performance and resource usage efficiency.

The execution of a well-conceived network plan is the most important mechanism to enhance network coherence. Network plans primarily attempt to optimally assign tasks to processors in order to achieve best use of computing resources in solving any given problem. Network plans also give intelligent agents an insight into what interactions may take place among agents, so that they can potentially adjust their own local plans to better achieve the PSN's objectives. However, Durfee et al have shown that knowledge of what interactions will take place is insufficient. The time when these interactions will occur is also a very important factor because agents also need to know when to expect them so that they can modify their local plans accordingly. Therefore, coherent co-operation requires that the interaction mechanisms must also provide predictive information to network agents. This implies a continuous exchange of meta-level information among agents while they are engaged in problem solving [Durfee 85][Durfee 87a].

Durfee et al found that interactions among agents need to be mediated through an appropriate communications policy so that the information exchanged is relevant, timely and complete. Otherwise, communications may become a source of distraction and counterproductive to agents; consuming computing resources, communications bandwidth, reduce performance and even ultimately prevent network coherence [Durfee 87a]. In the next section we will discuss specific approaches to implementing co-operative interaction among PSN agents.

2.2 AGENT INTERACTION MODE

The mode of interaction is one of the key system decisions that must be made for distributed problem solving because it establishes the mechanisms and affects the format for the exchange of information among agents. In the past, distributed KBSs interacted according to one of three interaction modes:

Active interaction wherein all agents communicate through message passing between relevant agents. The work of Durfee et al is a prime example of this approach [Durfee 87a]. Because communicating agents have the flexibility of sending messages to specific single agents, groups of agents, or all agents in the network, its most important advantage is that in large scale networks it can conserve communication bandwidth and reduce communication delays. This leads to an improvement in the timeliness of the transmitted information. Moreover, it supports the goal of an open system architecture

Passive interaction in distributed problem solving typically consists of establishing and maintaining multiple copies of a blackboard. Blackboards are memory segments dedicated to the storage of hypotheses, decisions, and other information to be manipulated by several processes [Hayes-Roth 83]. In blackboard systems, all information is simply posted in predefined locations in the blackboard data structure, from which processes requiring specific data items may simply read them. In blackboard based interaction, agents wishing to communicate will transmit blackboard updates to all other agents. Harmon's autonomous Ground Surveillance Robot (GSR) vehicle has demonstrated at least for small scale systems the viability of this approach [Harmon 86][Harmon 87a][Harmon 87b]. Since the same distributed blackboard must

satisfy the information requirements of all network agents, only a fraction of the information is relevant to any given agent. Since it may not be possible to determine in advance which information will be relevant, all of the information stored in each copy of the blackboard must be kept current. Clearly, in large scale networks, this approach can generate a great deal of unnecessary communication traffic.

Blackboards have also been used to establish point-to-point interfaces between pairs of problem solvers in small scale hierarchical systems. In Goldberg's approach, blackboards resided in segments of dual-port shared memory accessed by the relevant agents [Goldberg 85]. This technique is quite efficient in that the interfacing blackboards contain only that information which is relevant to the given agents. However, its applicability is limited to small scale systems because of the combinatorial explosion of memories as the scale of the system grows.

Indirect interaction occurs when agents observe their surroundings and the actions of neighbouring agents. No communications of any kind take place between them. Knowing their own goals or their mission, they somehow infer an optimum course of action in order to solve the network problem. Genesereth postulates that such systems could be applicable to autonomous military robotic vehicles such as tanks operating under Emission Control (EMCON) rules during a combat mission [Genesereth 86].

Indirect interaction is probably the most difficult type of interaction to implement because of uncertainties inherent in the interpretation of observations. These may reduce the ability of an agent to react quickly because corroborating evidence must be built up over a period of time to raise the system's confidence in its own hypotheses to the point where it can take action. Clearly, the lack of suitable sensors to drive the

process as well as the intrinsic temporal latency make us reject indirect interaction for use in distributed general problem solving. Therefore, we conclude that active interaction is the most flexible and efficient mode.

2.3 MECHANISMS FOR AGENT INTERACTION

The chief difference between stand-alone and co-operating agents is that in addition to knowledge for purposes of stand-alone problem solving, the knowledge domains of co-operating agents contain extensions which guide them in distributed problem solving and govern their interactions with other agents in the PSN. That is to say, they contain strategies for co-operative interaction. We note that these strategies span all four phases of the PSN problem solving process as discussed in Section 1.4.

A PSN designer attempting to implement co-operation mechanisms, must recognize that they must be appropriate to the type of problems to be solved by the network, to the characteristics of the individual agents and to the benefits which are desired from co-operation. We will now examine the two central issues in PSNs:

- 1) Approaches to planning and distributed problem solving which strongly influence problem decomposition, sub-problem distribution and answer synthesis;
- 2) Effects of inter-agent communication policies which strongly affect the efficiency of the solution of sub-problems.

2.3.1 CENTRALIZED PLANNING & PROBLEM SOLVING

A plan represents a desire to achieve a high level goal by performing a series of structured actions or tasks [Durfee 87a]. Since the early 1970's there was a great deal of interest within the AI community to develop methods for general planning and problem solving. Research initially focussed on games in order to get a basic understanding of the process, but the end objective remained the much more ambitious and serious applications associated with machine intelligence [Fikes 71] [Fikes 72] [Nilsson 80] [Hayes-Roth 83]. It was soon recognized that generalized planning and problem solving was a much more difficult task than had been originally envisaged. Certainly, for various restricted domains such as tic-tac-toe, chess as well as numerous other applications, good specialized planning methods have been found, but a robust universal approach has not yet evolved.

All KBSs require a planner function to formulate and oversee the solution of problems. A planner is defined as a program which controls one or more devices capable of carrying out actions in the real world in order to achieve some definite purpose [Swartout 88]. In general, planners fall into two broad classes: strategic planners and tactical planners. Strategic planners are capable of anticipating or predicting some of the consequences of its actions and choosing between various courses of action. Tactical planners, on the other hand, are concerned with what to do immediately in situations given uncertain or limited information. Because problem domains represent a continuum of characteristics, most sophisticated planner modules contain features of both types.

Depending on the specifics of the application, KBS builders currently use one of three main planning methods [Swartout 88]:

Linear Planning Plans are comprised of linear sequences of actions. Actions consist of mappings of sets of sentences to sets of sentences expressed in some appropriate language. The technique offers simplicity and rapid computation. However, the choice of language limits the expressivity of the plan. Also, some domains may not be represented, somewhat restricting its applicability. Despite this, linear planning has been implemented in Newell and Simon's General Problem Solver [Ernst 69] as well as Fikes and Nilsson's STRIPS experiments [Fikes 71].

Hierarchical Planning A plan consists of a hierarchy of tasks and sub-tasks. Tasks in turn consist of a series of inferences performed with appropriate knowledge sources. We note that this planning model matches almost exactly the Wielinga-Breuker layered model of structured expertise [Wielinga 86]. Hierarchical planning offers much more flexibility in ordering the commitment of tasks than does linear planning. In many KBSs where there is a natural structural hierarchy evident in the problem domain, such as in the design of standard structures, ICs, etc., hierarchical planning offers a very convenient approach [McDermott 77] [Verrilli 87].

Case Based Planning evolved from the observation that in human reasoning, many plans are simply modifications to plans which have worked in the past. Hence the development of new plans depends on the organization and storage of past experience. In case based planning, procedures are required for the debugging and refinement of old plans. Because each plan is an iteration of an existing one, case based planning for the most part, works considerably faster than the previous methods described above [Hammond 86].

Irrespective of the approach to planning, the control of reasoning within early stand-alone KBS was based on knowledge sources so that the highest rated knowledge source pending on their agenda was always executed first. This was fundamentally an opportunistic approach to problem solving which exploited opportunities to make the greatest possible gains in developing lines of reasoning. However, when applied to distributed problem solving, it suffered from having a too limited view of the global problem and could lead to blind alleys and wasted resources [Durfee 87a].

In plan-based agents, opportunism still drives the problem solving strategy, but it is always the highest rated plan which is executed next; not the highest rated knowledge source [Durfee 87a]. Plan-based systems must, of course, possess a mechanism for rating potential plans and choosing the most promising of those available. Working with plans, a KBS will not be as easily distracted by interesting new knowledge sources which may suddenly come to its attention. This is particularly relevant to PSN agents who can be affected by external messages. We note that in a monolithic KBS the planning function is greatly simplified in that it does not have to take into consideration the problem solving activities of any other agents. In this case, coherence is guaranteed because the planner has a complete view of how the problem will be solved.

2.3.2 APPROACHES TO PLANNING & DISTRIBUTED PROBLEM SOLVING

Davis and Smith have defined distributed problem solving as being a co-operative activity of a group of decentralized and loosely coupled knowledge sources (KSs) [Davis 83]. To maximize coherence, distributed problem solving requires the formation of network-wide strategic plans which will co-ordinate the actions of all net-

work agents. The process of plan creation reconciles the global network view of the problem with existing long, medium and short term agent plans. This allows an agent to make more intelligent decisions about the direction of its own local reasoning [Durfee 87a]. An important consideration in the development of a network plan is that the modularity and grain size of task decomposition must be sufficiently coarse, so that the overheads of distributed processing do not outweigh the benefits [Davis 83].

We will now examine three progressively more complex approaches to problem solving and consider how the network plans can be created, what is the basis for assigning tasks to agents, and how the execution of plans is controlled within the context of individual computing nodes. We note that these approaches are not mutually exclusive and in practice a PSN designer may well implement a composite problem solving approach.

2.3.2.1 Multi-Agent Planning

In the multi-agent planning approach, nodes choose a single node from among themselves and designate it as the network planner. As an alternative, the node which receives a problem from outside the PSN becomes the network planner. Nodes subsequently send all relevant information, such as their status and capabilities, to the network planner so that he can construct a global plan for the entire PSN and distribute detailed plans to each of the agent nodes. Driven by the arrival of information from other agents, each node is responsible for executing its portion of the network plan. It follows that the network planner is also responsible for merging the results of executed plans into a solution to the original problem.

The multi-agent planning approach has the advantage that the planning node has a global view of the problem and can in principle formulate plans which are coherent across the network. This implies that all important events and node interactions can be predicted and tightly synchronized to obtain optimal network performance. Conversely, situations with adverse consequences can be avoided. However, multi-agent planning has one major shortcoming in that it has a single point of failure. Should the hard-assigned planning node fail, then the entire network would be completely disabled.

As an alternative form of multi-agent planning, Durfee et al suggest that nodes could interactively create a joint plan through their knowledge of the capabilities of other nodes and a description of the network problem [Durfee 87a]. We see two drawbacks to this approach. Because no single agent has a complete view of the network plan, the agent plans would have to be evolved iteratively. This could be quite time consuming and costly in terms of network communication resources. Moreover, there is no guarantee that such plans would be both viable and coherent across the network.

2.3.2.2 Negotiation

In the negotiation approach each node which received a problem, decomposes it into a series of sub-tasks. The node must then determine whether it is capable of solving all of the sub-tasks either in the available time or at all. If not, then some or all of these sub-tasks must be assigned to other nodes in the PSN for execution in parallel.

The problem of assigning tasks to agents may be handled through a bidding scheme such as Davis & Smith's contract net protocol [Davis 83]. This approach recognizes that the network nodes may have differing capabilities. Whenever an agent cannot solve a problem, it broadcasts a description of the problem to all other agents in the PSN, soliciting requests for proposals from each. All agents who think that they are qualified, submit proposals to the originator agent based on their capabilities and on their workload status. It is up to the originator to enter into a dialogue with each responding agent and somehow verify whether they are indeed competent and what will the cost be of having the job done by the given respondent. The cost metric may consider a mix of appropriate factors such as the time required, the network resources, etc. Ultimately, the originator enters into a contract with the best responding node and assigns the job [Davis 83]. From then on, the receiving agent is fully responsible for carrying out the reasoning job and returning the results to the originator agent. The contract net protocol has been successfully applied in several application areas [Jarke 86] [Boettcher 87].

The coherence of a PSN depends largely on the appropriate assignment of sub-tasks to agents. Although in principle, sub-tasks should be given to the agent which is best qualified to perform them, blind assignment of sub-tasks to agents may preclude optimal sub-task assignment over the span of the whole problem. Without some form of prediction capabilities incorporating a global view of the problem, agents may accept any task given to them. As a consequence, they may not be available to accept a higher importance task for which they are uniquely qualified which follows soon after [Durfee 87a].

In the negotiation approach, we see potential problems in several areas. First, negotiating heterogeneous agents require some universal means of conferring about problems so that they can be understood by potential bidders. Otherwise its applicability is limited to homogeneous networks with a clearly established functionality and communication semantics. Second, negotiation processes require strategic knowledge to arrive at some agreement or at least to prevent impasses in the case of negotiation conflicts. Clearly, both of these areas require further research.

2.3.2.3 Functionally Accurate/Co-Operative Approach

In the Functionally Accurate/Co-Operative (FA/C) approach, agents co-operate through the generation and exchange of tentative, partial solutions which are based on local, incomplete views of the problem [Durfee 87a]. On the network level, problem solving takes place iteratively through the exchange and gradual filtering out of incomplete, inaccurate and inconsistent information. If all goes well, a consistent solution should emerge in the end. The FA/C approach has the advantage that all nodes participate in the planning process, and that there is no single point of failure in the network.

In the research done to-date by Durfee and Corkill, the FA/C approach has only been applied to small networks comprised of upto ten homogeneous agents working on a single problem [Durfee 84]. This strongly reduces the complexity of the planning and the frequency of interactions. It is not clear whether the FA/C method is applicable to larger networks and general problem solving without major modifications.

2.3.3 COMMUNICATION POLICIES & NETWORK COHERENCE

As mentioned earlier, Durfee et al found that to ensure optimal execution of agents' local plans, agents' knowledge of which interactions are going to take place needs to be supplemented with knowledge of when they will happen [Durfee 87a]. Therefore, in general, an event and information arrival prediction process is crucial for agents to be able to optimize their own plans in context of the network problem, thereby improving network coherence. Prediction implies that agents must somehow infer the plans of other agents in the network.

The above considerations are valid irrespective of the planning method chosen for the network. For example, the multi-agent planning approach requires accurate prediction of events before it can produce acceptable plans. It achieves this via the planner agent's global view of the problem and subsequent orchestration of all network activities. In contrast, although the negotiation and the FA/C approaches can operate after a fashion without adequate predictions, this lack may cause network incoherence which will degrade performance.

All of the problem solving techniques can improve on their predictions through their knowledge of network organization [Durfee 87a]. This implies having an awareness of the general capabilities and responsibilities of other agents and their communications patterns. The multi-agent planning approach already uses the organizational information communicated by the other agents. However, in the negotiation approach, organizational knowledge will allow negotiating agents to use focused addressing techniques to improve the assignments of sub-tasks to other agents. In the

FA/C approach, it allows nodes to decide which of their tasks on their agenda is most likely to further the achievement of the network goal.

We note that the use of organizational knowledge for prediction purposes is applicable only when the network is comprised of agents with heterogeneous capabilities and functions. Organizational knowledge exists by virtue of the disparate-ness of agent nodes and one's ability to choose among them to take advantage of their unique capabilities. According to Durfee et al, organizational knowledge can only make limited improvements to network coherence [Durfee 87a]. Although it assists in determining what each of the agents is likely to do, it cannot predict when important interactions will take place. Therefore, the best possible planning and network knowledge, if taken alone, are still inadequate in producing high network coherence.

To be able to predict the chronology of actions of other agents, each agent needs an insight into the reasoning processes currently taking place in other agents. This can best be provided through appropriate communication policies implemented across the network. Through them, agents can share appropriate information such as hypotheses, goals, facts, etc. Shared goals are primarily used to focus the activities of the receiving agents. On the other hand, shared hypotheses give an indication what the sending agent is attempting to do and provide predictive information. According to Durfee et al, the following three properties of the information communicated among agents have a major effect on the coherence of a PSN [Durfee 87a]:

- 1) **Relevance:** Measures for a given message, the amount of information that is consistent with the solution derived by the network. Irrelevant messages are a source of distraction to nodes attempting to integrate irrelevant or in-

consistent information into their own solutions. An increase in relevance generally increases the network coherence.

- 2) **Timeliness:** Measures the extent to which a given message is likely to alter the current activity of the receiving node. Timeliness depends not only on the contents of the message but also on the current state of the node. If the message will have no effect on the problem solving activity of the node, then there is no point in sending it. However, if a message will further the network goal, then it should be sent without delay.
- 3) **Completeness:** Measures the fraction of the complete solution that a message represents. Partial solutions may create distractions for the receiving nodes and it is desired to minimize distractions to conserve both the computational and communications resources of the network.

We note that these three factors are in fact interdependent. For example, higher completeness leads to higher relevance but reduced timeliness. In the next three sections we will show that blind communication of all available information among agents is counter productive because it may allow agents to become distracted from their primary tasks, and still work at cross purposes to each other at the expense of network performance. Instead, for coherent co-operation, agents require comprehensive communication policies to guide them in the exchange of information. The choice of communications policy to be implemented in a PSN is generally dependent on the problem solving environment and involves a trade-off among relevance, timeliness and completeness. We note that because of their global importance, the network com-

munication policy should reside on the strategy level of each agent's knowledge base [Wielinga 86].

2.3.3.1 Send-All

In the send-all policy, each agent transmits all of its candidate hypotheses and goals to other agents regardless of the degree of confidence of the information. This approach has two main drawbacks. For example, after sending very tentative hypotheses, the sending agent will immediately improve upon these and retransmit. This is very distracting to the receiving agents which will attempt to incorporate both transmissions into their own reasoning processes. Clearly, it may be better to refrain from transmitting information below some prescribed level of certainty. In addition, transmission of all information can generate an unnecessarily high communications load on the PSN.

2.3.3.2 Locally-Complete

Under this policy, agents only transmit locally-complete goals and hypotheses. A hypothesis is locally-complete if the given agent cannot improve on it any more. A goal is locally-complete if the hypotheses which stimulated its creation are all locally-complete, implying that the agent has satisfied the given goal and nothing more remains to be done in the reasoning process. The locally-complete policy is an improvement on the send-all policy in that it minimizes the number of transmitted messages while maximizing the completeness and relevance of each.

However, there may be a high degree of uncertainty whether a given hypothesis is indeed locally-complete. The queue of knowledge sources (KS) pend-

ing execution on the agenda must be examined to see whether there are any KSs present which could improve upon the hypothesis by extending its length or belief. Even if there are, their priority ratings may be so low that they may never be executed and the hypothesis may never be improved. There is also the possibility that on execution of the KSs, there may not be any improvement on the given hypothesis. The above decision process can be very time consuming making the locally-complete policy suffer from poor timeliness. On reflection, we can also postulate that some hypotheses which are not locally-complete but are transmitted in time, may still be very useful to other agents. Clearly, different criteria are required for the sharing of information with other agents.

2.3.3.3 First-and-Last

The first-and-last policy balances the desire for completeness and the desire for timeliness. For reasons of network performance it is important to reduce the number of small and irrelevant hypotheses transmitted among agents without denying them access to predictive information. The approach developed by Durfee et al is a modification of a locally-complete policy described above [Durfee 87a]. However, it has the added feature that if a locally-incomplete hypothesis does not incorporate any previously communicated hypothesis, it is brand new and should therefore be transmitted immediately. This will give potentially useful predictive information to the receiving agents. As before, all locally-complete hypotheses are transmitted for integration into the solutions of other agents. The transmission of all intermediate hypotheses is suppressed because they do not contribute to network coherence, but simply waste network resources.

2.3.4 HARD-WIRED INTERACTION

Other approaches, which are quite different from those developed by Durfee et al [Durfee 87a], have been applied to the problem of inter-agent co-operation. For example, in Takenouchi and Iwashita's integrated knowledge representation scheme [Takenouchi 87], high level reasoning problems are decomposed into a hierarchical structure of sub-problems and corresponding knowledge sources to solve each sub-problem. Through predefined flows of appropriate information between sub-problems, the solution to the highest level problem is derived.

The authors show that the technique is applicable not only to the organization of knowledge within a knowledge base, but it can also represent reasoning processes distributed across a network of nodes. For the latter case, the ability of agents to communicate and exchange data is of course implicit. By virtue of the knowledge source structure which emerges from its use, the scheme has a built-in problem solving strategy and assigns tasks to agents very much like a network planner described previously in the multi-agent planning approach. By establishing detailed specifications for data exchanged between agents, the scheme completely eliminates communication policies from each agent's knowledge base. Because all data flows are predetermined, it is capable of bridging disparate knowledge domains to share knowledge. Finally, the scheme can produce a high degree of network coherence and good computational performance. For these reasons, several researchers have used approaches similar to Takenouchi's with good results to build KBSs where the problem domain was hierarchically structured [Goldberg 85] [Verrilli 87].

However, Takenouchi's scheme suffers from inflexibility with respect to the problems which it can solve. Each problem must be foreseen and programmed directly into the structure of knowledge sources. This is irrespective of whether the implementation is in the form of a single monolithic knowledge base system or a network of intelligent agents. The scheme exhibits similar limitations as do conventionally coded programs with "hard wired" logic. Therefore, Takenouchi's scheme falls short from the standpoints of problem solving generality and flexibility.

2.4 SHORTCOMINGS OF CURRENT APPROACHES

We have seen that coherent co-operation between PSN agents requires highly complex mechanisms for problem solving and communications management, such as those proposed by Davis & Smith [Davis 83] and Durfee et al [Durfee 87a]. These approaches are certainly very promising as vehicles for conducting research in the fundamental principles of distributed problem solving. However, in their present form, it would be difficult to use them in general problem solving for reasons which are discussed below.

2.4.1 NODAL RESOURCE PENALTIES

The implementation of such complex mechanisms entails increased resource requirements for all agents in the PSN. More storage will be needed to store the additional software, increased communications traffic will consume part of the available bandwidth and finally, each agent will require additional computing power in order to maintain an acceptable level of reasoning performance. Otherwise, the network performance may be too low to be useful. Therefore, resource requirements

place a practical lower limit on the capabilities of the computing nodes which can function as intelligent problem solving agents. Below this limit, we enter the realm of fine grain parallelism such as connectionist networks. These are typically fixed function architectures with little or no intelligence in the individual nodes. While it is true that they may display intelligent behaviour, the individual simple processors are incapable of participating in sophisticated co-operative interactions with other intelligent agents.

2.4.2 THE PROBLEM OF HETEROGENEOUS AGENTS

The discussions of agent interaction strategies was based on research done with PSNs comprised of homogeneous problem solving agents. No evidence was found by the author that these methods were ever applied to PSNs composed of heterogeneous agents with disparate computing environments and capabilities.

Specialized knowledge domains may require heterogeneous knowledge representation and semantics. These would effectively block the exchange of goals, hypotheses and findings between agents, thereby preventing the exchange of the predictive information needed to maximize network coherence. Without this predictive information, network coherence would then be determined strictly by the effectiveness of the approach taken to the formulation of network plans and by the use of organizational knowledge. Without additional mechanisms to circumvent the above problems, the feasibility of general purpose network communications policies is in serious doubt.

We also note that the above agent heterogeneity issues prevent the sharing of arbitrary knowledge among problem solvers. It is recognized that in the case of specialized, fixed function networks, all data exchanges, communication protocols and semantics are established at design time. This allows the sharing of specific knowledge

among agents. In this regard, such PSNs resemble implementations of hard wired knowledge source networks proposed by Takenouchi & Iwashita [Takenouchi 87]. However, in a general problem solving environment comprised of heterogeneous agents, it is mandatory that arbitrary knowledge such as findings, be completely portable across the network. Clearly, a separate mechanism is required to solve the heterogeneity problem and facilitate agent interactions.

2.4.3 OVERSPECIALIZED PROBLEM SOLVING CAPABILITIES

Most PSNs built to date can be characterized as being overly specialized in their capabilities. This is understandable because the concept of PSNs is relatively new and researchers are still working at gaining a basic understanding of the phenomenology of agent interaction under very simplified experimental conditions. To simplify the problem, designers have purposefully limited PSNs to homogeneous agents, and optimized the agents to solve a single, well chosen problem. E.g.: Durfee's Distributed Vehicle Monitoring Testbed (DVMT) [Durfee 84] [Durfee 87a], or Davis & Smith's Distributed Sensing System (DSS) [Davis 83]. Highly specialized PSNs certainly do have their applications, but if a broadened problem solving capability is desired, such restrictions would be unacceptable, since they would be inconsistent with the problem solving objectives.

2.4.4 INTERACTION WITH USERS

Perhaps due to their highly specialized nature, PSNs are not usually designed for interactions with humans during the problem solving process. Most research efforts to date have considered a single point-of-access node providing an inter-

face for a user. The functional objectives of these PSNs allowed agents to operate without any dialogue with the user.

However, in a general problem solving environment, a mechanism is needed whereby any node tasked with a reasoning job may, if required, interrogate the user who originated the high level problem to gather relevant facts about the given sub-problem. Unfortunately, if interaction with users is allowed, agents' performance will be limited by the low speeds of human responses. This constitutes a serious bottleneck which may block the execution of the network plan, reducing agent utilization, network throughput and efficiency. Therefore, PSNs designed for general problem solving must be capable of working on multiple problems simultaneously, so that when the solution of any problem is blocked while waiting for a user response, the agent can work on another.

CHAPTER 3

THE BUILDING BLOCKS

OF CONSULTING NETWORKS

3.1 FUNCTIONAL EXTENSIONS TO THE PSN CONCEPT

From the discussion in Section 2.4, we see that the present agent interaction mechanisms cannot readily support general problem solving processes and are better suited to efficiently solving single, well chosen problems. To widen their applicability, it is suggested that PSNs require a change of focus and certain extensions in their functionality.

The primary goal of PSNs should not be specialization, but general problem solving in the widest sense. To distinguish the current generation of problem solving networks from those capable of true general problem solving, we propose the term Consulting Network (CN). Drawing an analogy to sophisticated distributed database applications, problem solving via Consulting Networks is not for the average home computer user, but for certain professional groups such as researchers, medical practitioners, engineers, lawyers, policy analysts, etc. The issues with which they deal are sufficiently important that the successful and correct solutions of their reasoning problems have an intrinsically high value to society. This alone would justify de-emphasizing network performance and efficiency issues in the design of PSNs provided that they were capable of supplying a consistent and reliable service to the user community. With less stringent requirements, it will be possible to simplify the mechanisms for distributed problem solving and agent interaction, and thereby reduce the computational overheads on the individual agents.

The central issue in general problem solving networks is the functional fusion of heterogeneous knowledge sources hosted by disparate agents. Since the current PSN concepts do not support this readily, the Consulting Network concept requires a separate mechanism to bridge heterogeneity and make knowledge completely sharable across the network.

Since Consulting Networks will be tasked with solving problems originating from users, a suitable mechanism must be provided to allow problem solving agents, to interact with humans during the problem solving process without unduly penalizing system performance. Human-machine interaction raises the important issue of how the user may express his original problem so that the machine understands what is required and can commence solving it.

The specific features of Consulting Network nodes is a design-time issue which depends on the size of the knowledge domains to be hosted and the level of service to be provided to users. To justify their existence, Consulting Networks must store and give access to massive amounts of knowledge on a wide variety of subjects. Consequently the computing nodes hosting problem solving agents will be hosted on mini or mainframe computers with abundant memory and disk storage resources as well as high performance computing engines. These represent large capital investments for which a reasonable rate of return will be expected. The availability of established knowledge sources to clientele must be maximized, and therefore Consulting Networks must be able to work on multiple user problems simultaneously. With multiple user problems present, the problem solving control processes must be able to keep track of all sub-problems in the network. This is necessary in order to integrate sub-problems solutions into the final solution to the given problem.

The above extensions comprise the distinguishing features between PSN concept and that of Consulting Networks. They give the ability to functionally unify otherwise isolated pools of codified knowledge into open architecture networks of distributed knowledge sources which can share knowledge among each other, much like a committee of human experts. Conceptually, Consulting Networks bear a strong resemblance to distributed database systems in that with the help of data communication technology, they can be tapped with equal ease by machines or humans whenever advice on complex or multi-disciplinary problems is required. Several potential applications for Consulting Network technology are shown below:

- 1) **A Financial Advisor** which is capable of reasoning about a client's financial objectives in context of the current Income Tax Act, pending tax legislation, various investment opportunities, demographic trends, as well as economic factors;
- 2) **A Medical Advisor** which assesses any given course of treatment in context of possible ramifications on all parts of the patient's organism. In case the proposed treatment may have adverse effects or may be ineffective, the Advisor could suggest alternatives. The system would consist of specialist knowledge sources for human diseases, physiology and pathology of all organs and systems in the human body plus pharmaceuticals and acceptable clinical treatments;
- 3) **A Research Advisor** providing access to the findings in any field of research such as cancer or AIDS research. By sharing codified knowledge among

major research centres or Centers of Excellence, new approaches could be tested from anywhere in the world against work done to date. The most promising directions of further research could be quickly established after eliminating repetitions of previous work and ideas with a low likelihood of success. A system such as this would supplement traditional literature searches wherein a researcher must not only retrieve, absorb and understand the relevant static knowledge from multiple sources, but must also filter out inconsistencies before he can establish his own approach;

- 4) **A Legal Advisor** capable of quickly reviewing all relevant bodies of law such as Torts, Contract Law, Criminal Law, Land Law, Personal Property Law, etc, all relevant legal precedents and render a judgement on any given case. Such a system would be invaluable in reducing the preparation time and the cost of court action and may well introduce a high level of consistency in the judgments rendered by the courts.
- 5) **An Operations Advisor for the International Space Station (ISS)** which would be capable of advising astronauts whether any proposed activity could be carried out within the constraints of the ISS systems status as well as other planned activities. It would function by consulting with the various other KBSs which monitor and control the ISS. In case of system failures, the Advisor would provide advice on how to reconfigure the systems in order to maximize the chances of survival for the spacecraft and crew. Such a system would find ready terrestrial applications in naval vessels, aircraft and even in industrial environments.

The remainder of this thesis examines how to implement the necessary extensions to the PSN concept and proposes a unified functional framework for the construction of Consulting Networks. In this Chapter, we will propose and discuss some high level system attributes and mechanisms necessary for the realization of Consulting Networks. These will provide a broad framework in which the above extensions to the PSN concept may be implemented. We defer a discussion of the interactions between these mechanisms until Chapter 6, where we will present a functional decomposition of a Consulting Network.

3.2 SYSTEM CONSIDERATIONS

3.2.1 AGENT CONNECTIVITY

For the general case one can assume that the agents comprising a Consulting Network will be widely distributed. Assuming that the CN agents will be interacting with each other through some form of message passing, a data communication network to interconnect them is an obvious given.

The physical architecture of this network is largely irrelevant to the functionality of the Consulting Network. Through the use of suitable data communication protocols, such as those compatible with the seven layer OSI model, the CN designer can bridge a wide variety of disparate network types, communication standards and computing host environments and provide the necessary logical interconnections between agents [Stallings 85]. However the logical architecture of a Consulting Network is quite important because it reflects the functional interrelationships between problem solving agents as well as the partitioning of authority and control.

Of the three canonical network architectures, the peer architecture is the most appropriate for use in Consulting Networks because it is the simplest conceptually and because it provides the maximum connectivity for knowledge sharing. There is no single centre of control and agent interactions are dictated strictly by the problem solving activities in progress. Messages may flow freely among all agents. However, depending on other aspects of the distributed problem solving process, peer architectures may result in excessively large spans of authority.

In hierarchical architectures, the position of each agent in the hierarchy is related to its responsibilities and authority. Typically, requests for co-operation will flow downward while responses and unsolicited queries will flow upward. Agents can only task agents one level lower down, with all lower level agents remaining invisible. This implies that if required, the logical architecture can be used to limit the span of authority of agents to an acceptable level by hiding the presence of other agents. This equates to the creation of subnets the presence of which may increase the control complexity in problem solving.

By this token, the net architecture is the least appropriate for Consulting Networks because of potentially irregular structure, multiple access paths and compartmentalization of reasoning resources. All of these factors introduce significant complexity into the control of the problem solving process. Justification for the use of a net architecture could only stem from security requirements wherein it is desirable to prevent access to specific knowledge.

3.2.2 PROBLEM-SOLVING AGENT REQUIREMENTS

In the original concept of PSNs, agents were only required to have the necessary computational and memory resources to support the software functionality, and to have communication capabilities to interact with other network agents. In principle, agents could be relatively simple such as single board computers.

However, agents participating in Consulting Networks (CNs) must meet greater performance and resource requirements to achieve the more sophisticated functionality of the network. CN agents will of course require data communications to support an active mode of interaction as discussed in Section 2.2. Perhaps the most important requirement is that agents must be capable of multi-tasking. This feature strongly affects the economic viability of the Consulting Network concept by improving the utilization of agent's computing resources. In brief, agent multi-tasking is essential for:

- The concurrent solution of multiple reasoning problems;
- The interaction of Problem Solving Agents with users during the problem solving process;
- The hosting of multiple Problem Solving Agents in a single physical node;
- The suspension of reasoning tasks which are waiting for the arrival of data from the user or results from subordinate agents;
- More sophisticated and efficient management of communication processes.

This enhanced functionality will be discussed in greater detail in subsequent sections of this thesis. Clearly, to support such functionality, Consulting Network agents will require a sophisticated operating system such as UNIX, NOS or VMS, multi-megabytes of memory and disk storage resources, as well as high performance computing engines.

3.2.3 MAN-MACHINE INTERACTION

In Consulting Networks, one of the most important parts of the problem solving process is the entry of user problems along with any applicable constraints and conditions, in such a way that the system will solve the correct problem. In order to express complex real-world problems, users require a man-machine interface with great expressiveness. A Natural Language Interface (NLI) is the preferred mode of interaction because of the flexibility and power of natural languages, plus greatly simplified user training requirements. NLI's entail Natural Language Processing (NLP) and Natural Language Understanding (NLU), both of which are major fields of research, and we can only briefly summarize some of the key issues as they relate to Consulting Networks.

Due to the syntactic complexities, semantic ambiguities, and context dependencies found in human languages, machines have great difficulties in understanding natural language text. For example, humans often use slang, dialectic expressions and words with double meanings, but are quite adept at picking up the necessary contextual cues to deduce the semantics. As a prelude to understanding of the semantics of the input text, NLI's use one of several types of parsers to transform natural language into the computer's internal form [Obermeier 87]. In general, this is

a difficult problem because not all human concepts may be accurately translated without contextual information, which adds complexity to the language's syntax. Syntactic complexity is directly reflected in the formal grammar used to parse statements in the given language. Since different human languages contain varying amounts of contextual information, this must be captured as a context-sensitive grammar in the natural language's parser. This implies that NLIs must be specific to each natural language.

NLP and NLU despite being very computation intensive, at present are not very robust, and fail to achieve 100% accuracy in all cases. Therefore, practical NLIs must satisfy at least two more requirements, namely those of verifiability and resilience. Verifiability means that the user must be able to verify the correctness of the system's understanding of his request. Resilience means that an NLI will gracefully recover from ambiguities and unknown words found in user requests, as well as its own linguistic deficiencies [Hendrix 87].

The problems of NLP and NLU become much more tractable by constraining the user to a simplified interface language which is restricted both in syntax and vocabulary. In effect, this ensures that statements from the user can be parsed with the help of regular or context-free grammars. Several software products exist, featuring constrained natural language interfaces, such as Texas Instruments' (TI) Natural Access. It displays a series of structured menus which offer only a limited choice of words for each syntactic element. Although such products are steps in the right direction, they lack sufficient expressiveness for use in Consulting Networks. They are far more suitable for narrow application domains than for expressing the wide variety of concepts needed for general problem solving.

Much more appropriate are approaches such as Jakobson's, who in his work on an intelligent database assistant, faced the identical problem of how to interface a naive user to the machine via a natural language [Jakobson 86]. His approach was an attempt to strike a balance between linguistic completeness, domain coverage, and robust and efficient parsing methods. The machine's understanding of text was for the most part, based on semantic case analysis which used stereotypes of grammatical constructs as an aid in the resolution of imperfect queries. Jakobson's NLI satisfied the verifiability and resilience requirements by giving the user ample opportunities to review and correct the machine's understanding of the problem. It conducted a dialogue with the user to establish his intentions and to resolve ambiguities or unknown objects. This greatly reduced the likelihood that the wrong problem would be solved by the system. We see that Jakobson's approach to the NLI is far more sophisticated and flexible than that of Texas Instruments. Because it recognizes and deals with many of the problems of NLP and NLU, it holds the promise of being able to support the types of dialogue envisaged for CN users. This of course needs to be confirmed by experiment.

One of the major differences between PSNs and Consulting Networks is that the user of CNs must be able to interact with individual problem solving agents while they are working on a particular aspect of his problem. This situation can arise because some agent requires specific facts or supplementary data to carry out his assigned reasoning task. Since the information may be available elsewhere in the Network, the agent must broadcast a data request to other agents. If the data cannot be found anywhere, the request will ultimately be presented to the user. After the user supplies the data, it will be forwarded to the agent which requested it originally so that

he can continue his reasoning. In this interaction process, the multi-tasking capabilities of CN agents help to conserve the agent's computing resources and improve node utilization. If the reasoning process is blocked due to pending data, the problem solving task can suspend itself until the required data arrives, leaving the agent with free CPU time to work on other tasks.

During problem solving, users may wish to enter supplementary problems related to the original problem or to request explanations from the distributed problem solvers. If the problem solvers discard the relevant working data on completion of their assigned sub-problems, it will not be possible to satisfy this requirement without having to re-enter all of the facts relating to the problem. Therefore, the interaction between the user and the Consulting Network must be session-oriented. This means that during the log-on procedure, the user must initiate a session with the network. Subsequently, all problem solving tasks which are involved in solving the given problem must stay instantiated but suspended, retaining all information pertaining to the problem, until the user either terminates them by starting a new problem or by ending the session.

One risk in the above approach is that failures of node hardware or communication facilities in the Consulting Network could generate an accumulation of suspended problem solving tasks which would tie up node memory and disk storage unnecessarily. Therefore, the session-oriented interaction scheme makes it necessary for agents to terminate all problem solving tasks which have been dormant for excessive periods of time and to release all related storage.

3.2.4 BRIDGING HETEROGENEITY

The objective in bridging problem solving agent heterogeneity is to functionally fuse disparate, knowledge bases which would otherwise be isolated within individual agents. Heterogeneity is taken to mean that agents have disparate computing environments, domains of specialization, and types of knowledge representation. If a Consulting Network is constructed of heterogeneous agents, all of these factors can complicate or even prevent its functioning unless a means is established for the bridging the relevant differences. We will discuss each of the above, determine to what degree they constitute a problem and if they do, we will propose appropriate solutions.

3.2.4.1 Disparate Computing Environments

Problem solving agents in a Consulting Network may be implemented on a variety of computer hardware, operating systems and programming languages. These may give rise to differences from machine to machine in the internal representation of numbers and text. Such differences are significant only in the exchange information among agents and can be readily handled through the imposition of data interchange standards on all agents wishing to participate in a Consulting Network. For example, the ASCII code may be used in all communications, and agents will have to convert incoming messages into their internal formats before processing them. Conversely, outgoing messages must be converted into ASCII.

3.2.4.2 Disparate Specializations

As given by basic set theory, the knowledge domains in different intelligent agents may be of three basic types:

- **Disjoint:** The knowledge domains of individual agents deal with completely different subject matter. This represents an ideal partitioning of expertise in a Consulting Network;
- **Intersecting:** Within the agents comprising the network, there are some overlaps in expertise. This is a more general and likely partitioning, especially if the network was constructed without due regard to knowledge domain structuring;
- **Overlapping:** The knowledge domains of some agents are completely identical. This type of partitioning may be found in redundant systems which have been purposefully made that way to ensure survivability or for other operational reasons.

The Chief Designer of a Consulting Network should strive to implement agents with disjoint knowledge domains as this will make each agent unique. The knowledge about agents' disparate capabilities comprises organizational knowledge which is very important in the assignment of reasoning tasks to agents [Durfee 87a]. Disjoint knowledge domains remove ambiguities in agent selection and are therefore highly desirable.

On the other hand, difficulties arise when the network contains agents with overlapping capabilities. Under these conditions, duplicate qualifications may reduce the ability of network organizational knowledge to discriminate among agents. Since

effective task distribution is a keystone of distributed problem solving. Consulting Networks require a means of capturing and using organizational knowledge without any loss of its discriminatory properties, and so that accurate assessments can be made of individual agents' capabilities. We propose an effective technique for just this purpose in Section 4.2.

3.2.4.3 Disparate Knowledge Representation

Real world symbolic knowledge has associated with it preferred representation formalisms, and therefore Consulting Networks will feature a wide variety of knowledge representation schemes. Complex knowledge domains within a single agent may well be implemented with several representation types such as frames, rules, formal logic, predicate logic, etc. We contend that knowledge sources (KSs) should never be exchanged among agents for incorporation into each other's knowledge bases. The primary difficulty is that the representation formalism of the sending agent may not be supported in the receiving agent. For example, the sender represents knowledge as rules, whereas the receiver uses some other formalism. In another example, an agent hosting a database management system, uses integrity constraint mechanisms, but wishes to send to an agent which does not recognize integrity constraints. Clearly, a direct KS exchange would not be feasible and a remapping of KSs between pairs of representation formalisms would be required. This is an extremely difficult problem because not all formalisms are amenable to remapping. For example, the representation of frames as first order logic may have been achieved for special cases, but no general algorithm is known. The direct exchange of KSs raises an important linguistic issue, in that each KS contains names of objects, relationships or concepts, which may be absent

from a receiving agent's vocabulary, or perhaps different semantics may be attached to them. A translation process would be required to ensure understanding.

We can alleviate the above problems somewhat by making a simplification in that individual knowledge sources will not be portable between agents; only findings. Therefore, agents must share only the results of their reasoning processes with other agents. Generically these will consist of:

- Strings of characters or tokens;
- Data structures such as records or arrays of numbers and/or characters. (We take a single character or number to be a degenerate data structure).

The above is deceiving in its simplicity, and one may conclude that few problems will be encountered in knowledge sharing. This is unfortunately not the case. Shared knowledge will consist of arbitrary concepts, objects, facts, data, complex inter-relationships, contextual information, etc. These can only be expressed by means of a highly expressive language with many of the attributes found in natural languages. This constitutes a linguistic problem of the same order of complexity as NLIs and which must be solved before knowledge domain fusion can be realized.

3.2.4.4 Linguistic Issues in Distributed Problem Solving

Within Consulting Networks, messages may originate from system users as well as from any of the Network agents. When a user poses a problem to an agent through a natural language interface, the NLI will determine the semantics of the problem as discussed in Section 3.2.3, above. As part of this process, the user's problem will be transformed into an internal machine form consisting of a sequence of symbols

will be transformed into an internal machine form consisting of a sequence of symbols or tokens. Subsequently, the agent will use this internal form to reason about the subject matter of the problem. The internal form is completely dependent on the characteristics of the given agent such as the knowledge representation method, the reasoning strategy, programming language, etc. Should this agent subsequently need to vector a reasoning task to another agent, it makes little sense for the task description to be expressed using the internal representation of the originating agent. Unless the tasked agent had the identical internal representation, the received task would simply not be understood. The same reasoning also applies to other communications such as the transmission of reasoning results and unsolicited messages.

The use of a human language does not constitute a solution because the receiving system would have to pass the message through an NLI before it could be understood. As discussed in Section 3.2.4, human languages exhibit syntactic complexities, contextual dependencies and semantic ambiguities which make the implementation of robust, general purpose NLIs extremely difficult. Also, for system throughput reasons the low performance of sophisticated NLIs constitutes an undesirable bottleneck.

Worst of all, inter-agent communications using a human language opens up the nightmare that problem solving agents manufactured in different parts of the world will understand different indigenous human languages. For example, a sender may communicate in English while the receiver understands only Japanese! Aside from the problem of developing and maintaining software for many-to-many language translation, the processing burden, sensitivities to contextual factors, and the attendant seman-

tic uncertainties introduced by multilingual translation are completely unacceptable in Consulting Networks.

3.2.4.5 KDL: The KBS Discourse Language

As we have seen, it is unlikely that vocabulary terms and concepts of one agent can be directly usable by other agents in the Network. Clearly, there is a strong requirement for a common language of discourse among co-operating agents to circumvent these problems. That is to say, co-operating agents need a "Lingua Franca" through which to formulate messages containing task descriptions, findings, responses and generally, a wide variety of concepts. Much research is currently being done in the theory of languages, and we assume that a suitable artificial Knowledge Base System Discourse Language (KDL) can be developed for this purpose.

Several inter-system languages already exist as a result of this linguistics research. For example, Jakobson et al developed a virtual database language called V/DELPHI and used it to implement an intelligent database assistant capable of bridging several distributed DBMSs having disparate structures, query languages and operating environments [Jakobson 86]. V/DELPHI serves as an umbrella language for several target languages. Its syntax was chosen to be such that queries expressed in V/DELPHI could also be expressed in each of the target languages. The rigorous structure of database queries makes them amenable to translation into V/DELPHI through the use of rules based on the source language grammar. An important part of a translation process is to convert the names of V/DELPHI objects into actual field and file names appearing in the target databases. The above database access problem has a

strong similarity to the problem of accessing disparate, distributed knowledge sources hosted by agents within a Consulting Network.

In a second and more sophisticated example, the Fujitsu Corporation of Japan has developed a prototype computer called "Atlas 2" for automatic translation between human languages such as Japanese, French, German and English [Hillenbrand 89]. Spanish, Chinese and Korean are expected to be added in the near future. A key component of the software is a coded, largely numerical language called "Interlingual". It serves as the hub of the translation process, in that text from a source language is first translated into Interlingual and then from Interlingual into the target language. This prevents the combinatorial explosion of dictionaries and grammatical knowledge bases which would become necessary for direct translation between arbitrary language pairs. At present, Interlingual can successfully translate only the typically stilted technical literature. More sophisticated prose and poetry, involving contextual dependencies and highly sophisticated grammatical constructs are currently beyond its capabilities. However, the high pay-off associated with the development of a successful general purpose translation computer will undoubtedly greatly stimulate the further development of Interlingual.

We contend that the principles underlying V/DELPHI and Interlingual may be applicable to the development of a KDL to support the communication of knowledge between Consulting Network agents. Although the design of the KDL is beyond the scope of this thesis we suggest some of its main attributes. Ideally the KDL should feature a regular, or at least a context-free grammar which directly supports the processes for distributed problem solving. The grammar must comprise protocol elements and discourse elements. The protocol elements will be used to mediate all com-

munications between agents and to assist in the parsing and identification of the message data fields. Typical protocol elements may consist of:

- Requests to open or terminate a session. (See Section 3.2.3);
- Requests for agent capability. (See Section 4.3);
- Descriptions of problem or tasks;
- Assignments of a given task to a specific agent;
- Requests for supplementary data on a specific subject. (See Section 3.2.3);
- Requests for explanations
- Transmissions of findings or the results of reasoning;
- Transmissions of supplementary data;
- Transmissions of explanations;
- Transmissions of an agent's problem solving status.

The discourse elements in the KDL grammar will apply only to message fields embedded by the above messages. However, much greater expressiveness is required to convey discourse elements such as:

- Task descriptions;
- Solution constraints;
- Requests for specific supplementary data;
- The expression of supplementary data;
- The results of reasoning;

All of these necessitate a rich syntax coupled with a comprehensive vocabulary to give the KDL the required semantic depth to express the subtleties of the problem domain concepts to be communicated. It is presently unclear whether or

not the KDL could be successfully implemented with a context-free grammar. If so, this would simplify the design of the parser used in the translation process described below and improve its performance. The KDL vocabulary must span the knowledge domains of all agents in the Consulting Network without redundant terms or ambiguities. It must consist of generic functional concepts and objects with precisely defined semantics. Perhaps a subset of a natural language lexicon with emended semantics could form an initial KDL vocabulary, with periodic additions implemented as needed. A most important aspect of KDL development is that its grammar and vocabulary must be subject to strict configuration control to prevent the development of dialects which would impede inter-agent understanding.

To interact with other potentially heterogeneous agents in the network, each agent will need to translate messages between his own private domain language and the KDL. The objective is to ensure comprehension of communications without imposing commonality within individual agents; merely at the interfaces. Therefore one of the fundamental attributes of a KDL - Domain Translator is that it must support mappings to and from domain-specific grammars and vocabularies, thereby allowing each agent to represent knowledge and to reason in its own private manner. There is however one restriction. The Translator must not attempt to translate proper names supplied by the user of the Consulting Network in the course of a consulting session. Such names may not be in any agent's vocabulary, yet they may play a part in the problem solving process as specific instances of generic objects recognized within the relevant knowledge domain. To alter them could result in a loss of information for the user. Figure 3-1 below, depicts the conceptual translation process to and from the KDL.

Figure 3-1: Conceptual KDL Translation Process

In order to add a new agent to the Consulting Network, its vocabulary must be integrated with the current KDL vocabulary and the revisions distributed to all of the agents' Translators. However, the grammar for the agent's domain-specific language should remain private because it is not required outside of his context. We see that if implemented properly, the KDL will be a key element in the process of bridging and functionally unifying heterogeneous knowledge sources.

CHAPTER 4

**CONSULTING NETWORK PROBLEM SOLVING
PROCESS**

4.1 INTRODUCTION

DAI researchers find it convenient to view distributed problem solving as consisting of four distinct phases [Davis 83]:

- 1) **Problem Decomposition Into Sub-Problems.** A planning mechanism in the PSN takes the given complex problem and divides it into several simpler tasks or sub-problems which are more tractable;
- 2) **Sub-Problem Distribution.** The sub-problems are then distributed among the network agents, taking into account issues such as the capabilities of individual agents vis-a-vis the sub-problem attributes, the balancing of the computational load, the conservation of communication bandwidth, etc;
- 3) **Sub-Problem Solution.** Each of the autonomous network agents solves its assigned sub-problem(s) using its local planning and reasoning resources;
- 4) **Answer Synthesis.** The answers of the individual agents are then fused together to generate a composite solution for the original problem.

In this Section, we will consider each of these phases and discuss the relevant low level mechanisms needed for the implementation of Consulting Networks.

4.2 PROBLEM DECOMPOSITION

Many knowledge and problem domains feature a natural structuring of the information or stereotypical formats. Therefore in planning, humans often take advantage of this by partitioning problems into a series of sub-problems to be solved separately and combining the results at the end. As discussed in Section 2.3.1, all attempts to produce a robust general problem solving methodology have failed so far. However, general planning within narrow knowledge domains has been shown to be viable [Hayes-Roth 83][Waterman 86].

We therefore believe that for purposes of Consulting Networks, a useful level of general problem solving can be achieved through hierarchical decomposition of the given problem into a tree of sub-problems all of which, we initially assume, can be locally solved. This process is called "problem reduction" and attempts to transform a problem description into a set of reduced or successor problem descriptions. Problem reduction implies that if we solve all of the successor problems, we will have the answer to the original problem [Nilsson 71]. Any locally unsolvable sub-problems must be distributed to agents who have the specific expertise to refine the overall plan in these areas and are capable of solving the sub-problems. This process, when performed by all relevant network agents will, in the end, produce a complete distributed sub-plan hierarchy. Subsequently, by merging the reasoning results from these agents in reverse hierarchical order, the solution to the original general problem can be produced. Depending on the attributes of the problem and the expertise of the problem solving agents, the final result should be functionally indistinguishable from that produced by a single general problem solver. An important advantage of hierarchical problem solv-

ing is that, unless the strategy requires multiple solutions to a specific sub-problem, a reasoning task will be given to only one agent. This avoids the difficult problem of dissimilar and possibly contradictory solutions which need to be resolved through the application of metaknowledge.

4.2.1 Problem Solving Through Limited Co-Operation

Hierarchical problem reduction will potentially involve only those knowledge sources (KSs) which can contribute something towards the problem's solution. If these KSs are distributed agents, in effect this limits both the scope of inter-agent co-operation and its potential complexity. To distinguish this type of co-operation from the network-wide co-operation studied by Durfee et al [Durfee 87a], we will refer to it as "limited co-operation", and upon which we propose to base the distributed problem solving process in Consulting Networks.

Under limited co-operation, agents are not required to co-operate with any other agents unless they have been explicitly requested to do so. Co-operation consists of performing specific, clearly defined reasoning tasks and returning the results of this reasoning to the tasking agents. An agent receiving an assignment may not refuse the job unless it is either malfunctioning or is off-line. We note that this conforms to the client-server model described in database management or distributed processing literature. On receipt of an assignment, the agent will formulate and execute a structured plan for carrying out the reasoning task. On completion of the reasoning task, the agent will send the results of the reasoning to the agent which assigned him the job. At this point and unless requested otherwise, the co-operative relationship between this pair of agents terminates.

Limited co-operation is a hierarchical, distributed problem solving process. In the event that the tasked agent does not possess the necessary knowledge sources or facts, and is blocked from executing the entire plan, it must request co-operative assistance from other agents in the network which do have the requisite capabilities. When the desired data items arrive from the other agents, they will be included in the agents fact base, thus enabling the completion of the original reasoning job. Limited co-operation is completely recursive throughout the network. For example, a human operator may enter a query into one network node which, depending on its capabilities, may subsequently spawn a series of reasoning sub-tasks to other agents. Successive stages of sub-task spawning may take place and will depend entirely on the local problem solving activities of each layer of agents. The result will be a propagation of parallel reasoning activity throughout the network which will terminate when the problem has been decomposed to such a degree that all involved agents are able to solve the component sub-problems identified in their local plans. In the end, the desired answer will be presented to the user.

The limited co-operation approach has two important advantages. First, it completely decentralizes the planning and control of network activities by allowing any agent to function as a local planner. It greatly reduces the scope of agent interactions, giving all agents the responsibility for controlling their interactions with two logical levels of neighbours: a superior and a limited number of subordinates. For these reasons the network does not have a single point of failure. Second, if task distribution is done via negotiation as discussed in Section 2.3.2.2, the task descriptions and interactions between agents will be completely specified, thus removing the need to accurately predict other agents' activities which, as we have seen earlier, is so hard to do

[Durfee 87a]. Therefore, under limited co-operation there is no need for agents to exchange hypotheses, goals and findings as discussed in the FA/C communication policy.

Because of hierarchical problem decomposition and fixed, limited scope interactions among agents, under limited co-operation the agents' Planner function can be characterized as a hierarchical tactical planner. See Section 2.3.1. It is recognized that due to its simplicity, limited co-operation may not yield optimal network performance, but it does have the potential of simplifying the control of the problem solving process and reducing the complexity of the network software.

4.3 SUB-PROBLEM DISTRIBUTION

The assignment of tasks to the best possible agents is critical to the success of distributed problem solving and constitutes one of the main aspects of planning. Durfee states that the use of organizational knowledge is very important in making better assignments of tasks to agents, thereby increasing network coherence. Organizational knowledge allows one to use focused addressing or task vectoring techniques to solicit co-operation from the best qualified agent [Durfee 87a]. However, several fundamental questions must be answered first:

- 1) How does an agent determine that it is at the limits of its knowledge domain and must seek help from another agent?
- 2) How can organizational knowledge describing the capabilities of network agents be best represented?

- 3) How can one use organizational knowledge to determine the competence of an agent on a given subject?
- 4) How can a task description be transformed into a list of qualifications which agents have to meet in order to be considered as potential candidates to perform a particular reasoning task?

In subsequent sections we will attempt to answer these questions and establish the functional requirements for further extensions to the flexible architecture for co-operating agents.

4.3.1 Determination of Knowledge Domain Limits

Limits to a system's knowledge domain may be determined either by careful design or declared as a default condition. In considering the first case, we know that a KBS's domain knowledge is captured and codified by the system's designers and knowledge engineers. It is not unreasonable to expect them to define the boundaries of their system's knowledge domain and to prevent attempts to reason beyond these limits. However, we note that any preventative measures must exist by careful design in order to make their outcome dependable and deterministic. We anticipate that to function well, these measures will require appropriate metaknowledge emplaced within each agent's knowledge base. For example, on detection of specific circumstances, such as an attempted access of data items or knowledge sources which do not belong in the domain of the given agent, a process may be initiated to invoke external help:

IF (CONDITIONS) THEN (REQUEST EXTERNAL HELP)

The encounter with domain limits during local planning or reasoning may arise under rather complex circumstances unforeseen by the system builders. To handle this eventuality, the capabilities of the agent's Planner and Scheduler functions would have to be enhanced to monitor, detect and handle these conditions. For example, during the formation of an agent's local plan, the agent may not recognize some or all of the objects in the given problem statement or in a sub-task component of the plan.

Also, during the reasoning process, a condition may arise when none of the pending goals can be satisfied because there are no more knowledge sources which could improve upon the outstanding hypotheses. This means that the reasoning has deadlocked and that the answer cannot be determined by the agent. In effect, this indicates that the limits of knowledge have been reached. When either one of the above two conditions has been detected, as a default procedure, the agent's Scheduler function should attempt to obtain external help. The declaration ANSWER UNKNOWN should be held in abeyance until it is determined that external help is unavailable.

The design of a robust general purpose mechanism for spawning sub-tasks to other agents above problems is both very complex and very important, lying at the heart of the Consulting Network concept. Its solution is beyond the scope of this thesis and requires additional research.

4.3.2 Representation of Organizational Knowledge

In judging the qualifications of an agent to solve a given problem, it defeats the purpose of co-operation to examine his codified domain knowledge. Instead, an examination of the agent's domain metaknowledge should be quite sufficient. In the context of limited co-operation, domain metaknowledge of an agent has been defined

to mean the high level knowledge about all subject matter or topics falling into an agent's domain of expertise. Assuming that each agent possesses his own domain metaknowledge, then domain metaknowledge is identical to self-knowledge which in turn is a subset of network organizational knowledge.

The object of capturing domain metaknowledge is to produce a complete and unambiguous catalogue of topics on which a given agent is capable of rendering judgement. The representation scheme must be capable of efficient implementation from the standpoint of memory/disk storage as well as provide rapid access. As an example, let us consider a zoological knowledge base system. The topic DIET is virtually meaningless if taken in isolation. It could apply equally to any animal and have several different meanings. However, DIET can become both meaningful and specific if it is put in context with more general, higher order knowledge, concepts or topics. For example, one may state that DIET pertains to Siamese cats which are a type of domesticated feline, which is a type of mammal, which is a class of animal. Suppose we express the above statement in the following form:

ANIMALS, MAMMALS, FELINES, DOMESTICATED, SIAMESE, DIET

This is easily distinguished from:

ANIMALS, MAMMALS, CANINES, DOMESTICATED, POODLES, DIET

which means the diet of poodles which are a type of domesticated canine, which is a type of mammal, which is a class of animal. Clearly, a subsumption relationship exists

between any level of topic and the one below it. As one goes from the general to the specific, we note that every level of topic inherits properties from higher level topics.

As shown above, each topic may be considered to be an element in a chain of descriptors which we call a topical chain. As one reads the topical chain from left to right, each element is a sub-topic of the element immediately preceding it. Therefore, the semantics of any topic may be uniquely established in context of higher order topics. It is our contention that all topics within the domain of expertise of an agent may be classified and catalogued in this way, thereby capturing its domain metaknowledge.

Although with hash sorting algorithms rapid access would not be a problem, it would be very inefficient from the standpoint of data storage to store topical chains as a tabular data structure. As shown below, there could be much repetition:

ANIMALS, MAMMALS, FELINES, DOMESTICATED, SIAMESE, DIET
ANIMALS, MAMMALS, FELINES, DOMESTICATED, SIAMESE, LIFESPAN
ANIMALS, MAMMALS, FELINES, WILD, OCELOTS, DIET
ANIMALS, MAMMALS, FELINES, WILD, OCELOTS, LIFESPAN
ANIMALS, MAMMALS, CANINES, DOMESTICATED, POODLES, DIET
ANIMALS, MAMMALS, CANINES, WILD, WOLVES,DIET
ANIMALS, REPTILES, etc.

Since there is a hierarchy in the ordering of the elements in a topical chain, it would be much more efficient to store this type of information as a tree or network as shown in Figure 4-1 below. Topics and sub-topics are simply depicted as nodes in the tree and the subsumption relations between topics are shown by arcs.

Figure 4-1: Storage of Topical Data as a Tree Structure

In AI literature, this type of data structure is known as a semantic net. Semantic nets are virtually a free-form method of representing knowledge. Without conceptual extensions, they have no formal semantics and contain no reasoning logic capable of computing likelihood, belief, disbelief, promise or scores [Schubert 76]. An ownership relationship may exist between nodes, implying associativity but not cause and effect. As such, semantic nets are useful in depicting inheritance and property hierarchies, category-feature relationships and complex class-feature relationships [Patrick 86].

Topically oriented semantic nets (TSNs) have been successfully used for the classification, storage and retrieval of topical knowledge [de Haan 86]. Although this

application is not directly relevant to assigning tasks to agents, it shows that TSNs are a very efficient and unambiguous way of organizing and storing topical information. We propose to apply TSNs as a conceptual tool to organize the domain metaknowledge of network agents and create a catalogue of agents' capabilities. Provided that the topical classification scheme maintains single inheritance throughout, TSNs may be readily implemented in software as m-way trees [Horowitz 82]. In subsequent discussions, for the sake of clarity, we will stay on the conceptual level and continue to refer to trees containing domain metaknowledge as TSNs. For task vectoring purposes, TSNs yield several very important advantages:

- 1) TSNs are completely independent of the knowledge domain contents and the knowledge representation method internal to the agent;
- 2) TSNs are completely independent of the agent's implementation language;
- 3) TSNs are completely independent of the host operating system and hardware environments;
- 4) If used strictly as a catalogue, TSNs do not appear to have any fundamental limitations. (One question remains regarding the fineness of knowledge needed to accurately identify candidate agents).

Therefore, due to these characteristics, we believe that TSNs are an excellent way of capturing the organizational knowledge of a Consulting Network.

4.3.3 Self-Knowledge and Agent Capability Assessment

Once the domain metaknowledge has been captured in TSN format for all agents in the network, it may subsequently be used to determine optimum tasking of network agents. The proposed method is relatively straightforward.

First, the task description must be analyzed to derive a chain of topical descriptors which exhibit a subsumption relationship between adjacent links when read from left to right. We will call this the Query Topical Chain (QTC). It is important that the QTC have enough topical descriptors to unambiguously classify the task subject matter. Then the agent's TSN must be thoroughly searched with the object of finding a branch containing a series of successive topical nodes which are the closest match to the QTC. It is deemed that an agent is ideally qualified to execute the task if the entire QTC is found in any branch of the TSN.

In practice, one may find that the degree of correlation between the QTC and various branches of the TSN is less than ideal. Consequently, a scoring algorithm must be applied to the pattern matching to derive a figure of merit or score. The score should be normalized by dividing by the maximum possible score for the given QTC. Hence it will have values ranging from 0.0 to 1.0. (We will discuss and propose a suitable scoring algorithm in Section 5.2.4). Finally, we may say that the agent is competent to perform the given task if the normalized score exceeds some predefined threshold level. We will now consider how to identify the most appropriate agent in the entire Consulting Network to perform a given task.

4.3.4 Selection of an Optimum Task Target

It is possible to use either organizational knowledge or self-knowledge to assign tasks to agents. The two types of knowledge differ only in their scope and in how they are used. Network organizational knowledge consists of the total domain metaknowledge for all of the agents in the network. Since domain metaknowledge is represented as TSNs, network organizational knowledge means the sum total of all TSNs. An agent possessing network organizational knowledge can at any time assess the capabilities of all network agents for purposes of task assignment. Self-knowledge on the other hand, consists of a single agent in possession of his own TSN, cataloguing his own domain metaknowledge. This agent can only judge his own capabilities with respect to solving a given problem and has no perception of the capabilities of the other agents in the network.

Although both approaches are based on QTC searches through TSNs to judge agents' capabilities, they are in fact quite different, they have different system implications, and will therefore be discussed separately in the next two Sections. Because of their importance in the task distribution process as well as their potential to affect Network performance, in Chapter 5 we will perform an analysis of both approaches to assess their relative merits.

4.3.4.1 Task Assignment Through Network Organizational Knowledge

In task assignment based on network organizational knowledge, each network agent is given the TSNs and corresponding agent identifiers for all other agents. When a given agent encounters a need for external reasoning help, he becomes respon-

sible for finding the most capable agent on a given subject through the use of the TSNs in his possession. The basic procedure is an extension of the one described above:

- 1) The task description must be converted into a QTC;
- 2) Each candidate agent's TSN must be searched to find the highest degree of pattern match between the QTC and consecutive strings of topical nodes. This will be expressed as a normalized score derived through a scoring algorithm. For each TSN searched, the maximum score achieved during the search and the corresponding agent identifier must be retained in preparation for the final step;
- 3) The tuples consisting of agent identifiers plus the maximum scores are placed into descending order. This list is the order of likelihood in which the various agents are capable of performing the given task. We will call this the Likelihood List (LIL).

The process of LIL derivation is shown in Figure 4-2 below. Subsequently, the tasking agent will use the LIL to vector tasks on the given topic. It can be seen that the task of determining an optimum task target has been reduced to a problem in pattern matching while searching through a tree structure. Search and pattern matching algorithms are well understood and documented in the literature [Horowitz 82].

Realistically, the order of likelihood contained in the LIL above, is only an estimate. If the most likely agent fails to perform the task, it is possible that some other agent further down the list may succeed. Therefore, the tasking system must be pre-

Figure 4-2: LIL Derivation and the Search for a Task Target

pared to evaluate the response of the tasked agent and react according to a task vectoring strategy similar to those shown below:

- 1) **One Shot Tasking:** Abandon any further tasking even though other candidate agents may exist in the Likelihood List (LIL);
- 2) **Successive Tasking:** Task each of the remaining agents until either a satisfactory answer is received or the LIL is exhausted;
- 3) **Thresholded Successive Tasking:** Task each of the remaining agents with a score above some preset threshold until either a satisfactory answer is received or the list of candidates is exhausted;
- 4) **Task Broadcast:** Task all agents appearing in the LIL, collect their responses, reject non-credible answers and construct a composite answer from the remaining responses to solve the original problem.

The first strategy, the one shot tasking is rather simplistic. It is usable only in those Consulting Networks where the agents' knowledge domains are all disjoint, the TSNs are very accurate and the QTCs comprise very detailed problem topical classifications. Then the probability of success on the first tasking attempt is very high. For more general applications, such perfection is unlikely and this strategy would all too often fail to access readily available knowledge from other agents.

The two successive tasking strategies (2 and 3), although they are more complex, than the first, can be implemented providing that an adequate criterion can be established for what constitutes a satisfactory result from performing a task. "Satisfactory"

implies a comparison to an expected result or to some comprehensive quality standard. This in turn implies that the tasking agent must have some additional knowledge which can be used to discriminate between answers.

Strategies 2 and 3 depend on the assumption that if agents are tasked in descending order of score, the first satisfactory response is probably usable to the task originator. Therefore the tasking process should be terminated either on receipt of the first satisfactory answer or when the list of candidate agents has been exhausted.

The fourth strategy, task broadcast, raises the interesting yet troublesome possibility that contradictory answers could be received from some of the queried agents. This too, could be quite difficult to resolve without some additional high level knowledge about the subject matter of the task and the credibility of the responding agents.

Although other strategies may be envisaged, of the four presented, it is felt that strategy 3, thresholded successive queries, holds the most promise. It is superior to strategy 1 because, if required, it is capable of tasking more than one agent in the Network. It is more efficient than strategies 2 and 4 which blindly task every agent in the Network irrespective of their levels of expertise. Most important of all, it avoids the worst pitfalls of the task response analysis in strategy 4. We acknowledge that strategy 3 still needs to assess agent responses and that further studies of this problem are indicated.

4.3.4.2 Task Assignment Through Self-knowledge

In the self-knowledge based approach to task assignment, agents do not have access to TSNs of other agents; only to their own. The approach still requires the derivation of a LIL, but the process is somewhat different from the network organizational knowledge approach in that agents will ask all other agents in the network to search their own TSNs. The proposed procedure for each agent is shown below:

- 1) The tasking agent converts the task description into a QTC as described previously and transmits it along with his identifier to all other agents in the network. We will call this type of message a Request for Capabilities (RFC) message;
- 2) Each agent who receives the RFC message, searches through his own TSN to establish his own degree of competence on the basis of the TSN search score;
- 3) Having searched his own TSN, each agent must decide whether to respond to the sender of the RFC message. The decision can be made on the basis that the TSN search score exceeded some preset minimum value. This threshold value can be established empirically for the network. If above the threshold, a response message will be sent to the RFC originator. Otherwise, to conserve network communications resources, the agent will completely abstain from responding and carry on with other activities;

- 4) An agent's response message will consist of the respondent's identifier and the TSN search score;
- 5) The RFC message originator collects all responses received within some allowable time limit and puts them into descending order to obtain the Likelihood List (LIL) as described in the organizational approach. We note that a time limit to receive responses is required because agents might be busy with other tasks and may be delayed in processing the RFC message;
- 6) Finally, having produced the LIL, the tasking agent must determine to which of the candidate agents to assign the reasoning task. Two approaches are suggested. The first is to use the task assignment strategies described in Section 4.3.4.1 to assign the reasoning task.

A second, an possibly complementary approach is to enter into negotiations with agents in the LIL to determine their availability and the costs associated with doing the job. One candidate mechanism is Davis and Smith's Contract Net Protocol which is gaining acceptance for agent task assignment [Davis 83] [Boettcher 87]. We note that in keeping with the proposed MMI concepts and the Consulting Network functionality, the Contract Net Protocol would have to be extended to support:

- Requests to open and end a consulting session;
- The distribution of supplementary questions or requests for explanations from the user;
- Requests and the forwarding of supplementary data;

Whatever the strategy used, the tasking agent will assign the reasoning task to the optimal agent by sending it a message containing a task description.

4.3.5 Relationship Between the KDL and the Task Distribution Process

In previous sections we showed how TSNs can be used to catalogue the domain knowledge of individual Network agents. This knowledge is of course the subject of discourse between co-operating agents. We also established that to ensure understanding among heterogeneous agents, all message traffic must take place in the KDL, as discussed in Section 3.2.4.5.

Success in locating likely task targets depends on being able to match QTCs against branches of the TSN. With all else being equal, this depends on whether the topical descriptors comprising the QTCs are found in the TSN. Clearly, it is undesirable to distribute and maintain TSNs in a variety of agents' languages. Since the KDL is designed to bridge linguistic differences between agents, it then follows that QTCs, TSNs and sub-problem descriptions must all share the same KDL vocabulary. Adherence to this convention is fundamental to the success of the task vectoring process. Figure 4-3 depicts the major stages through which sub-problems generated by a computer or a human are converted into vectored task messages to other agents, and indicates the important role of the KDL.

4.3.6 Generation of Query Topical Chains (QTCs)

The accurate and automatic generation of QTCs from task descriptions is one of the key processes in task distribution because it is the mechanism through which candidate agents are identified. In principle QTCs can be derived from problem

Figure 4-3: Role of the KDL in Task Vectoring

descriptions through the process of topical analysis. Topical analysis involves the application of the rules of grammar, vocabulary and a topical classification scheme. The process is far from trivial and requires further research. At present we depict it conceptually in Figure 4-4 and offer general guidelines for its implementation.

First, the problem statement must be parsed and analyzed with the help of the domain-specific grammar rules and vocabulary in order to identify all structural components such as the subject, predicate, the main and subordinate clauses. These in turn will be analyzed to extract candidates for QTC topics. Although nouns and verbs are the primary objects of this search, adjectives and clauses will generate sub-topics of the nouns or verbs which they modify. The outcome will be a series of topical elements.

Next, the above topical elements must be mapped into standard topical descriptors or keywords. As discussed in Section 5.2.2, as part of this mapping, these keywords may have to undergo an adjustment in granularity and a removal of redundant terms. The resulting keywords must then be placed into a hierarchical order according to a topical classification scheme which must be applicable across the entire Consulting Network. See Section 5.2.1. The result will be a QTC in the agent's language capturing the topical content of the given problem description.

The final step is to translate the QTC keywords into the KDL. This ensures that the QTC will be composed of words found in the KDL vocabulary, thus fulfilling the common vocabulary requirement discussed in Section 4.3.5. Now the QTC can be used to search through TSNs.

Figure 4-4: Formation of QTCs From Task Descriptions

4.3 SUB-PROBLEM SOLUTION

Having been assigned a reasoning task to carry out, the agent must first translate the task description from the KDL into his own private representation. He will then be able to formulate a local plan to solve the problem. The execution of the plan will be completely under the control of the agent's inference engine, with the reasoning taking place in complete isolation from events occurring in other parts of the Consulting Network.

As discussed in Section 3.2.3, should the agent require additional facts or declarative knowledge, he must request it from the agent which assigned him the reasoning task. After sending an appropriate request for data to his superior, in order to conserve computing resources, the agent will suspend himself until an answer arrives. If during the reasoning process, the agent discovers that he cannot solve the assigned problem or any one of the sub-problems, he will attempt to delegate them to other agents in the Consulting Network. This will be done in accordance with the limited co-operation approach described previously in Section 4.2.1. Having assigned the sub-problem to another agent, the given agent will continue working as long as possible. Should he run out of work, the agent will suspend himself pending the arrival of the required information from his subordinate(s). When finally the information arrives, the agent will incorporate it into his own fact base and continue with his reasoning process until his assigned problem is solved.

4.4 ANSWER SYNTHESIS

On each level of problem solving, the process of answer synthesis is exactly the reverse of the hierarchical problem reduction process described in Section 4.2. Each agent constructs a problem solving plan based on a reduction of his given task into a hierarchy of sub-tasks. It is in attempting to solve each of the sub-tasks individually that the sub-task delegation process takes place. Regardless of which agent actually solves any given sub-task, during the planning a "slot" is assigned for the solution to each sub-task. Consequently, when a solution is either computed locally or is received from a subordinate agent, it is simply placed into the appropriate place in the logical problem structure.

The appearance of solutions to any one of the sub-tasks may enable the agent to advance his reasoning towards the solution of his assigned task. When the final sub-problem solution appears and the reasoning process terminates, the given agent will in turn transmit the task solution to his superior in the problem solving hierarchy.

As discussed earlier, the economic viability of Consulting Networks depends on high utilization of individual agents. This implies that Consulting Networks must be capable of serving multiple users and working on multiple problems simultaneously. Therefore they require a mechanism to keep track of the multiple layers of sub-problems and problem solutions existing simultaneously in the network. Consider the problem solving hierarchy of entities shown in Figure 4-5 below, consisting of several users, two Network Agents, several Problem Solving Tasks spawned by one Network Agent and a tree of Sub-Problems belonging to one of the Problem Solving Tasks. Let us assume that problem solving is done in accordance with the limited co-opera-

Figure 4-5: The Problem Hierarchy and the Tracking of Sub-Tasks

tion concept and that human interaction is session oriented. See Sections 3.2.3 and 4.2.1. Then only four data items are needed by a task tracking mechanism to uniquely identify a sub-problem to any layer of agents and place it into proper context within their problem solving activities:

- The session identifier (S_i) which corresponds to the i -th user;
- The identifier of the j -th Network Agent (NA_j) which received a problem from a superior agent;
- The identifier of the k -th Problem Solving Task (PST_k) instantiated by the Network Agent to solve the given problem. (See below);
- A identifier of the l -th sub-problem (SP_l) which is to be delegated to another agent by the above Problem Solving Task.

This task tracking mechanism is made possible by session oriented problem solving and the fact that each Consulting Network agent is hosted within a computer capable of multi-tasking. All sub-problems spawned by any layer of Network agents must be tagged with a tuple consisting of S_i , NA_j , PST_k , SP_l . When an agent receives a task assignment from a superior agent, it immediately instantiates a Problem Solving Task for the duration of the session and passes it the tuple information. This effectively lays a trail for the return of the solution to the tasking superior agent for purposes of answer synthesis:

- The NA_j field allows the subordinate Problem Solving Task to determine to which superior physical Network Agent to transmit the sub-problem solution;
- The PST_k field allows the superior Network Agent to route the answer to the originating Problem Solving Task;

2 OF/DE 2

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

- The SP_i field allows the superior Problem Solving Task to place the answer in its own problem solving plan;
- The S_i field is used only to manage the assignment and subsequent release of agents' storage resources when a session terminates.

Task tracking is performed automatically by each layer of the problem solving hierarchy. Each layer of Problem Solving Tasks will simply append their own NA_j , PST_k and SP_i identifiers to the session identifier S_i and will be able to keep track of answers from multiple lower level agents. This process will culminate in the presentation to the user of the solution to his problem.

CHAPTER 5

ANALYSIS OF THE LIL DERIVATION PROCESS

5.1 INTRODUCTION

Effective task distribution among Network agents plays an important role in improving the coherence and therefore the performance of a Problem Solving Network [Durfee 87a]. In Section 4.3.4 we proposed the use of Network Organizational Knowledge (NOK) or Self-Knowledge (SK) to generate Likelihood Lists (LILs) containing candidate agents for task distribution purposes. Since in our approach the LIL derivation process is the central mechanism for task distribution, its efficiency and effectiveness will have a major effect on the functionality and performance of Consulting Networks. We define efficiency in terms of the elapsed time to generate the LIL and the computing resources needed. Effectiveness is a function of the adequacy of the results. We note that both of these will be affected by the design of the TSN searching and scoring algorithms.

In this Chapter we will discuss various ancillary issues relevant to TSNs and QTCs, and then propose a simple TSN searching and scoring algorithm to establish a design point around which future work can be oriented. Then we will formally analyze the Network Organizational Knowledge and the Self-Knowledge approaches for performing the TSN searches. Finally, we will present the results of a simulation of both strategies and compare them qualitatively.

5.2 THE TSN SEARCH AND SCORING ALGORITHMS

Although on the surface, algorithms to perform searches through TSNs and compute QTC matching scores are conceptually simple, their effectiveness is sensitive to factors such as:

- The topical classification scheme used;
- The granularity of domain metaknowledge capture;
- The weighting scheme used for QTC terms.

5.2.1 Effects of Topical Classification

Topical classification is the methodology of constructing topical hierarchies for both TSNs and QTCs. At the heart of any topical classification scheme is a set of rules or principles which govern the subsumption relation between pairs of topics and through which a consistent knowledge structure may be constructed for use within Consulting Networks.

The creation of TSNs will be the responsibility of the developers of the individual KBSs targeted for a given Consulting Network. However QTCs will be computed autonomously at run-time by each tasking agent. It is useful to impose the same topical classification scheme across the whole Network because this can be exploited to improve the performance of the TSN search process. Under a consistent classification scheme, terms within QTCs and TSNs will appear in a certain order, reflecting the subsumption relationship between generic topics. Then the design of the TSN search algorithm may be based on ordered matching. That is to say, the algorithm will expect

that the order in which QTC terms are arranged will match the order in which they are found in the TSN.

For example, suppose a pattern match is found within a TSN branch for the first three terms of a 7 term QTC. If the algorithm suddenly encounters a match with the seventh term, then there is no point in searching that branch of the TSN for terms 4, 5 and 6. The algorithm may terminate the examination of the given branch and proceed to the next branch, thereby saving time. However, in a Network without a global topical classification scheme, ordered matching would not work. Because out-of-order matching would have to be used in the algorithm, all terms in QTCs would have to be searched against all nodes in the TSNs and the process would take much longer.

We therefore conclude that global topical classification schemes are indeed beneficial in task vectoring. However, rather than inventing a new and unique scheme, it may be possible to adapt an existing one to Consulting Networks. For example, the Library of Congress scheme is used to catalogue books on the basis of their subject matter. It is extremely detailed and has been accepted virtually world wide. A second scheme with similar applications is the International Index of Scientific Translations. As an indicator of their effectiveness, we note that both schemes are widely used by literature search databases.

It is unimportant that these schemes have been developed for applications somewhat removed from Consulting Networks. What is very relevant, is that both espouse a logical classification methodology which may be useful in task vectoring. The whole issue of knowledge classification is sufficiently important to the success of the Consulting Network concept, that further research on this subject is clearly indicated.

5.2.2 Effects of Knowledge Granularity

Granularity of knowledge is a measure of the degree of fine detail that is recognized by the topical classification process. For example, a coarse granularity would recognize ENGINEERING as a topic. A finer granularity would recognize ELECTRICAL ENGINEERING, CIVIL ENGINEERING and MECHANICAL ENGINEERING, etc.

Knowledge granularity enters into the construction of TSNs as well as the derivation of QTCs. If the granularity of QTC knowledge is much coarser than that of TSNs, then the search process will not be able to identify agents with desired specific knowledge. Also, the Network TSNs will be relatively speaking unnecessarily large, the Network agents will require more storage and the search process will take longer. Conversely, if the granularity of QTCs is finer than that of TSNs, searches would not find matches for QTC terms. Candidate scores would suffer unnecessarily and reduce the ability of the process to discriminate among candidate agents to the detriment of the task distribution process. It is therefore important that the Network Design Authority impose a knowledge capture procedure which yields a uniform granularity of Network topical knowledge.

Unfortunately, since the granularity of QTCs is largely determined by the information content of problem statements, a granularity mismatch with TSNs becomes inevitable and must be rectified at run-time. Granularity matching consists of mapping each of the raw QTC terms extracted from problem statements into standard generic terms which match the level of granularity found in TSNs. Considering the case where

the granularity of a QTC is finer than that of a TSN, say the raw QTC consists of the terms:

MAMMALS, DOMESTICATED, DOGS, SPANIEL, COCKER, DIET

The granularity matching process first notes that DOGS is a non-standard term which should be replaced by the generic term CANINES. Also, SPANIEL as well as COCKER should both also map into CANINES. This yields an intermediate QTC consisting of:

MAMMALS, DOMESTICATED, CANINES, CANINES, CANINES, DIET

As a final step, the QTC is condensed by the removal of repetitive terms into:

MAMMALS, DOMESTICATED, CANINES, DIET

The result is a QTC which matches the granularity of Network TSNs. The case where the granularity of the QTC is too coarse is not harmful per se except, as we pointed out earlier, it results in a loss of ability to identify agents with specific knowledge. Since this case represents a lack of information, the fineness of granularity of QTCs cannot be meaningfully increased by remapping and must be accepted by the QTC derivation process. As an illustration, the final form of the QTC above cannot be completely reverse-mapped into the initial form because the process has no way of knowing that the specific information required was about cocker spaniels.

The only envisaged exception is in the case where the QTC has few terms and numerous agents could potentially respond to the RFC message. See Section 4.3.4.2. Then a limited expansion of the highest order term in the QTC would be beneficial to provide a slightly broader context for the specific information, thereby reducing the number of agent responses.

We observe that the granularity matching process in performing remapping, is quite capable of simultaneously translating terms from domain-specific vocabularies directly into the KDL. See Section 4.3.5. Strings of topical descriptors in any language have no syntax associated with them, only a specific ordering of terms which is preserved during translation.

5.2.3 Proposed TSN Search Algorithm

For purposes of this thesis, we present a recursive TSN search algorithm which is based on the most beneficial simplifying assumptions: ordered matches and uniform knowledge granularity. This will result in best performance in terms of speed and candidate agent discrimination. Future versions may well address more difficult conditions at the cost of increased algorithmic complexity and slower execution.

Because the TSN is not normally a dynamic data structure, we have chosen to represent it as a static m-way tree [Horowitz 82]. The approach we have taken is to traverse the tree in preorder form (i.e. depth first). For each branch of the TSN we will extract and construct a list of topics. Only after an entire branch has been extracted will pattern matching against the QTC be attempted and a branch score computed. The maximum branch score in normalized form will be taken as the TSN's score and hence the given agent's ability to respond on the specified topic. As shown in Figure 5-1

below, the algorithm consists of three procedural modules. Compiled Pascal code listings for the individual modules can be found in Appendix A. The comments in the module headers and interspersed among the lines of code completely document the functions of each module.

Figure 5-1: TSN Search Algorithm Structure Chart

5.2.4 Proposed Scoring Algorithm

The scoring algorithm computes a figure of merit which is a reflection of how good the pattern match is between the QTC and a given agent's TSN. The central issue in the design of the algorithm is the weighting factors to be placed on successful matches with the terms in the QTC. In this regard we have three potential strategies:

- All terms are of equal significance and therefore the weights are all uniform;
- Higher order terms corresponding to higher order topics are more significant than lower order terms. Therefore the weighting factors are larger for higher order terms;
- Lower order terms corresponding to lower order topics are more significant than higher order terms. Therefore the weighting factors are larger for lower order terms.

These strategies are reflections of the value to be placed on specificity versus generality. One may argue that matches with the low order topics (i.e. least significant) in a QTC indicate that a given agent has the specific knowledge to perform the reasoning task. However, if these low order topics are not put into the context of higher order topics, it could lead to significant errors in the determination of agents' capabilities. For example, the QTC consisting of terms like DISEASES, INFECTIOUS, BACTERIAL, INTESTINAL_TRACT may apply equally well to a variety of life forms ranging from humans to farm animals and fish. To a medical practitioner wishing to get advice on how to treat a human patient, the difference is of course cru-

cial. We therefore conclude that higher order terms must be given larger weights than lower order terms.

Having established the general strategy, we are still left with the question of what is the optimum sequence of values for the weighting factors. Say we have the above QTC with binary weighting factors:

QTC TERMS	WEIGHTS	CASE A	CASE B
DISEASES	8	Match	
INFECTIOUS	4		Match
BACTERIAL	2		Match
INTESTINAL_TRACT	1		Match
QTC SCORE	15	8	7

In Case A, a single match was found with the highest order term, while in Case B, the three lower order terms matched the QTC. Clearly, the agent corresponding to Case A would be tasked with the reasoning job without any regard to relevant specific knowledge. We conclude that binary weightings are inappropriate because they allow high order terms to completely dominate the agent selection process. This is just as undesirable as having lower order terms dominate. Clearly, the optimum weighting function represents a compromise which recognizes both the importance of contextual information found in high order terms as well as the contributions of low order terms. We therefore conclude that the determination of an optimal weighting function requires additional analyses and research.

A proposed generic scoring algorithm incorporating arbitrary weighting factors was imbedded in procedure **MATCH** shown above. This algorithm conforms to the preliminary description found in Section 4.2.3 in that the output variable **Branch_Score** is the raw branch score obtained from pattern matching of a QTC with the list of topics found in a single branch of the TSN. The determination of the maximum **Branch_Score** is performed in procedure **FIND_LEAF**. Finally, the score normalization is performed within procedure **TSN_SEARCH** to make the results fall in the range from 0.0 to 1.0. A score of 0.0 denotes that the given agent has absolutely no expertise on the query subject, whereas a score of 1.0 indicates that a perfectly qualified agent has been identified.

5.2.5 Performance of the TSN Search Process

The TSN search and scoring algorithms shown in Section 5.2.4 were compiled on a Borland Turbo Pascal 5.0 compiler on an Everex System 1800B PC/AT clone running MS-DOS Version 3.30. This machine featured an Intel 80286 processor with a Computing Index of 11.2 relative to an IBM PC/XT.

A test case TSN consisting of 20 nodes was constructed along with a series of QTCs ranging from 1 to 5 terms. By repeating the test run continuously for 1,000 times, the author was able to measure that the computation of normalized scores took from 0.050 to 0.051 seconds. This is not exceptionally fast, but in a real Consulting Network application the TSN search algorithm would be coded in assembly language to maximize performance.

5.3 ANALYSIS OF TSN SEARCH STRATEGIES

Since the LIL derivation strategy plays such a key role in the reasoning task assignment process, we will now formally analyze the LIL derivation strategies based on Network Organizational Knowledge (NOK) and on Self-Knowledge (SK) to see how they compare against each other. The basis for our comparison will be the following performance parameters:

- 1) The elapsed time for a tasking agent to derive a LIL;
- 2) The computational load on the tasking agent in deriving LILs;
- 3) The overall computational load on an agent operating under a given strategy.

It is recognized that this analysis is on a very high level and quite limited in scope because of the early stage of development of the Consulting Network concept. In the future, only when the details of candidate implementation mechanisms are firmed up, can a meaningful performance model of the entire CN be constructed which includes reasoning processes. This analysis will certainly have to be extended to estimate other performance parameters. In the meantime, we purposefully restrict ourselves just to the two LIL derivation strategies. Let us begin by defining the following variables which will be used in the analysis:

F_s = Average frequency of subtask generation from each agent under prevailing load

F_{so} = Average frequency of subtask generation from each agent under no-load conditions

K_1, K_2 = Constants

K_r = Fraction of agents responding to an RFC message

L_{LIL} = Computational load on an agent associated with deriving LILs

L_o = Overall computational load on an agent

L_r = Computational load on the RFC receiver

N_a = Number of agents in the Network

T_{LIL} = Average time to derive a LIL

T_m = Time to send or receive a message

T_{mp} = Average message propagation time

T_{search} = Average time to search TSNs for all agents except task originator's

T_{sort} = Average time to sort a candidate agent's response into proper place in the LIL

T_{TSN} = Average time to search a single agent's TSN

5.3.1 The Network Organizational Knowledge Approach

In analyzing the Network Organizational Knowledge (NOK) approach we have made the following simplifying assumptions:

- 1) The QTC has already been derived from the task description and the RFC message is ready for transmission;

- 2) Miscellaneous processing overheads in the agents' nodes is ignored.

The time required to search all agents' TSNs except that of the agent originating the task is:

$$T_{\text{search}} = (N_a - 1) T_{\text{TSN}} \quad (1)$$

Then the time to derive the LIL is simply the search time plus the time to sort the candidate agent ids and scores:

$$T_{\text{LIL}} = T_{\text{search}} + (N_a - 1) T_{\text{sort}} \quad (2)$$

Assuming that an agent will be attempting to derive LILs for task purposes at a frequency F_s , the steady state computational load will be:

$$L_{\text{LIL}} = F_s T_{\text{LIL}} \quad (3)$$

Substituting equations (2) and then (1) into (3) we get:

$$L_{\text{LIL}} = F_s (N_a - 1) (T_{\text{TSN}} + T_{\text{sort}}) \quad (4)$$

Since under the NOK approach, the derivation of LILs imposes a computational load only on the task originator, the overall load on an agent will be:

$$L_o = L_{\text{LIL}} \quad (5)$$

However, the frequency of LIL derivations depends on the reserve processing capacity of the agent. Hence:

$$F_s = F_{so}(1 - L_o) \quad (6)$$

Substituting (4) and (6) into (5) and solving for L_o we get:

$$L_o = F_{so} T_{\text{LIL}} / (1 + F_{so} T_{\text{LIL}}) \quad (7)$$

$$= L_{\text{LIL}} \quad (8)$$

5.3.2 The Self-Knowledge Approach

In analyzing the Self-Knowledge (SK) approach we have made the following simplifying assumptions:

- 1) The QTC has already been derived from the task and the RFC message is ready for transmission;
- 2) The originating agent needs to transmit only one single broadcast RFC message to reach all other agents in the Network;
- 3) The time required to transmit or receive the RFC message is the same and includes all of the overhead of communications protocols and message processing;
- 4) The communications medium is an arbitrary Network and the average message propagation time all agents is T_{mp}
- 5) Miscellaneous processing overheads in the agents' nodes is ignored;
- 6) The messages of nodes responding to the RFC message arrive without conflict and with zero intervals between them. Since the originating agent is not kept waiting, this represents an ideal case.

Figure 5-2 shows an event timing diagram for processes taking place across the Network associated with the emission of a single RFC message. From the diagram we see that the time needed to derive the LIL is:

Figure 5-2: Network Event Timing Diagram for RFC Message Emission

$$T_{LIL} = T_m + T_{mp} + T_{TSN} + K_r(N_a - 1) T_m + T_{mp} + K_r(N_a - 1) T_{sort} \quad (9)$$

$$= T_{TSN} + \{K_r(N_a - 1) + 1\} T_m + 2 T_{mp} + K_r(N_a - 1) T_{sort} \quad (10)$$

An agent originating RFCs with an average frequency F_s will experience the following computational load associated with LIL derivation:

$$L_{LIL} = F_s \{T_m + K_r(N_a - 1) T_m + K_r(N_a - 1) T_{sort}\} \quad (11)$$

$$= F_s K_1 \quad (12)$$

This is clearly less than the load given by equation (7) because a large part of the work has been carried out by agents receiving the RFC message and also because the originator is not busy while the message propagates to the other agents. Throughout the Network, $(N_a - 1)$ tasking agents will each be generating RFC broadcast messages with a frequency F_s . Since the receiver of an RFC message must take action, then the loading due to the arrival of RFC messages will be:

$$L_r = F_s (T_{TSN} + 2 T_m)(N_a - 1) \quad (13)$$

$$= F_s K_2 \quad (14)$$

Hence for any agent operating in a SK based Network, the overall load due to queries which it originates plus RFCs which it receives will be:

$$L_o = L_{LIL} + L_r \quad (15)$$

Substituting equations (6), (12) and (14) into (15) we get:

$$L_o = F_{so}(K_1 + K_2)/\{1 + F_{so}(K_1 + K_2)\} \quad (16)$$

Finally, substituting (6) and (16) into (12) we obtain a revised expression for L_{LIL} :

$$L_{LIL} = F_{so} K_1/\{1 + F_{so}(K_1 + K_2)\} \quad (17)$$

5.3.3 Discussion of Relative Merits

The above performance equations were programmed in Lotus 123 for the following representative parameter values:

Number of agents in the Network (N_a)	= 2 to 100
No-load subtask frequency (F_{so})	= 0.1 Hz.
Fraction of agents responding (K_r)	= 0.1
TSN search time (T_{TSN})	= 0.015 sec.
Time to sort candidate agent response (T_{sort})	= 0.000005 sec.
Time to send/receive message (T_m)	= 0.001 sec.
Message propagation time (T_{mp})	= 0.001 sec.

The above parameters are not based on any specific algorithms, software or processors. They are merely order of magnitude values which can yield a strictly qualitative appreciation of the relative performance of the NOK versus the SK approach. Figures 5-3 to 5-6 below, show comparison plots of the results for the two approaches. By virtue of the mathematical formulation, these plots represent upper bounds on performance. It is clear that the NOK based approach is very slow in deriving LILs whereas the SK approach is extremely fast and is rather insensitive to growth in the number of Network agents. The computational load due to LIL derivation is also much higher in the NOK approach.

However, when we compare the overall computation loads, as shown in Figure 5-5, it becomes clear that the SK based approach is extremely heavily penalized by the steady state arrivals of RFC messages which must be processed. The loads are virtually identical with the SK approach generating about 3% higher load on the agents

Figure 5-3: LIL Derivation Time vs Number of Network Agents

Figure 5-4: LIL Computation Load vs Number of Network Agents

Figure 5-5: Total Computation Load vs Number of Network Agents

than the NOK approach. Effects of this load penalty will be exacerbated whenever a task induces the generation of cascaded tasks among other agents. One can expect that as the scale of the Network or the size of knowledge bases is increased, the SK approach will probably result in network saturation before the NOK approach.

For both approaches, the overall load characteristics clearly indicate that the Network architecture must have a limit which is dependent on the performance of individual agents, on the number of agents and on the average size of agents' TSNs. Beyond this limit, the Network would simply become too burdened with non-productive work, leaving less and less time to perform useful reasoning tasks. These problems may be handled through a hierarchical structuring of the problem solving agents ac-

cording to their domains of expertise. This will limit the span of control and reduce them to a manageable level.

Figure 5-6 below, portrays the relationship of the LIL derivation time to the overall load on network agents. As stated above, the overall load is not a strong distinguishing parameter in the selection of strategy. However, the LIL derivation time for the NOK approach is much greater than for the SK approach. With everything else being equal, the size of the network will establish the average load on individual agents. Therefore, the use of the SK approach will to some degree offset the slight load penalty by yielding much faster derivation of LILs. One can expect that in practice, the overall load on an agent will probably have spikey characteristics. With the use of the SK

Figure 5-6:

LIL Derivation Time vs Load

approach for task distribution, fast LIL derivation may allow agents to take advantage of momentarily reduced loading and continue with task assignment, thus expediting the solution to the network problem.

There are, however, other considerations which may affect the decision as to which strategy to use. Under the NOK approach, task originators will themselves narrow down their choices of agents without involving other agents. This will significantly reduce the load on Network communications facilities such as bandwidth, storage buffers, etc. Another issue is that of managing agents' TSNs. Under the NOK approach, each agent in the Network will possess every other agent's TSN, which will greatly increase the required resources across the network to store TSNs. Multiple copies of TSNs raise the problem of configuration management. An agent undergoing changes to his knowledge base must send updates for his TSN to all other agents as soon as possible. Otherwise the other agents may have invalid expectations about the capabilities of the agent in question to the detriment of the task distribution process.

In contrast, the SK is based on each agent having to store only his own TSN. Since multiple copies of TSNs will not be required, this will reduce the TSN storage requirements of the Network. By the same token, each agent will need to update only his own TSN should his knowledge base change, thereby simplifying the configuration management problem. This constitutes a very important advantage for the SK approach because it helps to reduce the coupling between Network agents. Reduced coupling will in turn will promote an open architecture for Consulting Networks and permit completely transparent deletion or addition of agents.

In summary, from the above, we see that agent overall loading is not a strong distinguishing factor between the two approaches. However, the speed with which LILs

can be derived, TSN configuration management issues, the degree of openness of the CN architecture, and network resource requirements certainly are. In this respect, the SK approach is judged to be superior to the NOK approach in all aspects but in the use of network communications resources. In the next Chapter we will describe the functionality of a Consulting Network based on the Self-Knowledge approach.

CHAPTER 6

A FUNCTIONAL MODEL OF CONSULTING

NETWORKS - A FRAMEWORK

FOR KNOWLEDGE DOMAIN FUSION

6.1 INTRODUCTION

The objective of this chapter is to create a framework for the understanding of Consulting Networks. We will describe their functionality on a sufficiently high level of abstraction that the reader will not get lost in the details of the implementation technologies described in Chapters 3 and 4. We will trace the processes taking place during a representative problem solving session. By focusing on the flow of data and control, we will and show that the proposed mechanisms for agent interaction are consistent and complete. Provided that the individual mechanisms can be implemented, the Consulting Network concept will indeed be capable of supporting a useful level of general problem solving.

In carrying out this decomposition, two assumptions were made:

- All of the mechanisms described in Sections 3 and 4 are operational;
- The reader is reasonably familiar with the relevant technologies which were discussed Chapters 3 and 4, and is able to appreciate the underlying details.

The discussion will revolve around a hierarchical functional decomposition of a Consulting Network using the graphics oriented Structured Analysis Design Technique (SADT) methodology [Ross 77]. Systems depicted by means of SADT consist of a hierarchical series of diagrams. Objects on each level of diagram consist of functions, directed data flows, directed control signals between functions, and external sources and destinations for the data and control flows.

The basic graphic symbology of SADT is very simple. Boxes represent functions while arrows represent directed data and control flows. Data items and control signals are given as blocks of text adjoining the relevant arrows. External data and control flow sources and destinations are represented as text labels. Data flows are allowed to enter function boxes only from the left and emerge only from the right or bottom. Control flows can enter function boxes only from the top. If appropriate, data flows can be used as control signals to other functions.

The SADT methodology demands a strict accounting of all control signals and data items flowing into and out of each function box, and, by the same token, between adjacent levels of diagrams. The purpose of disclosing a lower level of detail of any function box is to show how the control signals and data items will be generated or used internally by still lower level sub-functions. The process can proceed to any level of functional detail desired.

SADT is completely divorced from implementation issues: is a given function to be implemented in hardware or in software, what kind of processor architecture is required, etc. Instead it represents systems at the highest possible level of functional abstraction, thereby promoting clarity of understanding of the functional interrelationships. Figures 6-1 to 6-7 depict the functionality of a Consulting Network.

6.2 NETWORK FUNCTIONAL DECOMPOSITION

6.2.1 The System

We will begin the functional decomposition by viewing the Consulting Network as a system and represented as a box with specific inputs and outputs. See Figure 6-1. It is envisaged that the user will access the Consulting Network through a simple terminal or PC plus a modem. During log-on, the user's identification will be established and a consultation session will commence.

Problem solving will be conducted as an interactive session, requiring the user's constant presence at his terminal or PC. The user will interact with the Network primarily through function menus in order to use the predefined functions such as transaction logging, printing to a printer, etc. The operator's inputs comprise Miscellaneous Inputs, while the generation of prompts, menus, screen displays, error messages, etc are Miscellaneous Outputs.

To express his problem, the human will use natural language. Because of uncertainties encountered in natural language understanding, the Network will conduct a structured dialogue with the user to help establish his true intentions with respect to the problem to be solved. Users have a far greater understanding of language than computers. Therefore, the user's main function will be to confirm or to modify the Network's understanding of the problem by entering additional text to clarify misconceptions. This process will allow the Network to identify the problem's discrete structural elements and linguistic terms as an aid in establishing the problem's semantics. During this formulation of the problem statement, the Network will be restricted to using simple, straightforward grammatical structures and unambiguous terminology.

Figure 6-1: The System Level

When finally the user confirms that the Network completely understands the problem, the Network will attempt to solve it.

During problem solving, the Network may require supplementary data such as facts, names, numerical data, etc. Therefore, it will ask the user to supply the missing information so that the problem solving may proceed. It will be the responsibility of the user to respond as quickly as possible, or this will impact the time needed to derive the final solution. In due course, the Network will present the user with the solution to his problem.

The Network will preserve the problem solving state of the last problem solved. This will allow the user to enter a supplementary problem relating to the previous problem and not have to re-enter all of the supplementary data entered previously. More important, it will allow him to ask the Network to explain its reasoning on the given problem. When the user specifies the beginning of a new problem, all data relating to the previous problem will be permanently erased to save Network resources. At any time, the user may terminate the session and log off. In this case too, all data relating to the problem solving will be erased.

6.2.2 Level 0 - The Consulting Network Function

On the Consulting Network function level we see that the Network consists of two kinds of nodes: Network Access nodes and Problem Solving Agent nodes. See Figure 6-2.

Network Access Nodes (NANs) specialize in performing human interface functions discussed in Section 6.2 such as:

Figure 6-2: Level 0 - The Consulting Network Function

- Generation of function menus, prompts and error messages due to faulty inputs from the user;
- Providing assistance to the user to formulate his problem /ia problem formulation dialogues;
- Querying the user for supplementary data;
- Presentation of problem solutions to the user;
- Explanations for reasoning results;
- Etc.

Problem Solving Agent Nodes (PSANs), on the other hand, contain specialized knowledge sources which will be tasked with solving parts of the user's problem. Due to the hierarchical method of problem solving, the Network Access Node which interfaces with the user will pass the correctly formulated problem statement to the most appropriate PSAN who will begin the actual problem solving.

We must emphasize that the interaction between the NAN and the first PSAN must be identical to the interaction between any pair of PSANs, as discussed in Section 4.3.4. This is shown in the symmetry of data flows in and out of the NAN and the first PSAN. After completing the formulation of the user's problem, the NAN will broadcast a Negotiation message called an RFC (Request For Capability) to determine the capabilities of other network agents with respect to the problem. Each PSAN possessing appropriate qualifications will respond to the NAN with another type of Negotiation message called a RFC Response. In accordance with the strategy described in Section 4.3.4.2, and possibly after exchanging further negotiation messages, the task of solving the problem will be awarded to a single PSAN. Prior to task award, a session on behalf of the user will be initiated by means of a Session Request

message to the PSAN. The PSAN will confirm the session establishment by responding with a Session Grant message to the NAN.

During the problem solving session, the first PSAN may query the NAN for supplementary data which the NAN will obtain from the user if it does not already possess it. (We remind the reader that the request for the supplementary data may have originated from a PSAN far down in the problem solving hierarchy. The first PSAN is merely the interface to the NAN). In due course, the PSAN will send the solution to the given problem to the NAN for presentation to the user.

If desired, the user may request an explanation of the reasoning underlying the problem just solved. This request in the form of an Explanation Request message will propagate from the user, through the NAN and subsequently through the entire network of PSANs which participated in the problem solving session. The relevant PSANs will respond with Explanation messages consisting of conventional text. These will be assembled bottom-up through the hierarchy of PSANs and forwarded to the user. As long as the problem subject remains the same, the user may issue additional problems or explanation requests to the Consulting Network. When the user starts a new problem, the NAN will terminate the session with the PSAN hierarchy by means of a Session End message. This will of course erase all user data throughout the network pertaining to the previous problem, and release the reasoning resources for other uses. Session termination will be performed automatically should the user log off.

6.2.3 Level B - The Problem Solving Agent Function

Within a single Problem Solving Agent Node, there are 5 main functions. See Figure 6-3.

The Receive Incoming Messages (RIM) function executes the relevant data communications protocols to capture all messages targeted at the given PSAN. As discussed in Section 3.2.1, the RIM function communication protocols should be compatible with the OSI model to ensure maximum agent connectivity for the Consulting Network. We can divide incoming messages into two groups depending on their origin:

- From superior agents, Request For Capability (RFC), Session Request, Session Termination Request, Problem Statement, Supplementary Data and Explanation Request messages;
- From subordinate agents, RFC Response, Session Grant, Problem Solution, Supplementary Data Request, Explanation messages.

The Manage Problem Solving Node (MPSN) Function is responsible for decoding all incoming messages passed to it by the RIM Function and routing them to one of the remaining three functions in the Problem Solving Agent. We will now examine the processing of specific message types.

When the PSAN receives an RFC message containing a QTC, the message will be forwarded to the Determine Agent Capability (DAC) Function. Within the DAC Function, the given PSAN's TSN will be searched for matches against the received QTC, an agent capability score will be computed. In accordance with the Network strategy on RFC handling, if the score exceeds the predetermined response threshold, the DAC Function will formulate an RFC Response message which will be

Figure 6-3: Level B - The Problem Solving Agent (PSA) Function

transmitted to the RFC Message originator via the Send Outgoing Messages (SOM) Function. If the score is too low, the DAC Function will not respond in order to conserve Network communication bandwidth.

Whenever the MPSN Function receives a Session Request message, it will instantiate an independent Solve Problem (SP) task to solve the expected problem and deal with any related follow-on tasks. To highlight this feature, the SP Function has been shown as multiple function boxes. This is where their multi-tasking capabilities allow PSANs to handle multiple reasoning problems simultaneously, synthesize solutions and interface with humans without subjecting the host processor environment to gross utilization inefficiencies. This mechanism will be driven by status information received from the SP Function. As we shall see below, SP Function status information will allow the MPSN Function to:

- Suspend SP tasks awaiting information from subordinate or superior agents;
- Awaken tasks again when relevant information arrives;
- Dynamically create or terminate problem solvers as problem solving sessions begin and end.

On receiving a Session End message for a particular session, the MPSN Function will command the relevant SP task to propagate the message downward through the hierarchy of agents under its control. When the SP task returns a Subordinates Terminated status, the MPSN Function will terminate the task. To handle the eventuality of hardware failures in the Consulting Network and to conserve nodal resources the MPSN Function must terminate all PS tasks which have been inactive for an inordinate length of time.

In the problem solving agent hierarchy, the Solve Problem (SP) Function instantiated by the MPSN Function is the intermediary between superior agents and subordinate agents. Driven by the messages received from superior agents via the MPSN Function, the SP Function will attempt to solve the given problem. If it is unable to find the solution through its own knowledge sources, the PS Function will delegate reasoning tasks to a still lower level of subordinate agents. Whether dealing with superior or subordinate agents, the SP Function's outgoing message traffic will go via the Send Outgoing Messages (SOM) Function.

As discussed in Section 3.2.4.5, problem solving involves translation to and from the KDL. Therefore the SP Function requires access to the syntax, grammar and vocabularies both of the KDL and the KBS's domain language. This knowledge will reside in a hard disk based or RAM based library within each PSAN.

The Send Outgoing Messages (SOM) Function is responsible for transmitting all messages originating within the PSAN to the specified destinations over the applicable communications medium. The message types are identical to those received by the RIM Function but with the origins reversed. For exactly the same reasons as the RIM Function, the SOM Function communication protocols must conform to the OSI model.

6.2.4 Level BB - The Manage Problem Solving Node Function

As shown in Figure 6-4, the Manage Problem Solving Node (MPSN) Function performs five subordinate functions:

The Decode & Route Message (DRM) Function examines messages arriving from the RIM Function and decides whether they should be passed on for action

Figure 6-4: Level BB - The Manage Problem Solving Node (MPSN) Function

to existing SP tasks or do they require other actions. The latter consist of instantiating, suspending, awakening or terminating SP tasks and comprise the remaining lesser functions.

The Instantiate SP Task (ISPT) Function instantiates or creates a problem solver SP task to handle problems relating to the given session. On instantiation, the SP task is added to a table of active SP tasks managed by the problem solving agent and the SP task will now be eligible to receive and send messages. A new task will only be instantiated if the above SP Task Table does not already contain a task for the given session which has completed its assigned reasoning, as evidenced by a Status = End.

The Suspend SP Task (SSPT) Function suspends a given SP task when the MPSN Function receives an Idle or End status from the task. This indicates that all reasoning processes within the task are finished or blocked and are awaiting the arrival of outside information. The SP tasks's status will be amended appropriately in the SP task table.

Conversely, when a message arrives for a suspended task, the Awaken SP Task (ASPT) Function awakens it, thus enabling it to receive the message and continue reasoning. Here too, the SP tasks's status will be amended in the SP task table.

The Terminate SP Task (TSPT) Function allows the MPSN Function to terminate any given SP task under two conditions:

- If the task has been inactive for an excessively long time;
- If a Session End message has been received for a given session.

Task inactivity will be determined by a set of timers within the MPSN Function. Each SP task's personal timer will be reset when the SP task is originally instantiated, whenever the SP task sends or receives a message, or when the SP task informs

the MPSN Function that it is idle or has finished solving the given problem. The timeout period should be set as one of the Network control parameters.

In case of Session End, the SP task will be reawakened if it has been suspended and will be commanded to inform its subordinate agents (if any) of the Session End and to release their resources. Finally, when the given SP task responds with a Subordinates Terminated status, the SP task itself will be terminated and deleted from the SP task management table.

Figure 6-5 shows a state transition diagram which clarifies the interaction of the five MPSN functions.

6.2.5 Level BD - The Solve Problem Function

Figure 6-6 shows the internals of the Solve Problem (SP) Function. Since the concept of Consulting Networks recognizes agent heterogeneity, the SP Function will be allowed to operate with its own knowledge representation scheme, domain vocabulary, semantics, and use its own reasoning techniques. Consequently, all messages incoming from the MPSN Function must be translated from the KDL into the domain-specific language and representation format. This process will be done by the KDL - Domain Translation & Message Routing (KDTMR) Function. The translation will use the syntaxes, grammars and vocabularies of the KDL and Domain languages. Some additional translation rules will be required to discriminate between messages and fields requiring translation and those which do not. Following translation, the KDTMR Function will forward each message to other functions within the SP Function.

Figure 6-5: MPSN Function State Transition Diagram

Figure 6-6: Level BD - The Solve Problem (SP) Function

Conversely, the Domain - KDL Translation (DKT) Function will perform an identical translation process for all outgoing messages generated within the SP Function. After translation, all messages will be sent to the SOM Function located within the Problem Solving Agent Function.

If the KDTMR Function receives a problem statement, it will be forwarded to the Create Plan (CP) Function which will generate a hierarchical plan to solve the given problem. See Section 4.2. Because this plan will be needed later to combine the solutions to the sub-problems, the CP Function must send it to the Combine Solutions (CS) Function. See Section 4.5.

Initially, all sub-problems will be passed to the Solve Locally Solvable Sub-Problems (SLSSP) Function on the assumption that they can be solved within the scope of the given agent. During the problem solving, the inference engine of the SP Function may in fact determine that some of the sub-problems cannot be solved locally as per the discussion in Section 4.3.1. Therefore, this gives rise to two classes of sub-problems: locally solvable and locally unsolvable sub-problems. The locally solvable sub-problems will be solved by the SP Function using the agent's domain knowledge and their solutions will be sent to the CS Function.

If during its problem solving activities the SLSSP Function discovers the need for supplementary data, it may generate Supplementary Data Request messages to its superior agent and dispatch them via the Domain - KDL Translation (DKT) Function. In due course, the required data will be provided by the superior agent via the KDTMR Function, stored in the agent's fact base and then used to continue the reasoning.

In accordance with the limited co-operation concept, the solutions of locally unsolvable sub-problems will be delegated to other agents in the Consulting Network by the Solve Locally Unsolvable Sub-Problems (SLUSP) Function. Delegation requires that the SLUSP Function interact with one layer of subordinate agents. The process will be driven by incoming messages arriving from the KDTMR Function plus outgoing messages sent to the DKT Function. An important aspect of the SLUSP Function is that it will establish sessions with and manage subordinate agents and handle their requests for supplementary data. The internals of the SLUSP Function will be discussed in more detail below.

We recognize that the identification of locally unsolvable sub-problems may entail some amendments to the problem solving plans created by the CP Function. These must be forwarded to the CS Function and integrated with the prevailing version of the plan.

The Combine Solutions (CS) Function will be responsible for synthesizing answers to the given problem using the hierarchical problem solving plan originally formulated by the CP Function and amended by the SLSSP Function. The synthesis process consists of fitting the solutions to all of the component sub-problems into the plan structure, irrespective of where they were solved, completing the reasoning process and producing a single answer for the superior (i.e. tasking) agent. Should the agent receive an explanation request from the superior agent, the CS Function will be responsible for producing an explanation. This will be initially done by passing on the explanation request to all of the subordinate agents. When their responses start arriving, the CS Function will place their explanation text into the structure of the plan and then generate a single composite explanation message to the superior agent.

As discussed previously in the Manage Problem Solving Node (MPSN) Function, the SLSSP, SLUSP and the CS function will send their internal status to the MPSN Function to aid in the efficient management of nodal computing resources.

We wish to again emphasize the difference between the implementation and functional architectures of a system. As we have seen, the functions described in the Solve Problem Function in fact span the building blocks of a typical KBS: the planner, scheduler, inference engine, explanation generator, blackboard, etc. These have not been made visible in this decomposition process because they relate to a specific implementation and would hide the functional interrelationships between Consulting Network functions.

6.2.6 Level BDD - The Solve Locally Unsolvable Sub-Problems Function

The purpose of the Solve Locally Unsolvable Sub-Problems (SLUSP) Function is to solve the given problem by means of the reasoning resources residing within other Network agents. See Figure 6-7. This entails the delegation of each of the locally unsolvable sub-problems to the most capable agent in the Consulting Network and the forwarding the solutions to the Combine Solutions Function residing within the Solve Problem (SP) Function. The typical process for each sub-problem is described below.

The first step is to derive a Request For Capability (RFC) message to all agents in the Network. An RFC message consists of a QTC encapsulated in certain message protocol elements. This process will be carried out by the Generate RFC Message (GRM) Function and will follow the procedures described in Sections 4.3.6 for QTC generation and in 5.2.2 for granularity matching. We note that the incoming lo-

Figure 6-7: Level BDD - The Solve Locally Unsolvable Sub-Problem (SLUSP) Function

cally unsolvable sub-problem statement is expressed in the domain-specific representation. Therefore, the QTC will be derived using domain vocabulary terms. On completion, the RFC message will be dispatched to the Domain - KDL Translation (DKL) Function residing within the Solve Problem (SP) Function. This initiates the process of broadcasting the RFC message to all agents in the Consulting Network.

All messages sent to the SLUSP Function will arrive via the KDTMR Function. The Produce LIL (PL) Function will capture all RFC Response messages from network agents and will construct a Likelihood List (LIL) on the basis of which the reasoning task will be assigned to a single agent.

This LIL produced by the CR Function will be sent to the Manage Subordinate Agents (MSA) Function. The MSA Function will carry out a built-in strategy for the task assignment process. Subject to further research, one of the strategies discussed in Section 4.3.4 will be used. If required, further negotiation messages can be exchanged between potential task targets and the MSA Function. Ultimately, the MSA Function will send a Session Request message to the targeted agent followed by a tasking message containing a description of the sub-problem to be solved. The targeted agent will respond with a Session Grant message and the targeted agent will commence problem solving.

The MSA Function will respond to requests for supplementary data from its subordinate agents, and forward any requests to superior agents via the DKT Function. When the required data is supplied by the superior agent, the MSA Function will forward it to the agent which requested it. On receipt of a Session End message, the MSA Function will dispatch Session End messages to all subordinate agents which it controls, thereby releasing their resources for other uses.

The SLUSP Function will report its internal status to the Solve Problem (SP) Function as an aid in the efficient management of the problem solving node. Of special interest are occurrences of GRM, PL and MSA Function idleness and also when the MSA Function terminates its subordinate agents on receipt of the Session End command.

6.3 FUNCTIONAL DECOMPOSITION SUMMARY

The above hierarchical functional decomposition of a Consulting Network provides a framework for the understanding of the interactions among the underlying processes described in Chapters 3 and 4. It describes the functions of the user, the Network Access Nodes and the Problem Solving Agent Nodes. It clearly shows that under the limited co-operation concept, the above entities may readily interact with each other in order to carry out the reasoning tasks assigned to the Consulting Network by a user. The processes described allow the Network to readily support multiple simultaneous users and problems, thereby increasing the Network's potential utilization. Most important of all, mechanisms have been defined which are capable of bridging agent heterogeneity with the objective of knowledge domain fusion.

We have purposefully avoided discussing the handling of error conditions which may arise during Network operation. To do so would have introduced an additional level of complexity which would have confused the understanding of the basic Network processes. We do however recognize that error handling is an important system design issue which must precede any attempts to implement the described functions.

The above framework does not in any way constitute an implementation architecture for Consulting Networks. SADT functions seldom map efficiently onto Buhr Diagrams, Data Flow Diagrams, system structure charts or other design paradigms. To derive an actual design would require the solution of all technical problems and research areas mentioned previously. This constitutes an effort which is beyond the scope of this thesis.

CHAPTER 7

THESIS SUMMARY

7.1 CONTRIBUTIONS TO DAI RESEARCH

This thesis presents several contributions to the field of Distributed Artificial Intelligence. It has been shown that current Problem Solving Networks (PSNs) are unable to deal with multiple general problems and are more suitable for solving very specific single problems which have a natural spatial distribution of processing, control and data. In our work we have de-emphasized the importance of maximizing network coherence and performance because they lead to very complex agent interaction mechanisms as in current PSNs. Instead we have focused on general problem solving as the key objective.

A new type of PSN has been proposed called a Consulting Network, composed of heterogeneous problem solving agents and capable of sharing knowledge among themselves, much like a team of human experts. General knowledge sharing makes possible the functional fusion of multiple disparate knowledge domains, and allows the Network to perform general problem solving in various fields of expertise. The Consulting Network concept is based on active interaction involving the exchange of messages. Communications must conform to the OSI model to ensure maximum connectivity between agents over a wide range of computing platforms and communications media.

A simplified agent interaction process called limited co-operation has been proposed, based on the client-server model. It limits the duration and span of interactions, and distributes the responsibility for planning, problem solving and control

among a hierarchy of agents, and is driven by a hierarchical decomposition of Network problems. Under this approach, agents lacking expertise on a specific topic may delegate reasoning tasks to other, more capable agents. Task assignment through negotiation predefines the agent interactions and removes the need for prediction, which has been a particularly difficult objective to achieve in previously tried interaction mechanisms. Agents are required to co-operate only when requested to do so, and only for the duration of a single problem solving session. For each tasking agent, this limited mode of co-operation reduces the span of command and interactions to a restricted number of subordinates.

It has been shown how agents with multi-tasking capabilities can readily overcome two major shortcomings of current PSNs, namely very limited interaction with users during the problem solving process and an inability to solve more than one problem at a time.

An important result of this work has been the description of mechanisms to bridge agent heterogeneity, wherein agents have disparate computing environments, knowledge representation forms, reasoning techniques, vocabularies and semantics. We recognized that agent heterogeneity is an unavoidable characteristic of distributed general problem solvers, as it arises naturally from the variety of ways in which different domains of knowledge must be represented. To circumvent the heterogeneity problem, we have proposed and described the high level attributes of a Knowledge Based System Discourse Language (KDL) which functions as a "lingua franca" throughout the Consulting Network, allowing disparate agents to interact during problem solving and exchange findings.

A highly effective sub-problem distribution process is essential to the success of problem solving through limited co-operation. We applied topically oriented semantic nets (TSNs) as a conceptual tool to create a catalogue of the domain metaknowledge of problem solving agents; organizing and storing topics on which agents had expertise. We showed how topical keywords could be extracted directly from problem statements could then be used to search through TSNs and identify the most capable agents in the Network to whom to delegate reasoning tasks. To study the idea further, we developed a recursive TSN search and score algorithm using Pascal and measured its performance on a PC/AT. With a 20 node TSN implemented as a static m-way tree, and 5 term QTCs, the approximate time to compute a normalized match score was 0.051 seconds.

Two different techniques were proposed for compiling lists of qualified agents: network organizational knowledge (NOK) and self-knowledge (SK). Both methods were mathematically analysed and simulated at a high level to gain a qualitative appreciation of their relative performance and merits. Little difference was noted in the overall load on agents operating under either approach. However, the SK based approach proved to be very much faster than the NOK approach in deriving likelihood lists (LILs), albeit at the expense of increased communication load on the Network. The SK approach was also perceived to be more beneficial because it simplifies the problem of configuration control among Network agents, thereby promoting the goal of an open architecture of distributed knowledge sources.

Finally, we carried out a detailed, hierarchical, functional decomposition of a Consulting Network using the graphics oriented Structured Analysis Design Technique (SADT) methodology. The resulting functional architecture provides a good

framework for understanding the operation of Consulting Networks. It shows how the various mechanisms must fit together and interact to achieve the objectives of Consulting Networks. We concluded that functionally, the concept of Consulting Networks is logical, consistent and reasonably complete. As such, there is a very strong likelihood that in the near future, with additional research, Consulting Networks could be successfully implemented on the basis of this functional model.

7.2 LIMITATIONS OF CONSULTING NETWORKS

Despite their potential for functionally fusing disparate knowledge domains the Consulting Network concept currently has certain limitations of which both the would-be implementer and the user must be aware. Specifically, the approach places some restrictions on the type of problems which could be solved through the Network.

The concept depends entirely on agents being able to hierarchically decompose problems at least one level down. This restriction, for example, prevents the solution of problems which cannot be hierarchically structured, or problems with multiple start and end points. Because very many real world problems do in fact have a natural hierarchical structuring, we anticipate that Consulting Networks will be capable of a useful level of general problem solving despite this constraint.

Consulting Networks do not directly support iterative problem solving among agents. This is not a limitation of Consulting Networks per se, but rather it depends on the problem domains and reasoning processes within individual Network agents. We note that iterative respecification has been successfully implemented in some distributed hierarchical problem solvers [Verrilli 87]. We anticipate that if desired, similar capabilities can also be introduced into Consulting Networks.

Problems which are ill defined will cause difficulties for Consulting Networks. This is not a unique problem, because even monolithic KBSs must be given suitable metaknowledge to tackle ill defined tasks. Nevertheless, Network Access Nodes must ensure that problem statements received from users are clear and well defined before they attempt to solve them. The automatic recognition of poor problem definition is a highly important process requiring more study.

The Consulting Network concept cannot handle non-monotonic reasoning across the Network unless non-monotonicity is explicitly embedded within the strategic knowledge of agents. This is due to the hierarchical nature of the problem solving under limited co-operation. Tasking agents are responsible for providing their subordinates with all of the parameters or data needed to solve the given problem. Individual agents certainly may perform non-monotonic reasoning locally. However, for each problem statement an agent will supply only a single answer to his superior irrespective of how it was derived. Once the answer has been given, the agent will not be aware of developments elsewhere in the Network which may have a bearing on the problem just solved. He will not be given any opportunity to revise or retract the answer just given unless, as part of the same problem solving session, the superior agent tasks him with the same problem but with changed inputs.

Consulting Networks should never be asked to supply the user with a "knowledge dump" on a given subject. Aside from the problem of interpreting and understanding the resulting stream of knowledge sources, on many common subjects the volume of information may be far too large to be casually transmitted over the Network.

A final limitation of Consulting Networks is that their task distribution performance depends largely on the ability of agents to quickly search through TSNs. If done strictly in software, in large scale Consulting Networks with many simultaneous problem solving sessions, the search process could bog down the system to an unacceptable level of performance. Clearly, the performance of Consulting Networks need to be investigated further.

7.3 CONCLUSIONS

Based on our work, we would like to make the following observations and conclusions:

- 1) If the identified problem areas can be solved, the Consulting Network concept shows excellent promise of being able to functionally unify multiple discrete and disparate knowledge bases so that the end result is functionally indistinguishable from a single monolithic knowledge base hosted within a single computer. The result will be a powerful reasoning engine capable of solving a wide range of real life problems. The most important design objective of Consulting Networks is to maintain an open architecture. Therefore, the construction of knowledge sources and KBSs to participate in Consulting Networks may spawn a knowledge engineering industry on an unprecedented scale. As a precursor to the construction of a proof-of-concept testbed, the Consulting Network concept should be investigated further, starting with the search for solutions to the technical problems identified in Section 7.4;
- 2) The limited co-operation approach described herein, is appropriate only for flexible problem solving situations where the problem set is not known a priori. However, if a Network is to perform a fixed set of functions, then limited co-operation is simply not required and the Network should be implemented along the lines of Takenouchi's "hard wired" problem solving strategy [Takenouchi 87]. This will create a very inflexible but highly

coherent system. Every agent requiring the reasoning services of another agent will know beforehand exactly to which agent to delegate specific reasoning tasks. Therefore, the Network agents will function in a lockstep manner;

- 3) The self-knowledge based approach to Likelihood List derivation is highly attractive because it makes good use of parallel computation in pattern matching of Query Topical Chains and TSNs. It also simplifies the management of TSN information across the PSN. Its main failing however, is the high processing load on agents due to the arrival rate of RFC messages, forcing them to repeatedly perform TSN searches. This may constitute a significant burden on individual agents and reduce the amount of time available to perform useful computing work.

According to Shastri, semantic networks may be encoded as connectionist hardware networks [Shastri 87a]. Such networks have been demonstrated as being able to solve pattern matching problems such as TSN searches in extremely short times which are proportional to the depth of the conceptual hierarchy. Shastri's work offers the opportunity to build connectionist hardware co-processors for Consulting Network agents in order to greatly speed up the TSN search and score process while off-loading their CPUs. This approach is thought to be quite feasible because the domains of expertise of established network agents should be relatively invariant, making the TSN search process amenable to implementation in hardware. If success-

ful, connectionist TSN search co-processors would greatly increase the attractiveness of the self-knowledge approach to LIL derivation.

- 4) The single biggest challenge in constructing Consulting Networks is linguistic. A powerful, expressive and robust NLI is required to interface users with a Consulting Network. However, the characteristics of the NLI are dependent on advances in NLP and NLU. As discussed in Sections 3.2.3 and 3.2.4, their current state of development may not yet allow the implementation of an NLI which is suitable for general problem solving. Linguistics is also a crucial issue in the process of knowledge sharing among agents. The design of the KDL proposed in Section 3.2.4.5 is of the same order of difficulty as the creation of a highly expressive and flexible natural language but without semantic ambiguities. Without solutions to these problems, Consulting Networks cannot achieve their full potential.

7.4 SUGGESTIONS FOR FURTHER RESEARCH

The Consulting Network concept appears to be an extremely rich area of follow-on research as evidenced by the identification in previous Sections of numerous problem areas requiring further investigation. Their complexity precluded further analysis as part of this thesis. However, their resolution is a necessary precursor to the construction of a testbed Consulting Network. We have briefly summarized the most significant ones below:

- 1) Design of a natural language interface through which users and Network Access Nodes would jointly formulate valid statements of problems for the Network to solve;
- 2) Determine strategies for decomposing problem statements into a series of sub-problems comprising the next lower level of problems in the problem solving hierarchy;
- 3) Determine the conditions under which an agent should automatically ask for reasoning help from other agents in the Network;
- 4) Investigate knowledge classification schemes to be used in the construction of Topically Oriented Semantic Nets (TSNs) and Query Topical Chains (QTCs);
- 5) Investigate the derivation of QTCs from a grammatical analysis of problem statements expressed either in a human language or in a form internal to a KBS;

- 6) The investigation of strategies for task assignment when there are multiple potential targets available in the Likelihood List (LIL);
- 7) Design of the Knowledge Base System Discourse Language (KDL);
- 8) Design of a generic translator between the KDL and an agent's internal representation;
- 9) Design of a highly efficient search and score algorithm capable of performing fast pattern matching between QTCs and TSNs;
- 10) Investigation of optimum weighting factor functions for computing TSN search scores, so that a good compromise is achieved between specificity and generality of agents' knowledge;
- 11) The design and construction of a connectionist co-processor implementation of the TSN search and score algorithm as discussed in item 3 of Section 7.3.
- 12) Study the role of the relevant existing and emerging standards in the development of Consulting Networks.

APPENDIX A

**LISTINGS OF THE TSN SEARCH & SCORE
ALGORITHM**

```

PROCEDURE TSN_SEARCH (Topic : TopicType;
                        Child_Ptr : CPTType;
                        QTC : QTCType;
                        QTC_Length : Integer;
                        Weights : WTSType;
                        Max_QTC_Length: Integer;
                        VAR Norm_Max_Score: Real);

```

{ This procedure searches through a TSN and performs pattern matching of the terms in the given QTC against the topical information stored in the TSN. It computes a normalized score which indicates how knowledgeable is the given KBS on the subject matter expressed by the QTC. A score value of 1.0 indicates perfect capability, while 0.0 indicates no capability.

{ DEFINITIONS OF VARIABLES & PARAMETERS:

NOTE: These are applicable to all procedures in the algorithm

Branch_List	List of topical keywords extracted from any given branch of the TSN
Branch_List_Length	Number of terms in the Branch_List array
Branch_Score	The score computed from matching the QTC against the Branch_List array
Child_Ptr	A 2-D array of child node pointers
Max_QTC_Length	Maximum allowable length of the QTC
Max_Score	The maximum branch score achieved during the TSN search
Node_Ident	Identifier of a specific TSN node}
Norm_Max_Score	Normalized maximum branch score
Perfect_Score	The maximum possible score if a match is found for each term in the QTC
QTC	Query Topical Chain array
QTC_Length	Number of terms in the QTC array
Topic	Array of the topical keywords stored in the nodes of the TSN

TSN_Order	The maximum number of child nodes in any TSNnode
Weights	Array of weighting factors used to compute the contribution to the branch score of branch list matches with any QTC term

VAR

Max_Score : Real;
 Perfect_Score : Real;
 I : Integer;
 Branch_List_Length : Integer;

BEGIN

{INITIALIZE VARIABLES}

Perfect_Score := 0.0;

Max_Score := 0.0;

Branch_List_Length := 0;

{SEARCH FOR ALL LEAF NODES IN THE TSN AND GET THE
 MAXIMUM BRANCH SCORE}

FIND_LEAF (Node_Ident[1], Topic, Child_Ptr, QTC, QTC_Length,
 Weights, Max_QTC_Length, Branch_List_Length, Max_Score);

{COMPUTE THE MAXIMUM POSSIBLE SCORE TO NORMALIZE THE
 MAXIMUM BRANCH SCORE}

FOR I := 1 TO QTC_Length DO BEGIN

 Perfect_Score := Perfect_Score + Weights[I]

END; {FOR}

{COMPUTE THE NORMALIZED TSN SCORE}

Norm_Max_Score := Max_Score/Perfect_Score

END; {TSN_SEARCH}

```

PROCEDURE FIND_LEAF  (Node_Ident : Integer;
                      Topic : TopicType;
                      Child_Ptr : CPtrType;
                      QTC : QTCType;
                      QTC_Length: Integer;
                      Weights : WTSType;
                      Max_QTC_Length : Integer;
                      VAR Branch_List_Length : Integer;
                      VAR Max_Score : Real);

```

{ This procedure searches through all branches of a TSN searching for leaf nodes. For each branch, it constructs a list of the topics contained in each branch of the TSN. On detection of a leaf node, the list of TSN branch topics is matched against the given QTC and a branch score is computed. If this score exceeds the maximum branch score, then the maximum branch score is updated. }

```

VAR
  Leaf_Node : Boolean;
  Child_Ptr_Index : Integer;
  Branch_Score : Real;

```

```

BEGIN
  { ON ENTRY INTO TSN NODE, INITIALIZE VARIABLES }
  Leaf_Node := TRUE;
  Child_Ptr_Index := 1;
  Branch_List_Length := Branch_List_Length + 1;
  Branch_List[Branch_List_Length] := Topic[Node_Ident];

  { LOOK FOR LEAF NODES IN ALL CHILD BRANCHES }
  WHILE (Child_Ptr_Index <= TSN_Order) AND
        (Child_Ptr[Node_Ident, Child_Ptr_Index] <> 0) DO BEGIN

```

```

    Leaf_Node := FALSE;
    Find_Leaf(Child_Ptr[Node_Ident, Child_Ptr_Index],
              Topic, Child_Ptr, QTC, QTC_Length, Weights,
              Max_QTC_Length, Branch_List_Length, Max_Score);
    Child_Ptr_Index := Child_Ptr_Index + 1
  END; {WHILE}

  {WHEN A LEAF NODE HAS BEEN FOUND, MATCH THE QTC AGAINST
  THE BRANCH TOPICS AND COMPUTE THE BRANCH SCORE. IF THIS
  IS A NEW HIGH, THEN REDEFINE THE MAXIMUM BRANCH SCORE}
  IF Leaf_Node THEN BEGIN
    Match (Branch_List, Branch_List_Length, QTC, QTC_Length,
           Weights, Max_QTC_Length, Branch_Score);
    IF Branch_Score > Max_Score THEN
      Max_Score := Branch_Score
  END; {IF}

  {DECREMENT THE BRANCH LIST LENGTH WHEN LEAVING NODE}
  Branch_List_Length := Branch_List_Length - 1;

END {FIND_LEAF};

```

```

PROCEDURE MATCH (Branch_List : BLType;
                  Branch_List_Length : Integer;
                  QTC : QTCType;
                  QTC_Length : Integer;
                  Weights : WTSType;
                  Max_QTC_Length : Integer;
                  VAR Branch_Score : Real);

```

{ This procedure examines the terms in the branch list and compares them with the QTC. For each branch list term that matches a QTC term, the branch score is incremented by an amount determined by the corresponding weighting factor. The result is a branch score. }

```

VAR

```

```

    QTC_Term_Start_Ptr : Integer;
    QTC_Term_Ctr : Integer;
    BL_Term_Ctr : Integer;

```

```

BEGIN

```

```

    { INITIALIZE VARIABLES }

```

```

    Branch_Score := 0.0;

```

```

    QTC_Term_Start_Ptr := QTC_Length;

```

```

    { EXAMINE THE ENTIRE BRANCH LIST }

```

```

    FOR BL_Term_Ctr := 1 TO Branch_List_Length DO BEGIN

```

```

        { CHECK REMAINING TERMS IN THE QTC }

```

```

        FOR QTC_Term_Ctr := QTC_Term_Start_Ptr DOWNTO 1 DO BEGIN

```

```
{WHEN A MATCH HAS BEEN FOUND, RECOMPUTE THE  
BRANCH SCORE AND SET THE POINTER TO THE REMAINING  
PART OF THE QTC}  
IF (QTC[QTC_Term_Ctr] = Branch_List[BL_Term_Ctr]) THEN BEGIN  
    Branch_Score := Branch_Score + Weights[QTC_Term_Ctr];  
    QTC_Term_Start_Ptr := QTC_Term_Ctr - 1;  
    QTC_Term_Ctr := 1  
END; {IF}
```

```
END; {FOR}
```

```
END; {FOR}
```

```
END; {MATCH}
```

REFERENCES

LIST OF REFERENCES

[Boettcher 87]

K. Boettcher, D. Perschbacher and C. Wessel, "Coordination of Distributed Agents in Tactical Situations", Proc IEEE 1987 NAECON, 1987, pp. 1421-1426.

[Burt 86]

T.E. Burt, "Combat System Integration in the US Navy", International Defence Review 1/1986, pp. 31-36.

[Clancey 85a]

William J. Clancey, "Heuristic Classification", Artificial Intelligence, 1985, 27, pp. 215-251.

[Clancey 85b]

William J. Clancey, "Acquiring, Representing and Evaluating a Competence Model of Diagnostic Strategy", Stanford Knowledge Systems Laboratory, KSI 84-2, 1985.

[Corkill 83]

D.D. Corkill and V.R. Lesser, "The Use of Meta-Level Control for Coordination in a Distributed Problem Solving Network", Proc of the 8-th IJCAI, 1983, pp. 748-756.

[Corkill 86]

D.D. Corkill, K.Q. Gallagher and K.E. Murray, "GBB: A Generic Blackboard Development System", Proc AAAI 86, 1986, pp. 1008-1014.

[DATAPRO 89]

"ISO Reference Model for Open Systems Interconnection (OSI)", DATAPRO Reports on Computer Communications, McGraw-Hill, Inc., Feb 1989, pp. C07-500-301.

[Davis 80]

R. Davis, "Metarules: Reasoning About Control", Artificial Intelligence, 1980, 15, pp. 179-222.

[Davis 83]

R. Davis and R.G. Smith, "Negotiation as a Metaphor for Distributed Problem Solving", Artificial Intelligence 20, 1983, pp. 63 - 109.

[Durfee 84]

E.H. Durfee, D.D. Corkill and V.R. Lesser, "Distributing a Distributed Problem Solving Network Simulator", Proc Real-Time Systems Symposium, 1984, pp. 237- 246.

[Durfee 85]

E.H. Durfee, V.R. Lesser and D.D. Corkill, "Increasing Coherence in a Distributed Problem Solving Network", Proc IJCAI 1985, pp. 1025- 1030.

[Durfee 87a]

E.H. Durfee, V. Lesser and D.D. Corkill, "Cooperation Through Communication in a Distributed Problem Solving Network", Distributed Artificial Intelligence, Morgan Kaufmann Publishers, Inc., 1987, pp. 29-58.

[Durfee 87b]

E.H. Durfee, V.R. Lesser and D. Corkill, "Coherent Cooperation Among Communicating Problem Solvers", IEEE Trans. on Computers, Vol C-36, No 11, November 1987, pp. 1275-1291.

[Ernst 69]

G. Ernst and A. Newell, "GPS: A Case Study in Generality and Problem Solving", New York: Academic, 1969.

[Feigenbaum 84]

E.A. Feigenbaum and P. McCorduck, "The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World", The New American Library Inc., 1984

[Fikes 71]

R.E Fikes and N.J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", Artificial Intelligence 2, 1971, pp. 189-208.

[Fikes 72]

R.E. Fikes, P.E. Hart and N.J. Nilsson, "Learning and Executing Generalized Robot Plans", Artificial Intelligence 3(4), 1972, pp. 251-288.

[Frenzel 87]

L.E. Frenzel, Jr., "Crash Course In Artificial Intelligence and Expert Systems", H.W. Sams & Co., Indianapolis IN, 1987.

[Frost 86]

R. Frost, "Introduction to Knowledge Base Systems", Macmillan Publishing Company, New York, 1986.

[Genesereth 86]

M.R. Genesereth, M.L. Ginsberg and J.S. Rosenschein, "Cooperation Without Communication", Proc of AAAI 86, pp. 51-57.

[Goldberg 85]

M. Goldberg, D.G. Goodenough, M. Alvo and G.M. Karam, "A Hierarchical Expert System for Updating Forestry Maps with Landsat Data", Proc IEEE, Vol 73, No 6, June 1985, pp. 1054-1063.

[de Haan 86]

J. de Haan and L.K. Schubert, "Inference in a Topically Oriented Semantic Net", Proc AAAI 86, 1986, pp. 334-338.

[Hammond 86]

K. Hammond, "CHEF: A Model of Case-Based Planning", Proc of Fifth National Conf on AI, 1986, pp. 267-271.

[Hayes-Roth 83]

F. Hayes-Roth, D.A. Waterman and D.B. Lenat, "Building Expert Systems", Addison-Wesley Publishing Co., Inc., Reading, Mass., 1983.

[Harmon 86]

S.Y. Harmon, "Sensor Data Fusion Through a Distributed Blackboard", Proc of the 1986 International Conf. on Robotics and Automation, San Francisco, April 1986, pp. 1449-1454.

[Harmon 87a]

S.Y. Harmon, "The Ground Surveillance Robot (GSR): An Autonomous Vehicle Designed to Transit Unknown Terrain", IEEE Journal of Robotics & Automation, June 1987, pp. 266-279.

[Harmon 87b]

S.Y. Harmon, "Implementation of Complex Robot Subsystems on Distributed Computing Resources", NATO ASI Series Vol B33, Machine Intelligence and Knowledge Engineering for Robotic Applications, Springer-Verlag, Berlin Heidelberg 1987, pp. 406-435.

[Hendrix 87]

G.G. Hendrix & B.A. Walker, "The Intelligent Assistant", Byte, Vol 12, No 14, December 1987, pp 251-258.

[Hillenbrand 89]

B. Hillenbrand, "Trying to Decipher Babel", Time Magazine, July 24, 1989, pp. 47.

[Horowitz 82]

E. Horowitz and S. Sahni, "Fundamentals of Data Structures", Computer Science Press, Inc., Rockville, Md., 1982.

[Huhns 87a]

M.N. Huhns, "Foreword", Distributed Artificial Intelligence, Morgan Kaufmann Publishers, Inc., 1987, pp. v-ix

[Huhns 87b]

M.N. Huhns, "DAI for Document Retrieval: The MINDS Project", Distributed Artificial Intelligence, Morgan Kaufmann Publishers, Inc., 1987, pp. 249-283.

[Jakobson 86]

G. Jakobson, C. Lafond, E. Nyberg and G. Piatetsky-Shapiro, "An Intelligent Database Assistant", IEEE Expert, Summer 1986, pp. 65-79.

[Jarke 86]

M. Jarke, "Knowledge Sharing and Negotiation Support in Multiperson Decision Support Systems", Decision Support Systems (Netherlands), Vol. 2, No. 1, 1986, pp. 93-102.

[Kaye 87]

A.R. Kaye and G.M. Karam, "Cooperating Knowledge-Based Assistants for the Office", ACM Trans on Office Information Systems, Vol 5, No 4, October 1987, pp. 297-326.

[McDermott 77]

D.V. McDermott, 'Flexibility and Efficiency in a Computer Program for Designing Circuits', Technical Report, AI Memo 402, AI Laboratory, MIT 1977.

[Nilsson 71]

N.J. Nilsson, "Problem Solving Methods in Artificial Intelligence", McGraw Hill Book Company, 1971, pp. 85.

[Nilsson 80]

N.J. Nilsson, "Principles of Artificial Intelligence", Morgan Kaufmann Publishers, Inc., 1980.

[Obermeier 87]

K.K. Obermeier, "Natural Language Processing", Byte, Vol 12, No 14, December 1987, pp 225-232.

[Patrick 86]

E.A. Patrick and J.M. Fattu, "Artificial Intelligence With Statistical Pattern Recognition", Prentice-Hall, Inc., Englewood Cliffs NJ, 1986, pp. 83.

[Retelle 86]

J.P. Retelle, Jr. and M. Kaul, "The Pilot's Associate - Aerospace Application of Artificial Intelligence", Signal, June 1986, pp. 100-105.

[Ross 77]

D.T. Ross, "Structured Analysis (SA): A Language for Communicating Ideas", IEEE Trans on Software Engineering, Jan 77, pp. 107.

[Schubert 76]

L.K. Schubert, "Extending the Expressive Power of Semantic Networks", Artificial Intelligence 7, 1976, pp. 163-198.

[Shastri 87a]

L. Shastri, "A Connectionist Encoding of Semantic Networks", Distributed Artificial Intelligence, Morgan Kaufmann Publishers, Inc., 1987, pp. 177-202.

[Shastri 87b]

L. Shastri, "Massive Parallelism in Artificial Intelligence", Applied Optics, Vol 26, No. 10, May 15, 1987, pp. 1829-1844.

[Shaw 87]

D.E. Shaw, "On the Range of Applicability of an Artificial Intelligence Machine", Artificial Intelligence, Vol. 32 No. 2, May 1987, pp. 151-172.

[Shumaker 86]

R.P. Shumaker and J. Franklin, "Artificial Intelligence in Military Applications", Signal, June 1986, pp. 29-48.

[Smith 86]

J.M. Smith, "Expert Database Systems: A Database Perspective", Expert Database Systems, The Benjamin/Cummings Publishing Company, Inc., 1986, pp. 3-15.

[Sridharan 87]

N.S. Sridharan, "1986 Workshop on Distributed AI", AI Magazine, Fall 1987, pp. 75-85.

[Stallings 85]

W. Stalings, "Data and Computer Communications", Macmillan Publishing Co., New York, 1985, pp. 385.

[Sternberg 82]

R.J. Sternberg, "Reasoning, Problem Solving and Intelligence", Handbook of Human Intelligence, Cambridge MA, Cambridge University Press, 1982.

[Swartout 88]

W. Swartout, "DARPA Santa Cruz Workshop on Planning", AI Magazine, Summer 1988, pp. 115-131.

[Takenouchi 87]

H. Takenouchi and Y. Iwashita, "An Integrated Knowledge Representation Scheme for Expert Systems", Expert Systems, February 1987, Vol 4, No 1, pp. 38-43.

[Verrilli 87]

R.J. Verrilli, K.L. Meunier, J.R. Dixon and M.K. Simmons, "Iterative Respecification Management: A Model for Problem-Solving Networks in Mechanical Design", Proc ASME: Computers in Engineering Conf 1987, pp. 103-112.

[Waltz 86]

E.L. Waltz and D.M. Buede, "Data Fusion and Decision Support for Command and Control", Trans on Systems, Man and Cybernetics, Vol. SMC-16, No. 6, Nov-Dec 1986, pp. 895-879.

[Waterman 86]

D.A. Waterman, "A Guide to Expert Systems", Addison-Wesley, 1986.

[Wielinga 86]

B.J. Wielinga and J.A. Breuker, "Models of Expertise", ECAI 1986, pp. 306-318.

[Winston 77]

P. Winston, "Artificial Intelligence", Addison-Wesley Publishing Co., Reading, MA, 1977.

[Youngers 86]

M. Youngers, J. Franklin, C.L. Carmody and A. Meyrowitz, "Improving C³: The Potential of Artificial Intelligence", Signal, June 1986, pp. 51-55.

END

2020/9 1

FIN