

SEL  
G41  
76.9  
N38  
S46  
1989

**SEMANTIC  
STRUCTURES**  
Advances in  
Natural Language  
Processing

Edited by  
**David L. Waltz**  
Thinking Machines Corporation and  
Brandeis University

**LEA** LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS  
1989 Hillsdale, New Jersey Hove and London

106619707  
1989

1989

# Contents

PREFACE vii

## 1. KNOWLEDGE INTERACTIONS AND INTEGRATED PARSING FOR NARRATIVE COMPREHENSION 1

Michael G. Dyer

- Introduction 1
- Foundations in Comprehension 10
- Multiple Knowledge Sources in BORIS 23
- Generalizing Scripts with Multiple Perspectives 24
- Processes of Comprehension 30
- Memory Modification During Question Answering 36
- Theory of Affect 40
- Thematic Abstraction Units 44
- Future Research 50
- Conclusions 52
- Acknowledgments 54
- References 54

## 2. EVENT CONCEPT COHERENCE 57

Richard Alterman

- Introduction 57
- How NEXUS Works 59
- In the Context of Some Other Representation Schemes 71
- Summary and Conclusions 83
- Acknowledgments 85
- References 86

Copyright © 1989 by Lawrence Erlbaum Associates, Inc.  
All rights reserved. No part of this book may be reproduced in any form, by photostat, microfilm, retrieval system, or any other means, without the prior written permission of the publisher.

Lawrence Erlbaum Associates, Inc., Publishers  
365 Broadway  
Hillsdale, New Jersey 07642

### Library of Congress Cataloging-in-Publication Data

Semantic structures.

Bibliography: p.

Includes index.

1. Natural language processing (Computer science)
2. Programming languages (Electronic computers)—Semantics. I. Waltz, David L.

QA76.9.N38S46 1989 006.3'5 88-33420

ISBN 0-89859-817-6

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

<b>3. LEARNING WORD MEANINGS FROM EXAMPLES</b>	<b>89</b>
Robert C. Berwick	
Introduction	89
The Acquisition Procedure	95
Implementation and Examples	104
Learning Nouns and Class Hierarchies	109
Conclusions	117
References	117
Appendix 1: The Stories	119
Appendix 2: The Matching Function	120
<b>4. AN INTRODUCTION TO PLOT UNITS</b>	<b>125</b>
Wendy G. Lehaert and Cynthia L. Loiseau	
Understanding and Representation	125
The Plot Unit Representational System	128
Summarization from Plot Units	139
Evidence for the Psychological Validity of Plot Units	152
An Overall View	157
References	163
<b>5. NATURAL LANGUAGE DESCRIPTION OF TIME-VARYING SCENES</b>	<b>167</b>
Bernd Neumann	
Overview	167
Representing the Scheme	170
Events	173
Verbalization	177
Composing a Description	181
Discussion	192
Acknowledgments	195
References	195
Appendix A	197
Appendix B	202
<b>AUTHOR INDEX</b>	<b>207</b>
<b>SUBJECT INDEX</b>	<b>213</b>

## Preface

I want to build a machine that will be proud of us.  
—Danny Hillis

Natural language (NL) understanding is central to the goals of artificial intelligence. Any truly intelligent machine must be capable of carrying on a conversation. It would be difficult to ascertain that a machine was "proud of us" unless it could tell us, both directly, and indirectly via paraphrases under questioning. It is not essential that conversations be in spoken language—they could be in sign language or via a keyboard and screen—but the machine must be able to converse.

Dialogue—in particular clarification dialogue—is essential if we are to avoid disasters caused by the misunderstanding of robots and other intelligent interactive systems of the future, designed to carry out actions on our behalf. One-way communication won't do. Machines that speak (or print) the contents of their own memory are already with us, but no one considers them to be intelligent. We would be very impressed by a mute computer that could carry out linguistic instructions with perfect reliability, but I do not think that it is possible to build such a machine, for the same reasons that

human assistants can not work satisfactorily (except for trivial or repetitive tasks) unless they are able to engage in discussions about their assignments. Computer programs that can describe their surroundings will also need to be conversationalists: To be convincingly intelligent, such programs must be able to perceive and interpret the events around them with respect to the goals they permit or prevent; moreover, they must be capable of interpreting events from different perspectives as goals shift. A machine that always describes the world from a single perspective will not seem particularly intelligent or interesting.

This book is an interim report on the grand enterprise of devising a machine that can use natural language as fluently as a human. What has really been accomplished over the nearly 40 years since this goal was first formulated (in Turing's famous test)? What obstacles still need to be overcome in order to achieve this goal? This preface offers a broad perspective on these questions and uses this perspective as the setting for the five chapters that comprise the rest of this volume.

## KNOWLEDGE

First, and perhaps most critically, intelligent conversational programs must have knowledge. Natural conversation depends on a vast body of shared knowledge of word meanings, concepts, conventions, and strategies for expressing these concepts in language. Conversations may also require quite specific knowledge on the conversational partner's goals, propensities, and history, as well as knowledge of the situational context, potentially including knowledge of the perceptual world. Sentences within a conversation are not, in general, self-contained. They express meaning only to the extent that a listener can be convinced that he or she has been able to identify the right topic of conversation and decipher what comment the speaker has made about this topic and why.

Exactly what knowledge must a system have? How can such knowledge be collected, represented, and learned? I will return to this topic. Much of this book, including the chapters by Dyer, Lehnert, and Loisel, Alterman, and Neumann, is concerned with knowledge—especially knowledge about relatively large units of natural language, such as stories or conversations, as opposed to knowledge of phrases or sentences. Berwick's chapter shows how word meanings can be learned from text.

## Architectures and Algorithms

This book is also concerned with mechanisms and architectures for natural language processing (NLP). In some areas of artificial intelligence—mathematical reasoning, problem solving, vision, and games, for example—it may be possible to build systems that perform at or above human levels, yet operate by different principles. However, using language truly fluently and appropriately requires a system to make the same inferences, to be surprised, amused, or confused at the same points, to be reminded of the same shared knowledge and to make the same judgements about listener's reactions to its utterances that a human language user would. Thus, I believe it is very unlikely that we can build a machine that understands and generates natural language without modeling human language processing mechanisms, and memory structure, in some detail.

What are these mechanisms and structures like? In the years since I first began to study artificial intelligence (AI), I have gone from a belief that natural language understanding is possible on any universal computer by direct modeling of human behavior and the stream of consciousness—modeling of The Knowledge Level (Newell, 1981)—to a belief that, while we probably won't need to model the full detail of the brain's neural action and connectivity, we will probably need to use massively parallel, Society of Mind-like models (Minsky, 1987) that embody some deep correspondences with the "modules" and architecture of cognition.<sup>1</sup>

The need for such models is inspired in part by what is known about the brain, but important influences also come from introspection and from observations of human behavior: The sheer speed of language understanding—in sharp contrast with the slowness of individual neurons—argues strongly for a massively parallel system where different aspects of processing (e.g. phonetic, syntactic, se-

<sup>1</sup>In the Society of Mind model, systems are comprised of very large numbers of independent agents, each of which connects to and communicates with a number of other agents in the system. Each agent is active whenever the concept, percept, or feature it represents is recognized, suggested, or hypothesized. Agents become active through the influence of other agents and/or sensory inputs. Each communication link between agents carries very simple messages, encoding the level of activation of the sending agent. Groups of interconnected agents are organized via hierarchy and other structuring patterns into "agencies," modules that are responsible for classes of functions and actions.

memory operations) are handled by different mechanisms (Pollack, 1985). Both the triggering of particular "free" and memory items and the selection of appropriate sentences strongly suggest the joint action of a great many autonomous influences, many of which exert very subtle influences.

An important consequence of Society of Mind models is that different meanings are represented by different distributed patterns of activity across the same set of agents; that is, meaning is represented "in place." In contrast, traditional AI (and von Neumann machine) processing paradigms assume a central processing unit that selects and processes items from memory, one at a time; items must be moved out of their places of storage to be processed, and only one (or a very small number) of memory items can be considered at any given instant. Thus, if the Society of Mind view is correct, the vast amount of attention expended in AI on heuristics, knowledge representation schemes, and algorithms for finding the most important memory items is largely irrelevant to actual natural language processing—no correlates will be found to exist in the brain for such functions and structures. However, knowledge representation remains a central issue—perhaps the central issue in natural language processing. Only the portions of the field concerned with efficient implementations of search on von Neumann machines can be scrapped.

The Society of Mind theory, along with massively parallel connectionist or "neural net" architectures (Rumelhart & McClelland, 1986; Waltz & Feldman, 1988) have played important roles in stimulating the development of massively parallel computers, such as the Connection Machine (Hillis, 1985). These architectures follow in the tradition of perceptrons (Minsky & Papert, 1969; Rosenblatt, 1962) and are inspired, in part, by neurophysiology (although they so far bear little resemblance to actual nervous systems). Natural language researchers who want to use such models and hardware have been required to dramatically rethink their algorithms and data structures. Parallel hardware seems destined to become much more important in the future, not so much because it matches the mechanisms and structure of NL processing in the brain, but because these architectures are the only ones we know that can scale to arbitrarily large size and power, and because such architectures promise to be far more cost-effective than serial machines. Natural language algorithms that match or can be adapted to parallel hardware will dominate serial implementations, and only the nonparallelizable remnants will still be carried out on serial hardware. Fortunately,

most semantically-oriented processing, including nearly all the research described in this book, can be relatively easily parallelized: For example, individual word senses can be assigned to processors or connectionist nodes, or such concepts can be represented as distributed patterns on parallel networks (Rumelhart & McClelland, 1986). Special-purpose speech processing chips, based on analog, rather than digital, processing, have already been constructed (Hopfield, 1988). However, syntactic processing has so far resisted significant parallelization, though there are promising methods for aspects of syntactic analysis (Church, 1988; Ejerhed, 1988; Selman & Hirst, 1985; Small, 1983; Waltz & Pollack, 1985).

### NATURAL LANGUAGE KNOWLEDGE

There are two kinds of science: physics and stamp collecting.  
—E. Rutherford

For the last 35 years, natural language processing has more closely resembled stamp collecting than physics. An early hope and expectation (Simon, 1965) was that a relatively small number of basic powerful principles would suffice to achieve artificial intelligence. For example, much early effort centered on heuristic search in AI and transformational grammars in linguistics; both these are more akin to physics than stamp collecting. It now seems clear that intelligence in general and natural language in particular involves vast numbers of facts, procedures, rules, heuristics, and so forth, none of which is in itself very significant. Moreover, it is unlikely that this body of material can be reduced to a small set of primitives or operations.

To the extent that phenomena reminiscent of physics have been discovered in the study of natural language, they have concerned primarily the syntactic regularities of language. Noam Chomsky has argued that the structure of any language can be characterized by a quite small number of innate parameters. For example, parameters may specify such items as the ordinary order of sentences: SVO (Subject Verb Object) as in English versus SOV as in German; or the standard order of nominals and modifiers: (adjective\* nominal) as in English versus (nominal adjective\*) as in French.<sup>2</sup> Each individual

<sup>2</sup>(adjective\*) represents zero, one, or many adjectives concatenated together, as in the phrase: "the big white run-down house".

learns such aspects of his native language by setting the appropriate values for these parameters.

There has also been some physics-like success in the area called "lexical semantics" (Pustejovsky, in press; Jackendoff, 1983). Lexical semantics postulates that all language is based on a very small number of primitive semantic objects (e.g., states, actions, and state-changes) and relations between them (e.g., temporal succession, agentive actions, and causality). Any verb is associated with a particular structure of these objects and relations, along with a set of particular mandatory and optional features that constrain the ways in which the word can participate in sentences. Although there are some regularities in the lexicon, most words are idiosyncratic and must be learned one by one.

Words alone comprise a large body of data. The Random House Unabridged Dictionary (Stein & Urdang, 1981) defines roughly 300,000 words. Because, on the average, each word has 3.7 senses, on the order of a full-fledged NLP system must be able to handle about 1,000,000 distinct meanings. (A reasonable guess for the vocabulary of an educated adult is probably about half this number. These estimates omit many phrases and proper nouns that could be considered lexical entities.) All these estimates contain many phrases and proper nouns. Each word sense may in turn require the storage of a large amount of knowledge. For example, consider a verb such as "eat." In order to use "eat" appropriately, we need to know when its use is literal, metaphorical but familiar, novel and metaphorical, inappropriate, and so forth. A program will need to understand that "the man ate hay" is unusual, whereas "the mouse ate the cheese" is a cliché. We know about hundreds of kinds of eaters, human and animal, and hundreds of foods, leading to tens of thousands of possible eater/food combinations. Although "eat" may be a particularly difficult example, even if we only store on the average of 10 facts about each word sense, this would still lead to as many as 5 million basic word facts.

Moreover, we each know a great many particular facts about history, geography, politics, entertainment, literature, sports, games, industry, and so forth, which are important to include if we are to be able to understand or generate appropriate linguistic references for natural conversation. We can estimate an upper bound on the amount of such knowledge: Assume that we were to learn one chunk of 100 bits every second for 16 hours every day for 20 years; we would learn 50 billion bits (Hillis, 1988). This amount of information does not seem to present an obstacle for hardware—a current Connection Machine can store 4 billion bits of information in its fast

memory, and denser memory chips should allow 64 billion bits of fast storage by the early 1990s. However, collecting, representing, and using this information seems a very daunting task indeed.

How could we get this information into a machine? One possible answer is to gather a team of "stamp collectors," and there is a real-life example of this approach: Doug Lenat and a number of coworkers are building CYC, a system into which they are placing, by hand-coding, all the information in an encyclopedia (Lenat et al., 1986). Lenat estimates this will require 10 years and cost \$50 million. Lenat is using a frame-based representation language (Brachman & Levesque, 1983; Minsky, 1975) to encode the information. He hopes that his group can use information in the system to accelerate the learning rate of the system as the project comes closer to completion. However, representing basic knowledge of language itself seems substantially harder than representing facts; we do not know what structures and operations are appropriate for expressing linguistic and lexical knowledge, and there is no organized repository of knowledge. In addition, finding and using pieces of encyclopedic knowledge to help encode new encyclopedia entries is a much easier task than using this knowledge to support the understanding of general conversations or text passages. A very different technological alternative to hand-coding is discussed in the last section of this preface.

Learning (or knowledge acquisition) thus presents a set of key problems that must be solved if we are to build natural conversational systems. Each chapter of this book bears in some way on this issue. Berwick's chapter deals with learning directly; Neumann shows methods whereby perceptual inputs can generate linguistic descriptions and appropriate corresponding memory structures; Dyer, Alterman, and Lehnert and Loisele each deal primarily with representation of knowledge, and thus implicitly constrain the types of linguistic learning that will be needed to build a full-scale system. Each of this last group also offers novel insights into the evolving of meaning structures, in parallel, in a Society of Mind-like understanding system.

## UNITS OF KNOWLEDGE

What are the natural units of knowledge that constitute and support natural language understanding? I will argue here that such units span a large range of sizes and types. At the highest level, entire topics can be "primed" by context. For example, the sentence "the group lacked an identity" will quite effortlessly evoke one in-

terpretation if its context is mathematics; quite another if its context is psychology. (See [Waltz & Pollack, 1985] and [Bookman, 1988] for some proposed mechanisms to handle such phenomena.) At the finest levels language can interact with perceptual and mental imaging processes. This is necessary to explain how we judge the meanings of descriptions, such as the shades of difference between "over," "above," "just above," "somewhat above," "far above," "almost on top of," and so forth. Such distinctions can force crucially different inferences: "Just above the roof of the house" suggests quite different inferences from "the shelf just above the desk." The issues are explored in Neumann's chapter in this book. (See also [Herskovits, 1985; Langacker, 1986; Maddox, 1988; Talmy, 1988; Waltz, 1981a, 1981b, 1982; Waltz & Boggess, 1979]). Between these extremes lie the types of units that introspectively seem most critical: stories, sentences (or predications), phrases, and words. Most of the chapters in this book are concerned with these units: Berwick with words and phrases; Dyer, Lehnert, and Loisel with stories; and Alterman with words, sentences, and sequences of sentences.

## WORDS

During the 1970s and part of the 1980s, much of the AI natural language processing world used "primitives" and structures of primitives to represent word (and sentence) meanings. Proposed primitives include those of "conceptual dependency" (Schank, 1975; see also Dyer's chapter in this volume for a history and explanation of CD); "preference semantics" (Wilks, 1975); and the LNR system (Norman, Rumelhart, et al, 1975). Primitives are special sets of irreducible word-like units, chosen so that, ideally, structures built out of them can represent all possible linguistic meanings. In practice, primitives have actually proved limited, though less so than one might imagine. (See the illuminating discussion in Winograd, 1978, as well as Wilks, 1975 and Jackendoff, 1976. Because many different words can collapse into the same primitive or structure of primitives, primitive-based representations express phrases naturally. However, they may not allow a system to express important distinctions in text. Thus, for example, in CD "eat," "nibble at," "wolf," "smoke" (as a cigarette), "inject" (as a drug), and many other words are all represented by a single primitive—INGEST. Although one would want to be able to make some inferences that are common to all these cases (i.e., some substance is at first outside

and later inside the agent's body), INGEST is too crude to allow importantly different inferences. For example, "wolf" suggests that its agent was hungry or in a hurry, whereas "nibble at" suggests its agent's disinterest in or dislike of the food (Waltz, 1982).

As mentioned earlier, most natural language researchers have abandoned primitives and now make the assumption that each word is distinct, requiring its own, substantially more elaborate, structure (Jackendoff, 1983; Pustejovsky, 1988; Small, 1983). Berwick's chapter in this volume deals with the acquisition of relatively elaborate structures to represent word meanings. In the work reported in Alterman's chapter, each word is represented in terms of the network of other words (or concepts) closely related to it (e.g., subsequences of actions, situations that hold before or after actions, goals or reasons for actions, etc.). I expect this current trend will continue toward finer-grain representations, specialized for each word, and linked into networks (Brachman & Levesque, 1983; Sowa, 1984; Woods, 1975).

## STORIES

Stories seem uniquely important because they are apparently the mind's preferred unit of memory: Stories are important carriers of cultural wisdom in a form that seems compelling and easily digested. It is instructive to compare their palatability with the oppressive pedantry of lectures of prescribed rules of action (Brewer & Lichtenstein, 1982; Dorfman, 1985; Dorfman & Waltz, 1981; Wilensky, 1983). Proverbs, a particularly valued form of language, encapsulate stories (Schank, 1986). I will not elaborate further here on the representation of stories, because they are so extensively and well treated in the chapters by Dyer, and Lehnert and Loisel.

## AUTOMATING THE BUILDING OF FULL-SCALE METHODOLOGIES FOR NATURAL LANGUAGE SYSTEMS

For most of its history, AI has been concerned with "toy problems." This is true of the chapters in this volume: All deal with specific natural language passages, chosen to be representative but still limited. Scaling up presents difficulties: At the two extremes of the spectrum, one can hand-code (as in CYC [Lenat, et al, 1986]), or one can use methods to automatically build NLP systems. To date there

have been very few practical applications of natural language processing,<sup>3</sup> and this fact has dampened the enthusiasm of funding agencies and companies that support research in this area.<sup>4</sup> Fortunately, there are signs that this situation can be improved by strategically merging AI/NLP and Information Retrieval (IR) technologies. Over the last several years, I have been involved with such AI/IR research; it offers novel opportunities both for automatic learning, and for building systems that have immediate practical value.

Series of experiments and discoveries led researchers at Thinking Machines, most notably Craig Stanfill and Brewster Kahle, to devise a document retrieval system that works in parallel on the Connection Machine (Stanfill & Kahle, 1986; Stanfill, 1988). The resulting system, now a commercial product, provides a high quality search through a clever interface that can be used effectively by a computer-naïve person after only about 5 minutes of training. The basic idea is this: A database of documents (e.g., news articles, abstracts, books, etc.) is distributed to each of the many processors of a Connection Machine (if documents are 2K bytes long, each processor can hold about 12 documents). The user types a few words (a question, description, or list of terms will do) and a carriage return; the terms are broadcast to all the processors in parallel along with a numerical "weight" indicating the importance of each term.<sup>5</sup> Each processor compares the terms with the contents of its documents, and adds the "weight" to the score for each document in which the term occurs. The headlines for the documents with the highest total scores are then displayed to the user. The user can view the text of each of these documents by clicking a mouse button while pointing to the headline. When a user sees a document (or paragraph of a document)

<sup>3</sup>Commercially available natural language systems include INTELLECT (Harris, 1977), a product of Artificial Intelligence Corp., which allows users to get information from a database by typing questions in English; NL Menu (Tennant, 1983), a product of Texas Instruments, that lets users build natural language "front ends" for programs—one uses a mouse to select the words for each sentence from a set of menu choices, insuring that the system will "understand" any user input; Q and A, a natural language database front end from Semantic Inc., and EPISTLE, a grammar correcting system from IBM.

<sup>4</sup>DARPA, the Defense Advanced Research Projects Agency, expects to spend only about \$1.4 million in fiscal 1989 on natural language processing, compared to over \$7 million the previous year. Much of funding has been switched to support speech processing research.

<sup>5</sup>Term weights are assigned automatically by a program that pre-processes the text and updates the database. The number of occurrences of each term is saved, and weights computed proportional to the negative log of the probability of occurrences of each term.

that answers their request, the user can mark the document "good" by pointing and clicking the mouse. The system collects all the terms from all the documents marked "good" along with the initial words the user typed, and repeats the search process described above, but now with many more terms (often several hundred). Each search requires less than a second, even on databases up to 10 CBytes. This method, called "relevance feedback," generally produces substantially better search than is possible with Boolean search systems (Blair & Maron, 1985).<sup>6</sup>

This system offers two interesting opportunities to extend natural language processing systems:

1. It is highly desirable to add natural language pre- and post-processing to the existing system, to improve its performance, and to extend its capabilities. For example, we are building recognizers that can find, label, and store lists of terms that refer to company names, geographic locations, names of persons, etc. Ultimately, this will help users to ask and obtain answers to questions that would be very difficult to phrase as Boolean queries. For example: Typing "Earnings reports for New England utility companies" would expand to "Earnings reports for Maine, New Hampshire, Vermont, Massachusetts, Connecticut, Rhode Island utilities, power companies, electric companies, Edison, power, light . . ." In addition, natural language processing systems will allow us to post-process retrieved documents, to filter out irrelevant articles, and thus improve the performance of the system from the user's point of view.

2. The retrieval system itself can be adapted to extract phrase sentence and paragraph "templates" or patterns, in order to aid the building of recognizers for particular topics or types of stories. Such processing can provide empirical data on language usage that would be very difficult to find or invent any other way, leading to "dictionaries" of multi-word and multi-sentence language patterns and to FRUMP-like systems (DeJong, 1982) with broad subject coverage.<sup>7</sup>

<sup>6</sup>A commercial software system based on these ideas has been developed at Thinking Machines and purchased by Dow Jones, Inc.

<sup>7</sup>FRUMP recognized about 70 types of stylized or "scriptural" newspaper articles: auto accidents, diplomatic visits, burglaries, etc. Frump contained quite specific, hand-coded sets of patterns to match each of the types of information needed to fill in the values in its scripts for each story-type.



The most attractive part of this effort is that our system is immediately useful, and thus can pay for the research on its own augmentation. Other related research, using dictionaries or thesauruses, has become popular in recent years. Some striking successes have been achieved by Ken Church and coworkers at AT&T Bell Laboratories (Church, 1988; Ejerhed, 1988) using the augmented "Brown Corpus" (Kucera & Francis, 1982). The Brown Corpus consists of one million words of text, chosen to represent a wide range of text types and styles (newspaper and magazine articles, books on history, economics, etc.). It was "augmented" by Kucera and Francis by assigning each word in the corpus to one of about 450 classes, covering standard grammatical categories (noun, verb, adjective) but also including substantially finer distinctions (e.g., noun-agent of sentence; verb-complements of particular types). Church collected statistics on the probabilities that various words would follow particular other word (or category) combinations. This system has been used to judge the most likely categories for words in novel text taken from news-wire sources. Success rates for Church's system are in the range of 98-99%, much higher than for the best syntactic parsers (in the range of 33% [Salton, 1988, personal communication]).

All these current lines of research emphasize breadth of coverage, rather than depth of coverage, and are thus complementary to the goals of traditional AI NL processing research. All present attractive alternatives to hand-coding (Lenat, et al, 1986). And all can be used to accelerate the research into deep processing. I believe these general approaches will have great importance in the ultimate story of the achieving of truly intelligent systems.

## REFERENCES

- Blair, D., & Maron, M. (1985). An evaluation of retrieval effectiveness for a full-text document retrieval system. *Communications of the ACM*, 28, 289-299.
- Bookman, L. (1988). A connectionist scheme for modeling context. In D. Touretzky, C. Hinton, & T. Sejnowski (Eds.), *Proceedings of the 1988 Connectionist Models Summer School*. Los Altos, CA: Morgan Kaufmann.
- Brachman, R., & Levesque, H. (1983). *Readings in knowledge representation*. Los Alto, CA: Morgan Kaufman Publishers.
- Brewer, W.F., & Lichtenstein, E.H. (1982). Stories are to entertain: a structural-affect theory of stories. *Journal of Pragmatics*, 6, 473-486.
- Church, K. (1988). A stochastic parts program and noun phrase parser for unrestricted text. Unpublished manuscript, AT&T Bell Labs, Murray Hill, NJ.

- Dorfman, M. (1985). A model for understanding the points of stories. *Proceedings of the 7th Annual Conference of the Cognitive Science Society*, (pp. 262-63). Irvine, CA.
- Dejong, G. (1982). An Overview of the FRUMP System. In W. Lehnert & M. Ringle (Eds.), *Strategies for natural language processing*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Ejerhed, E. (1988). Finding clauses in unrestricted text by stochastic and finitary methods. Unpublished manuscript, AT&T Bell Labs, Murray Hill, NJ.
- Harris, L. (1977). Robot: A full performance natural language data base query system. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 903-904).
- Herskovits, A. (1985). Semantics and pragmatics of locative expression. *Cognitive Science*, 9, 341-378.
- Hillis, D. (1985). *The connection machine*. Cambridge, MA: MIT Press.
- Hillis, D. (1988). Intelligence as an emergent behavior; or, The songs of eden. *Daedalus, (Journal of the American Academy of Arts and Sciences)*, 175-189.
- Hopfield, J. (1988). Remarks to AAAI workshop on how slow components can think so fast. Stanford University, March 22-24.
- Jackendoff, R. (1983). *Semantics and cognition*. Cambridge, MA: MIT Press.
- Jackendoff, R. (1976). Toward an explanatory semantic representation. *Linguistic Inquiry*, 7, 89-150.
- Kucera, H., & Francis, W. (1982). *Frequency analysis of english usage*. Boston: Houghton Mifflin Company.
- Langacker, R.W. (1986). An introduction to cognitive grammar. *Cognitive Science*, 10, 1-40.
- Lenat, C., Prakash, M., & Shepherd, M. (1986). CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottleneck. *AI Magazine*, 4, 65-85.
- Minsky, M. (1975). A framework for representing knowledge. In P. Winston (Ed.), *The psychology of computer vision* (pp. 211-277). New York: McGraw-Hill.
- Maddox, A.B. (1988). *Perceptual interpretation and the transformation of events into language*. Doctoral dissertation, University of Illinois, Urbana, Illinois.
- Miller, G.A., & Johnson-Laird, P. (1976). *Language and perception*. Cambridge, MA: Harvard University Press.
- Minsky, M. (1987). *The society of mind*. New York: Simon & Schuster.
- Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. Cambridge, MA: MIT Press.
- Newell, R. (1981). The knowledge level. *AI Magazine*, 2, 1-20.
- Norman, D.A., Rumelhart, D.E., & LNR Research Group (1975). *Explorations in cognition*. San Francisco: W.H. Freeman.
- Pustejovsky, J. (in press). *Semantics and the lexicon*. Dordrecht, Holland: Kluwer.
- Rosenblatt, F. (1962). Strategic approaches to the study of brain models. In

- H. von Foerster (Ed.), *Principles of Self Organization*. Elmsford, NY: Pergamon Press.
- Rumelhart, D., & McClelland, J., et al. (1986). *Parallel distributed processing*. Cambridge, MA: MIT Press.
- Selman, B., & Hirst, G. (1985). A rule-based connectionist parsing system. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 212-219). Irvine, CA.
- Schank, R.C. (1986). *Explanation patterns: understanding mechanically and creatively*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Schank, R. (1975). *Conceptual information processing*. Amsterdam: North Holland.
- Simon, H. (1985). *The sciences of the artificial*. Cambridge, MA: MIT Press.
- Small, S. (1983). *Word expert parsing: A theory of distributed word-based natural language understanding*. Technical Report 954. Department of Computer Science, University of Maryland.
- Sowa, J. (1984). *Conceptual structures*. Reading, MA: Addison-Wesley.
- Staufil, C. (1988). *Parallel computing for information retrieval: recent developments*. Thinking Machines Corporation Technical Report #DR88-1.
- Stanfill, C., & Kahle, B. (1986). *Parallel free text search on the connection machine system*. *Communications of the ACM*, 29, 12.
- Stein, J.M., & Urdang, L. (1981). *The Random House dictionary of the English language*. NY: Random House.
- Talmy, L. (1988). *Force dynamics in language and cognition*. *Cognitive Science*, 12, 49-100.
- Tannant, H. (1984). *Menu-based natural language understanding*. AFIPS Proceedings of the National Computer Conference, Las Vegas, NV, 629-635.
- Waltz, D., & Feldman, J. (Eds.) (1988). *Connectionist models and their implementations*. Norwood, NJ: Ablex.
- Waltz, D.L. (1982). *Event shape diagrams*. *Proceedings of the National Conference on Artificial Intelligence* (pp. 84-87). Pittsburgh, PA.
- Waltz, D.L. (1981a). *Toward a detailed model of processing for language describing the physical world*. *Proceedings of International Joint Conference on Artificial Intelligence*, 81, (pp. 1-6). Vancouver, B.C.
- Waltz, D.L. (1981b). *Generating and understanding scene descriptions*. In Joshi, Sag, & Webber (Eds.), *Elements of discourse understanding* (pp. 266-282). Cambridge, UK: Cambridge University Press.
- Waltz, D.L., & Boggess, L. (1979). *Visual analog representations for natural language understanding*. *Proceedings 6th International Joint Conference on Artificial Intelligence* (pp. 926-934). Tokyo, Japan.
- Waltz, D.L., & Dorfman, M. (1983). *The holes in points*. *The Behavioral and Brain Sciences*, 6, 612-613.
- Waltz, D.L. & Pollack, J.B. (1985). *Massively parallel parsing: a strongly interactive model of natural language interpretation*. *Cognitive Science* 9, 1, 51-74. Also in Waltz & Feldman (1988), 181-204.
- Wilensky, R. (1983). *Story grammars versus story points*. *The Behavioral and Brain Sciences*, 6, 579-623.

- Wilks, Y.A. (1975). *An intelligent analyzer and understander of English*. *Communications of the ACM*, 18, 264-274.
- Winograd, T. (1978). *On primitives, prototypes, and other semantic anomalies*. In D. Waltz (Ed.), *TINLAP-2: Theoretical Issues in Natural Language Processing-2* (pp. 25-32). NY: Association for Computational Machinery.
- Woods, W.A. (1975). *What's in a link: foundations for semantic networks*. In D. Bobrow & A. Collins, *Representation and Understanding: Studies in Cognitive Science*, NY: Academic Press, 35-82. Also in Brachman & Levesque (1983), 217-241.

## CHAPTER 3

# Learning Word Meanings from Examples

ROBERT C. BERWICK  
Artificial Intelligence Laboratory  
Massachusetts Institute of Technology

### INTRODUCTION

This chapter describes an experimental computer program that deduces the meaning of novel verbs from the context of story descriptions. The key idea is a variation on Winston's (Winston, 1975) program that learned the structural descriptions of block world scenes. Instead of learning descriptions of toy block assemblies like ARCH and TOWER, the word-learning program acquires frame-based descriptions of English verbs like MURDER or DONATE. The program works by assuming that similar verbs will play similar causal roles in common story plots. For example, ASSASSINATE is like MURDER because both MURDER and ASSASSINATE cause similar things to happen and are caused by similar patterns and events. Intuitively, we learn about a new verb like ASSASSINATE as a kind of family resemblance variation on a core verb like MURDER. The program works in a similar way. Syntactic constraints derived from the parsing of story plots are used to drive an analogical matching procedure. Analogical matching gives a way to compare descriptions of known words to unknown words. The "meaning" of a new verb is learned by matching part of the causal network description of a story précis

containing the unknown word to a set of such descriptions derived from similar stories that contain only known words. The best match forges an assignment between objects and relations such that the unknown verb is matched to a known verb. The causal network surrounding the unknown item is then used as a scaffolding to construct a network representing the meaning of the novel word in a particular context. In some ways, the model is similar in spirit to that of Granger (1977), who also extracted context-based descriptions to acquire meaning descriptions of novel words. As we see, this way of describing word meaning has several advantages over definitional approaches.

A second aim of this research is to explore the interaction between syntax and semantics in learning. The word-learning program is embedded into a larger system that can acquire new syntactic rules for English, as described in Berwick (1979, 1980, 1982, 1985). The word-learning component uses the larger system's determination of the syntactic category of a new word and its predicate-argument structure. These last two abilities are based on the X theory of Jackendoff (1977) and a theory of syntax that assumes a strong principle of lexical transparency (roughly, that the semantic argument structure of a verb appears at all levels of representation). Here, syntax means simply the grammatical form of sentences, whereas semantics refers to such notions as case relationships (Agent, Affected Object), as well as the causal description language used for word matching itself. Although some have emphasized either syntax or semantics as a key to word learning, the position taken here is that either may serve as a constraint on the other. It is the interaction between syntax and semantics that drives word learning. In some cases, syntax helps semantics. We see several examples of this here, where prior knowledge of sentence structure plays a vital role in figuring out the candidate words for a causally based match. Recently, Landau and Gleitman (1985) have independently confirmed this theoretical finding by studying a blind child's acquisition of verb meanings. They found that syntactic constraints were crucial for successful acquisition of verbs such as *look at* or *see*. Language acquisition involves the complex interplay between these two sources of constraint.

Third, this chapter suggests a concrete computational model for the acquisition of word prototypes from positive examples. In addition, it also suggests a way of automatically generating A K O ("a-kind-of") hierarchies from positive examples. Importantly, this method follows a natural formal and empirically justified constraint on the evolution of categorizations first explored by Keil (1979).

The moral of this chapter is that constraints make learning possible. A learning theory can be built only if one has a good representation of what it is that is learned, and some idea of the constraints on that target state. For our theory, the constraints include a restricted vocabulary of thematic relations (a "case system") plus a causal description language. The hardest part of constructing a theory of word learning is figuring out just how to represent the "meaning" of a word. Fortunately, a variety of linguistic constraints can help us here. The second moral is that learning does not seem possible unless one almost already knows what is to be learned—a principle of "incremental learning." Whether all learning abides by these principles is not certain, but it seems true that so far all successful artificial intelligence (AI) learning programs have followed them.

Our first job, then, is to describe the representation for word meaning to be used by the learning program. Before we do so, however, it would be best to give an example of the kind of competence we want our word-learning system to display. As mentioned, a key underlying assumption is that the meaning of a word is determined by the role it plays in a causal network description of an event, and that similar words are those that play similar roles in the description of similar events. Consider the following scenario:

Suppose we are given two versions of the story of Macbeth, one reporting that "Macbeth murders Duncan" and the other that "Macbeth assassinates Duncan." Further suppose that MURDER is a known word but not ASSASSINATE. We should conclude that ASSASSINATE is most like MURDER, because, comparing stories, it seems to us that ASSASSINATE plays the same role that MURDER does in the Macbeth plot. We should also conclude that ASSASSINATE has political overtones, because we note that the Macbeth story includes such relations as *Macbeth wants to be king* and *Macbeth becomes king*. Probing further, later stories should inform us that the uses of ASSASSINATE and MURDER are slightly different, because MURDER need not carry that political connotations that ASSASSINATE does. We should also be able to use the story of Hamlet to deduce the same kind of relationship between MURDER and ASSASSINATE.

Here is the actual input for the Macbeth story:

MA is a story about Macbeth Lady-macbeth Duncan and Macduff.  
Macbeth is an evil noble. Lady-macbeth is a greedy ambitious woman.  
Duncan is a king. Macduff is a noble.  
Lady-macbeth persuades Macbeth to want to be king because she wants him to be king. She wants him to be king because she is greedy.

Now suppose that a number of other stories have been previously analyzed and stored in causal network form (e.g., Macbeth, Hamlet, Julius Caesar, and the Taming of the Shrew, as discussed in Winston, 1980), and that none of these previous plot summaries used the word *assassinate*. (Alternatively, the *Macbeth* story could be considered unknown, as long as other examples of murders, say, in *Hamlet*, were already analyzed.)

A causal network representation can be built for each of these stories. For the old *Macbeth* and *Hamlet* stories, this network will be nearly identical to the network built previously, but with some key differences that flow from the lack of understanding of *assassinate*. Still, as we see, syntactic constraints permit the entire network to be built, because the connections between objects and actions are actually syntactic.

The learning procedure then takes the causal network descriptions of each of the candidate stories and runs an analogy matching program, pairing objects and relations in the new story against objects and relations in each of the candidates until the best match is obtained. Note that the system can use either similar stories for an attempted match or exactly the same story (but with unknown words).

This is a graph matching problem. It tries all possible combinations of nodes (objects) and links (labeled with verbs, prepositions, or CAUSE) until it finds one that lines up best with the old story. This best match will pair up corresponding objects and links in the network representation of the new *Macbeth* story and the network of one of the other stories. In this case, the best match we'd murder and *assassinate*, because these have the most causal links in common. This is the first step in forging a representation of the meaning of the new verb *assassinate*.

This is what we want our system to do; the rest of the chapter says just how it is done. In the first Section, following much recent work on concept acquisition, I discuss why dictionary definitions are inappropriate for this task. The next section follows with an outline of a causal description language that does seem more suited for representing verb meanings and a definition of semantic similarity. The third section outlines the syntactic constraints used by the acquisition program and their connection to semantic representations. A detailed discussion of the program implementation and some examples of the system in action is given in the fourth section. It also compares the current system to an earlier attack on this problem, by Granger (1977). The final section is a discussion of current limitations and some proposed extensions for the learning of Nouns.

She can convince him because Macbeth loves her and also because Macbeth is wimpy.

Macbeth assassinates Duncan with a knife. Macbeth assassinates Duncan because Macbeth wants to be king. Lady-macbeth becomes insane. She kills herself because she is insane.

Macduff becomes angry because the king was assassinated. Macduff kills Macbeth because Macbeth assassinated Duncan and because Macduff is loyal to Duncan.  
Remember MA.

Using the techniques described in Winston (1980) and Katz and Winston (1982) the system builds a causal network description of the story, as shown in Figure 3.1. This network is basically an object-oriented semantic network, with objects, agents, and qualities (and sometimes propositional attitudes like beliefs or desires) serving as the nodes in the network, and verbs serving as the links between nodes.

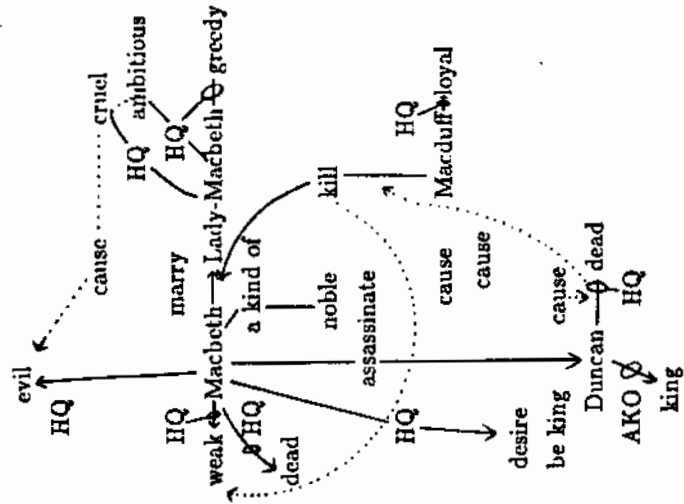


Figure 3.1. Causal networks describe stories

### THE INADEQUACY OF DEFINITIONS

If the key to any learning program is a good representation of what is learned, then we must have a good representation of word meanings before we can learn them. In fact, this has been the biggest stumbling block for word learning. What is the meaning of a word?

One common analysis has been *decompositional*. Words are viewed as chemical compounds, built out of constituent primitives. So for example, *dog* might be labeled *Animate*, *Four-legged*, *Mammal*. . . . This "word chemistry" is a classical technique that aims to account for our apparent "generative" ability with words—our ability to represent the meaning of a potentially unlimited number of words with finite means. Researchers as diverse as Schank and (much earlier) Katz and Fodor have embraced this method. For instance, as should be familiar, Schank takes the "meaning" of verbs as representable by a graphic language consisting of primitives like *TRANS* (physical movement) assembled into a picture of the relationship between objects in an event representation of a sentence.

The difficulties with a "pure" compositional account of word meanings are by now well known. There is a familiar skeptical tradition in 20th century philosophy of language, stemming in part from the later work of Wittgenstein, arguing that it is impossible to give definitions—necessary and sufficient conditions—for words. In a classical passage, Wittgenstein observed, for example, that there can be no definition of a *game*, because there is nothing held in common between board games, sports games, and the like. The most that can be said is that there is a kind of "family resemblance" between the members of the group of games. More recently, psychological arguments have been advanced that mitigate against definitional accounts of meaning. Fodor, Garrett, Walker, and Parkes (1980) pointed out that it is impossible to find "if-and-only-if" conditions for words, aside from examples of such words as jargon or kinship terms. (Note that AI programs for word acquisition have often focused on just these sorts of items, such as *bachelor*.) They also provide psycholinguistic evidence—based on reaction time tests—that feature decomposition into primitive elements is not carried out in on-line processing.

Whatever the status of these experiments and others like them (see, for example, Hayes-Roth and Hayes-Roth, 1977, who present similar evidence against decompositional theories), it is plain that "if-and-only-if" conditions for word definitions must be rejected. Definitions should admit exceptions and graded categorization judgments, as well as networks of family resemblances. What is the alternative? One approach that has been emerging out of work in AI and

### 3. LEARNING WORD MEANINGS FROM EXAMPLES

psychological analysis is that of *prototype theory*. Prototype theory holds that the criterial features of a word—say, *apple*—are described via some paradigm or core example. We picture an apple by conjuring up a familiar red object in our mind's eye. Even so, this prototypical apple admits exceptions and gradations in each of its characteristic qualities: the skin may be green, if the shape is right; the apple may be mis-shapen, if it is otherwise applelike; a huge apple is still an apple, and so on. In the traditional AI parlance, we may connect these various sorts of apples by means of *difference pointers* (Winston, 1975) indicating how and by how much the various members of the apple family deviate from core apple-ness.

Difference-pointer families avoid the pitfalls of necessary sufficient conditions because there are no necessary and sufficient conditions for any definitions. Indeed, there are no definitions at all, in the "if and only if" sense. Viewed this way, what we learn when we learn a new word is not some set of conditions, but a connection of some network of already-known words. This viewpoint is not novel, of course, being part of most network-based theories of semantic representation. What is new is a demonstration of just how this approach can be used for word learning.

At the same time, it is worth pointing out that these claims about word decomposition are controversial. An opposing, long-standing approach attempts to chop up word meanings, particularly those for verbs, into primitives. So for example, *kill* might be written as *CAUSE to die*. This is a tack favored by Gruber (1965), Schank (1973), and Jackendoff (1972), among others. In this chapter we straddle both sides of this difficult fence. Rejecting if-and-only-if conditions, we adopt a family resemblance model for meaning. At the same time, we decompose verbs into causal network diagrams.

### THE ACQUISITION PROCEDURE

#### The Causal Description Language

Having adopted a "family resemblance" model for meaning, we have still to say just what our network descriptors should be. As is well known, the choice of a semantic network vocabulary is a hazardous game. Ideally, one should follow empirical constraints on descriptors, but little is known about what these should be. Is color a basic element of our semantic descriptions? If so, what colors? What about geometric shapes? Such conditions can be multiplied endlessly. It would be useful to have criteria analogous to those for grammatical categories and features that would let us know that red is just as much a part of semantic descriptions as Noun is for syntactic

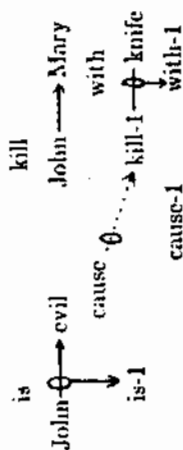


Figure 3.2. A causal network fragment

descriptions. The last section of this chapter explores one way of attacking these problems, based on work by Jackendoff (1977), but we defer the question for now.

Given that we do not know just what the right primitive elements should be, or even that there are primitive elements at all, in this work we have simply adopted the position that all elements are allowable as potential descriptors on our semantic network. The actual representation for the simple English stories in our database is called an *extensible-relation representation*. It was developed by Winston (1980).

This semantic net is object oriented, with agents, objects, and qualities serving as nodes and verbs tying them together. If more than one object is involved in an act, additional descriptions can be associated with the act-specifying relation. These additional descriptions are nodes that are related to other nodes. For example, the sentence *John killed Mary with a knife* because *John is evil* is depicted in Figure 3.2.

Note that in this scheme all individual words such as *knife* or *kill* are considered unanalyzable wholes. Nevertheless, it is still possible to relate one word to another word, by comparing the networks in which each is embedded.

### Causality

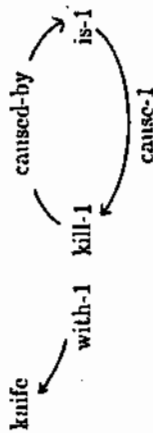
There is one more ingredient to add to our representational language that is a primitive, however. One of the key elements in describing a situation seems to be *causality*. To know what an event is about it is important to know what causes what and what is caused by what. Evidence for this view comes from a variety of sources. Developmental psychologists have discovered that even very young children are disposed to analyze complex mechanical scenes in causal terms, and they can do so correctly (Gelman & Gallistel, 1979). For example, a long row of standing dominoes can be successfully analyzed as capable of knocking open a jack-in-the-box, if the leftmost domino is

rapped with a rigid swinging stick long enough to reach it. Evidently quite young children know enough about causality to reason about even these complex situations.

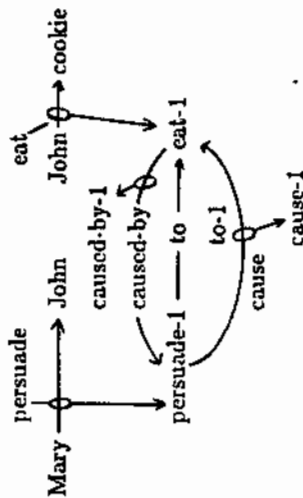
Then too, the ability to explain what happened depends on a causal description. It is no surprise that many AI programs that reason about physical processes rely on causal descriptions. Causality plays a prominent role in other representational languages for predicates, particularly in the work of Schank and Wilks. Finally, causality adds an implicational structure to an otherwise static description network. Without before-and-after links of some kind, it is impossible to describe a chain of events. Otherwise, words are just linked to each other without any directionality.

Given the central role of causality in the description of events, special cause links are included in the network description language. Because causes are so important, there is an extra mechanism, demons, associated with them. The following conventions for CAUSE demons are obeyed:

- If A causes B, then B is caused by A, where A can be people, relations, or objects and B can be other relations or acts. The caused-by links are automatically generated when causal relations are inserted in the database. In our previous example, there should be a caused-by link from kill-1 to is-1.



- If A persuades B to do C, then clearly, A causes C and C is caused by A. The appropriate links are added to the database. For example, *Mary persuaded John to eat a cookie* is pictured as:



are also used to make certain simple deductions as relations added to the database. For instance, if we add *Macbeth kills Duncan* to our database, a demon adds *Duncan is dead*. From a theoretical standpoint, these demons are simply meaning postulates, in Carnap's (1952) sense. But why does one add some set of meaning postulates rather than another? Why, for instance, add just "Romeo has-quality dead" rather than a whole list of additional deductions likely to be true if Romeo kills himself? The idea here is that the added deductions are the "simplest" meaning postulates associated with predicates, where "simplest" is a one-step deduction according to a common-sense language of description. In other words, folk psychological terms are used rather than a sophisticated language of scientific descriptions for depicting events: We say that Macbeth killed Duncan because Macbeth is evil, not because of some account averting to biochemical imbalances. To be sure, this way of limiting deduction is not perfect, but it at least aims to stay close to the deductions people make.

Although the AKO and causal-based representation has been a useful testbed, it is not without problems. We use a preexisting set of AKO descriptors, such as *king* or *knife*. Ideally, these too should be learned. Also, it is not clear that *cause* suffices to describe all verbal relationships. Stative verbs like *be* that describe changes in properties or motion verbs like *run* are not so easily cast in terms of causal interactions with objects. A cause-based matcher will have nothing to work with here, and so will fail. In the fourth section of this chapter we propose modifications to handle some of these difficulties.

### Thematic Role Structure and Meaning

In addition to the causal network description, there is one other component of word meaning used by the learning program. This is the linguistic notion of *thematic roles*, also called *case frames*. Thematic-role descriptions have been discussed in detail by Fillmore (1968) and Gruber (1965), but in fact are part of almost every linguistic theory. Intuitively, we think of a verb as requiring several thematic arguments that flesh out a picture of the event the verb describes. For example, *eat* takes at least three arguments: the eater, the thing eaten (sometimes tacit); and optionally the manner, instrument, or place of eating. For our purposes here, what is important is that particular thematic roles are ordinarily linked to specific grammatical positions in a particular language. In English, the subject position is canonically the "doer" of the action or the agent. Import-

tantly, the notion of subject is syntactic in English, because it is the first noun phrase under the sentence in a parse tree. Likewise, the object position canonically plays the *affected object* or *patient* role. This correspondence between form and meaning implies that thematic structure can be recovered from syntactic form, at least in clear cases.

Interestingly enough, possible form-meaning assignments vary parametrically within narrow limits. That is, although other languages (unlike English) do not link the subject syntactic position with the thematic role of agent, the allowed variations are quite limited. Linguistic analysis suggests a twofold division of the world's languages, into either the *accusative languages*, with a subject-agent and object-patient connection; or the *ergative languages*, with the reverse object-agent and subject-patient links. Once this pattern is fixed for a given language, it need not be relearned for individual verbs, although there may be exceptions to, for example, the generally accusative character of a language. In English, for instance, there is a systematic class of verbs whose subjects are affected objects rather than agents; this data and the table here are from Levin (1983).

- The glass broke.
- The puzzle disappeared.
- The book fell off the shelf.

This analysis carves up English verbs into a few classes according to their argument structure. Basically, there are two argument verbs with the accusative pattern; one argument verbs with a subject-agent link; and the ergative one argument verbs like those just mentioned:

2 arg. verbs	1 arg. verbs (Agent only)	1 arg. verbs (Patient only)
hit	talk	come
kick	cough	break <sup>1</sup>
push	wave	appear
give	sing	fall
take	sniff	go

The learning program exploits these regularities by using the form-meaning correspondence to guide network matching. No matter what the meaning of a novel verb, and unless there are specific syntactic clues that indicate otherwise, one can assume that the subject will be the agent and the object will be the patient. But this means that the subject-verb-object links so crucial for the meaning



tion can be partially built even for an unknown verb. Syntactic constraints are enough in most cases. Then, the network matching program can use this information to attempt to pair up only agents with other agents, patients with patients, and so forth. This filtering saves a good deal of effort. For example, if we have that "X kills Y," then naturally X and Y will wind up playing different causal roles later in the story. Given "W assassinates Z," it would thus be a waste of time to try to pair the person killed in one story with the assassinator in the other. Thematic filtering prevents this. Granger (1977) used a similar method within a very different semantic representation. He too found that thematic roles were a crucial factor in looking for a representation for a novel word. Granger did not use thematic roles directly, but instead looked at prepositional types and their noun phrase objects. In English, this amounts to the same thing as a thematic role. For example, *by* plus an animate noun phrase often corresponds to an agent thematic role. This would not work in a language where thematic roles are unmarked by prepositions. It also fails in cases where thematic role varies from its canonical prepositional assignment; for example, in the vase broke, *vase* is not the agent. This is one reason why it is better to work through a mediating syntactic theory that directly defines thematic roles, and then use these for filtering candidate matches.

With the basics of the syntax-semantics connection described, some details are in order. The techniques described in Berwick (1985), based on the Marcus (1980) parser, are used to syntactically analyze the input. Consider a sentence such as *Macbeth assassinates Duncan*. How can we assign the correct thematic roles in this case if *assassinate* is an unknown word?

First of all, we may assume that all normal English sentences are known to be in the form noun phrase—verb phrase. Unless there is evidence to the contrary, the parser will predict that a noun phrase (*Macbeth*) begins this sentence as well. With the NP disposed of, the parser now predicts that a verb phrase (VP) should be found. Given that English verb phrases must be headed by verbs, *assassinates* is forced to be a verb. In this way, syntax actually constrains the lexical category of the unknown word. (There are, of course, other possibilities. Suppose that an adverb intervened between NP and verb, as in, *Macbeth quickly dispatched Duncan*. In this case, given that *quickly* is known as an adverb, again a VP is predicted, and *assassinates* must be its head. Note that *Duncan* is assumed known as a noun, hence cannot be the verb heading the verb phrase.)

Finally, *Duncan* is parsed as an NP and noted as an object of the unknown verb, as it is by definition. Given the canonical rela-

tionship between thematic arguments and syntactic positions, we also know that the object is the patient and that *Macbeth* is the agent. So we already know quite a bit about *assassinates* without really understanding its meaning.

### Semantic Similarity and Matching

We are now ready to describe the matching criteria used for judging when two verbs are "close" in meaning. Two verbs are judged semantically identical if:

1. Their thematic role structures and selectional restrictions are the same;
2. The same objects are present in the causal representations of both verbs; and
3. Their causal links overlap exactly.

Selectional restrictions amount to simply the type checking of arguments. For example, *admire* demands an animate subject (*John admires sincerity* but not *Sincerity admires John*). (The final section discusses in more detail how selectional restrictions may be learned.)

These three conditions are identical to what Salveter (1982) used in the MORAN system for learning Schank-type representations of words. Plainly, however, these exact-match conditions are too strong. Similar verbs can violate any or all of conditions (1)–(3). Verbs may differ in argument structure and yet be alike in meaning: *Eat* takes an object argument and *dine* does not (*John ate an apple* vs. *John dined an apple*). Similar verbs may have different selectional restrictions: *Assassinates* differs from *murder* in that it requires the thing killed to be a person and a political figure. The requirement of object identity is also too strong. It is clear that two verbs may be nearly synonymous, and yet one verb might appear in a story mentioning only Hamlet, whereas the other verb occurs with *Macbeth*. Finally, similar verbs may have causal structures that do not exactly overlap (e.g., *steal* usually leads to a pattern of causes and effects that is distinct from *take*, yet the two verbs are alike in some ways).

All of these considerations suggest that one give up the tight constraint of exact matches for a graded system of inexact matching. Computationally, the problem is one of (directed) graph isomorphism. We want to find the best possible correspondence between graphs  $G_1$  and  $G_2$  labeled with cause, verb, and preposition links.

Aside from this labeled correspondence, then, the matching process is completely syntactic; only the pattern of the graphs matters in matching. In addition, two nodes are not considered for matching unless they play the same thematic roles (agent, patient, and so forth) in the graph representations (in particular, two objects may be matched if they denote different instances of the "same" object, e.g., KNIFE-1 and KNIFE-3). (It might also be reasonable to use a weakened condition here that permits two objects to be matched even if they do not fill the same thematic roles, if they are the same object or nearly the same kind of object, but this change has not been investigated.)

The matching process is in general computationally intractable. There are  $n$  factorial possible 1-1 mappings of  $n$  nodes to  $n$  other nodes; there are more possible matches if a 1-1 correspondence is not required. Worse yet, the program must search as many graphs as there are possible stories with candidate verbs. We conclude that some potential matches must be filtered out in advance and the size of matches should be restricted if at all possible. Two heuristics have been adopted to implement this strategy.

The first heuristic limits the number of stories considered by the matcher. It is currently under development. The key idea is to look only at a few stories that are broadly similar to the story with the unknown verb. Similarity of stories is judged in terms of plot units, in the sense of Lehnert (1981). Roughly, plot-unit theory assumes that the causal relationships in a story can be summarized by extracting out the "molecular structure" of the causal network. This is done by imposing a theory of plot "molecular structure" onto the more basic causal network. Given a network description we can form a corresponding description in terms of basic plot units like loss, gain, or intentional problem resolution. The advantage of this approach is that it boils down entire stories to supergraphs of just a few nodes and links each. For instance, Lehnert showed how a complex story like *The Gift of the Magi* can be reduced to just 10 linked nodes. Applied to our current problem, the idea is to index all stories by their plot summaries and to retrieve only those that happen to closely match the current story under analysis. Of course, this extra indexing and retrieval matching itself takes time, but typically far fewer nodes are involved for matching. One difficulty is that Lehnert's plot summaries are based on a vocabulary of affective reactions, which may not be suitable for indexing causal regularities. What is needed is an approach like Lehnert's but with a different summarization vocabulary. This has not yet been done. Because it is not yet completely implemented, this heuristic currently replaced

by an oracle—a programmer who decides what stories to match against the network with the unknown verb.

The second heuristic reduces the number of nodes matched by considering only the local graph structure around the unknown verb. Specifically, we force the matching algorithm to consider only nodes directly linked to the unknown verb plus any nodes linked to these directly connected nodes, but no others. Intuitively, this makes sense: This means that distantly related events and relations are not brought in for matching at all. This is a brute force approach that directly reduces the number of matched nodes. In practice, although a full story could have a 100 or so nodes, this constraint reduces the nodes for matching down to 10 or so—an order of magnitude savings, with a corresponding savings in computation time. Finally, as mentioned, we do not consider match candidates that do not meet the tests of thematic role identity.

### Difference Pointers

Having found a match, the last job of the acquisition procedure is to calculate just how the new word differs from the old. This is at least as important as finding the best match, for only by finding differences do we learn anything new. The calculation is straightforward. We take the graphical representation of the local network around the old verb and add an annotated link indicating how the new verb differs from old. Alternatively, one could add pointers to the network representation of the new network, indicating differences. In this way, the system actually builds up a set of family resemblances. Words that are more distant from the "core" member of a family will have more difference pointers than words that are "closer" to the core. Note that this way of storing information already admits a prototype with allowable exceptions. A wormy or overly large apple can still be an apple because it will not violate enough of the prototypical apple qualities to be ruled out.

Constructing the appropriate difference pointers is often difficult. *Assassinate* differs from *murder* in that a political figure is involved, but how is the program to know (or learn) this? Currently, the difference pointer constructed in this example reveals two different AKO structures: for *assassinate* the murdered figure is AKO king, whereas for *murder* it is just AKO person. Eventually, we would like the system to connect a group of similar *assassinates* by lumping king, president, and so forth into a more general AKO political class.

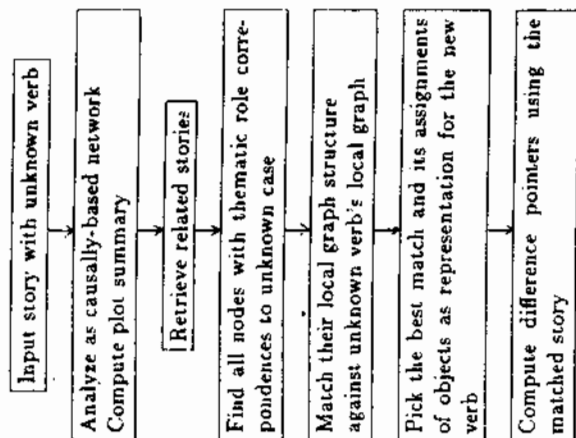


Figure 3.3. Flowchart for word learning

In general, human abilities at forming such natural classes are not well understood; some ways of handling this are discussed in the section on Learning AKO Hierarchies.

### Flowchart

We can now give a complete flow diagram of the learning procedure (see Figure 3.3).

### IMPLEMENTATION AND EXAMPLES

With the entire acquisition procedure described, this section presents some examples of the system in action. In order to fix the meaning of an unknown verb, stories are first translated into their respective extensible-relation representation. Then it provides our matcher with the unknown verb, the name of the story in which it appears, and the name of the stories to be used as precedents. The pieces of the network local to the known verb (the extent of the pieces is controlled by the user) are superimposed, one at a time, to

### 3. LEARNING WORD MEANINGS FROM EXAMPLES

the piece of the network local to the unknown verb. The verb that is probably the closest in meaning to the unknown verb will have the greatest number of matching links.

The examples use three Shakespearean plots, Hamlet (HA), Macbeth (MA), and Julius Caesar (JU):

HA is a story about a ghost Hamlet Gertrude Claudius and Laertes. Hamlet is a prince. The ghost is a dead king. Claudius is an evil king. Gertrude is a queen. Gertrude is a naive woman. Laertes is a man. The ghost was married to Gertrude.

Claudius, who is married to Gertrude, murdered the ghost because Claudius wanted to be king and because Claudius was evil. The ghost persuades Hamlet to kill Claudius because Claudius murdered the ghost. The ghost can persuade Hamlet to kill Claudius because Hamlet loves the ghost. Hamlet is unhappy because the ghost is dead. Claudius wants to kill Hamlet because Claudius is afraid of Hamlet. Claudius kills Gertrude. Hamlet kills Claudius because Claudius murdered the ghost and because Hamlet is loyal to the Ghost. Hamlet kills Claudius with a sword. Claudius persuades Laertes to kill Hamlet. Laertes kills Hamlet with a sword. Hamlet kills Laertes.  
Remember HA.

MA is a story about Macbeth Lady-macbeth Duncan and Macduff. Macbeth is an evil noble. Lady-macbeth is a greedy ambitious woman. Duncan is a king. Macduff is a noble.

Lady-macbeth persuades Macbeth to want to be king because she wants him to be king. She wants him to be king because she is greedy. She can convince him because Macbeth loves her and also because Macbeth is wimpy.

Macbeth assassinates Duncan with a knife. Macbeth assassinates Duncan because Macbeth wants to be king. Lady-macbeth becomes insane. She kills herself because she is insane.

Macduff becomes angry because the king was assassinated. Macduff kills Macbeth because Macbeth assassinated Duncan and because Macduff is loyal to Duncan.  
Remember MA.

JU is a story about Caesar Brutus Anthony and Cassius. Caesar is a general. He is an ambitious and foolish emperor. He is a man. Brutus who is an honest and unhappy man loves Rome. Anthony is a man. Cassius is a thin man.

Cassius convinces Brutus to murder Caesar because Cassius hates Caesar and also because Brutus is weak. Brutus murders Caesar with a knife. Brutus murders Caesar because Cassius told Brutus that Caesar was evil. Brutus is unhappy because Brutus murdered Caesar and because Brutus loved Caesar. Anthony who loved Caesar persuades some people to attack Brutus because Brutus murdered Caesar. Brutus

attacks the people because the people attacked him. Cassius also attacks the people. Brutus kills himself. He kills himself because he is unhappy. Cassius kills himself also.  
Remember JL.

To begin matching, we need to obtain the relations that are linked to the verbs. An auxiliary function, GET-RELATIONS, returns the root of each different relation linked to each verb. Invoking this function on MURDER-1 returns the list (CAUSE CAUSED-BY). Note that MURDER-1 can be thought of as premeditated, first degree murder.

In this implementation, two nodes or links in a network are possible matches if:

- they are the same object or
- they fill the same thematic roles or
- they are connected by AKO links (i.e., Macbeth and Gertrude are both a-kind-of persons) or

Macalusa (1984) is chiefly responsible for the matcher's implementation, and we follow her description of its next steps:

Because verbs frequently have multiple instances of the same relations, we risk not finding the best match were we to pair up the instances randomly. How do we decide whether CAUSE-15 or CAUSE-21 matches better with, say, CAUSE-2000? We make this decision by matching the objects of each relation to the objects of the relations of the candidate verbs. In practice, most conflicts are settled at this level. However, if there is still uncertainty, we choose the first match. The consequences of this choice will be examined later.

The routine that performs the steps indicated above is called DO-MATCH, and it returns the matching relations and matching objects. Returning the objects is necessary because we allow the user to control the extent of the local network which is to be matched. The variable controlling the locality of the match is \*MATCH-LEVEL, and if its value is greater than 1, DO-MATCH is called on each pair of matching objects. The matching relations are collected into a list by the procedure calling DO-MATCH. This procedure checks membership before adding a new pair of matching relations since our extensible-relation representation allows cycles. When there are no more new matches, or when we have reached the specified \*MATCH-LEVEL, the scoring function is applied to the list of matching relations. We use a simple scoring function, one which merely counts the number of pairs of relations. (p. 5)

Let us run through an example. Let us assume that Hamlet is retrieved as the story closest to Macbeth in terms of plot summaries

or causal similarity. We then try to find a verb closest to murder, where the score is defined as described previously, by the number of matching pairs of relations.

- (setq \*match-level 1)
1. (define-verb 'murder-1 'ha 'ma)  
Matching MURDER-1 with KILL-1 scores 1.  
Matching MURDER-1 with ASSASSINATE-1 scores 2.
  - (setq \*match-level 2)
  2. Matching MURDER-1 with ASSASSINATE-1 scores 3.

Two verbs score well: KILL-1 and ASSASSINATE-1. If we increase the local distance of matching, then ASSASSINATE-1 does better still. Most relations are eliminated by the case filter: 41 out of 66 in this example.

If we carry out the symmetric match we get this:

- (setq \*match-level 1)
1. (define-verb 'assassinate-1 'ma 'ha)  
Matching ASSASSINATE-1 with KILL-7 scores 1.  
Matching ASSASSINATE-1 with MURDER-1 scores 2.  
Matching ASSASSINATE-1 with KILL-4 scores 1.

Next, the program adds difference pointers connecting murder and assassinate. In this case, there really is no difference at all. All figures in both plots play political roles; indeed, both persons killed/assassinated (as matched) are rulers. A more interesting example is the matching of assassinate to murder in, for example, Romeo and Juliet. Here also, the match scores 2, but this time the players are not political. A difference pointer is constructed to indicate this subtlety (see Figure 3.4).

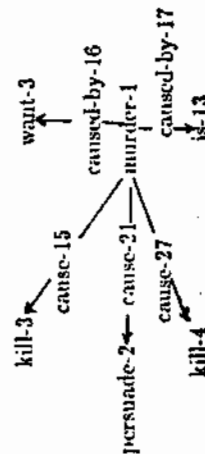


Figure 3.4. Two networks with difference pointers

Further stories should prompt the relevant generalization of the A K O RULER difference to a political classification. Note how this method builds up prototypes automatically. A causal network that reappears in many different plots will be richly connected to other networks of verbs, whereas a rarely occurring verb will be woven only weakly to other stories. Thus, what a "typical" verb is falls out automatically as a consequence of the matching procedure: it is simply a densely connected verb. (It will therefore be easily accessible, under whatever measure of access one uses, again as suggested by psycholinguistic work.)<sup>1</sup>

Macaïsa (1984) carried out some other interesting experiments with the matcher. Love and hate were similar, as were persuade and convince. Macaïsa's summary table runs as follows. The score is defined as previously stated, as the number of matching relations. The column labeled *filter* shows the number of relations in the two stories before and after a subcategorization filter is applied that removes relations that do not have the same case frame structure. Some of these results deserve further comment. Note that *attack* and *kill* are considered to match each other in *Julius Caesar* and *Hamlet*. Careful examination of the relevant subnetworks reveals that this conclusion is correct. *Attack* plays a big role in *Julius Caesar*. Here is the relevant excerpt:

Antony who loved Caesar persuades some people to attack Brutus because Brutus murdered Caesar. Brutus attacks the people because the people attacked him.

This is similar to *kill* in *Hamlet* because of a similar causal role. Both verbs are the result of a persuasion prompted by a death and love of another entity:

The ghost persuades Hamlet to kill Claudius because Claudius murdered the ghost. The ghost can persuade Hamlet to kill Claudius because Hamlet loves the ghost.

Initial tests of this method proved successful enough to encourage a full-scale test with a large verb computer database currently being

<sup>1</sup>Jackendoff (1983) correctly observed that because brains are finite we cannot store an infinite number of such prototypes. However, it is still possible to store rules to generate an arbitrary number of prototypes. It remains to investigate a "grammar" of this kind.

Verb	Story and Precedent	No. of Hits Before and After Filter	Matching Verbs	Score
murder-1	HA, MA	66, 25	kill-1 assassinate-1	1 2
murder-1	HA, JU	70, 25	kill-10 murder-2	1 2
assassinate-1	MA, HA	83, 31	kill-7 murder-1	1 2
attack-1	JU, HA	83, 30	kill-4 kill-7 kill-4	1 1 1
love-1	MA, JU	70, 25	hate-3	1
hate-1	JU, MA	66, 24	love-1	1
persuade-4	JU, HA	83, 21	persuade-2 persuade-3	2 1

developed by the MIT Linguistics Department. This database includes languages other than English. Among these are some ergative-type languages whose different syntactic-thematic linkages should provide a good test of the flexibility of the approach.

### LEARNING NOUNS AND CLASS HIERARCHIES

Having seen the learning procedure in action, it is time to step back and assess its competence. One complaint might be that people certainly do not learn the meaning of verbs in the manner suggested. For example, one could just use a dictionary. But this comment misses several points. Miller (1984) noted first of all that many verbs are learned before mastery of reading. Even so, examination of dictionary-reading behavior reveals that children will substitute known words rather than say an unknown word that is being defined by the very entry they are reading. This is only suggestive, but it hints that context-learning plays a dominant role even given learning from dictionary definitions. It certainly fulfills Wittgenstein's admonition "don't tell me the meaning, tell me the use" of a word.

More pointedly, there are many other aspects of the procedure subject to question. In this section, we review two parts of the learning systems behavior that are most open for revision. The first weak point is our choice of representational language, and the second our predefined set of AKO relationships. Two corresponding remedies are suggested: a modified representational language with some primitives, and a systematic method of generating class hierarchies.

### Beyond Causal Verbs

The system as designed so far performs adequately on verbs that have a rich causal structure, primarily verbs such as *love*, *murder*, or *persuade*. This is to be expected. The matcher uses causal relations, and so is sensitive to them. Verbs without rich causal relationships cannot be distinguished, because there is nothing to match. It is not clear, then, a simple causal structure is the right way to describe stative verbs like *be* or *become*, or motion verbs like *run*.

One alternative here is a description language outlined by Jackendoff (1983), originally proposed by Gruber (1965) and extended by Schank (1972) and Jackendoff (1972). It decomposes verbs of state or motion into primitives along a few principle axes or semantic "fields" such as temporal, possessional, or identificational, combined with three primitive verb types: *STAY*, *GO* and *BE*. *BE* denotes a state, *STAY* lack of motion, and *GO*, motion (over the semantic field). Jackendoff observed that a wide variety of verbs are modeled on the same scaffolding as these three verbs, but with modifications depending on the semantic field involved. For example, consider the temporal field. We find the following sorts of verbs:

is: *BE* The book is on the table  
 move: *GO* John moved the book to the table  
 kept: *STAY* The book was kept on the table

The same three primitives work for possessional verbs:

give: *GO* John gave a book to Mary  
 sell: *GO* object to recipient; "money" from recipient to agent  
 keep: *STAY* John kept the book

Finally, we have identificational examples:

is: *BE* John is a teacher  
 become: *GO* John became a teacher  
 remain: *STAY* John remained a teacher.

How can we graft this modified vocabulary onto our previous word-learning algorithm? All that really must be done is to expand our formerly atomic verb names such as *become* or *give* into their parts. This alters just the arc labels on the network descriptions. Matching will be modified to take note of *GO*, *STAY*, and *BE* labels, as well as semantic subfields. Consider the following example. *Macbeth* got a

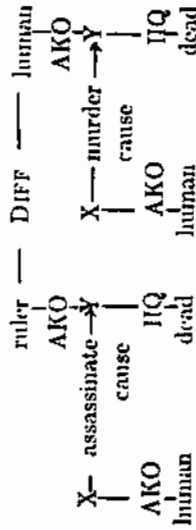


Figure 3.5. Old and new representations for *got*

knife has the two representations depicted in Figure 3.5. Now consider a verb like *obtain*. Not only does it participate in the same causal network as *get*, but it decomposes in the same way. The question is: How is this learned? Once again, we can simply use the network matching algorithm to tell us that *obtain* and *go* *possess* are paired up.

What are the advantages to this modification? Evidently, to fix a verb decomposition we must fix (at least) two parameters: first, one of the proto-verbs *GO*, *BE*, or *STAY*; second, one of the semantic fields. One positive example fixes one possibly variant of the verb's meaning, but it is easy to get others simply by choosing another parameter. For instance, if we set the semantic field to *IDENTIFICATIONAL*, then we get the verb *get* used in the sense, *Macbeth* got religion. Thus, given this decomposition, we can delineate a space of parametric variation outlining the space of possible verbs of this kind.<sup>2</sup>

It is interesting to compare this approach with another model for word acquisition, that of Granger (1977). Granger also determined the meaning of an unknown word via a matching procedure that extracted similar network representations across stories. Granger's work is founded on the same nondefinitional, context-based theory of meaning as ours, but differs in significant ways. Unlike our causally based representation, Granger's model used a conceptual dependency representation of a story. As mentioned earlier, Granger filtered story matches by (indirectly) computing the thematic roles for nouns, and retaining only those candidate matches whose thematic roles align with those of the unknown input. This representation language takes a stronger stand on decomposition than the one we adopted earlier, but is otherwise would be quite close to the modified decomposition model as just suggested. Granger's system had more trouble with verbs than nouns because its matching procedure

<sup>2</sup>Miller and Johnson-Laird (1976) probed further into a decompositional approach of this kind.

grounded on a conceptual dependency representation that in turn required one to know what verb one was analyzing. With an unknown verb, much information is absent, and so it is difficult to start matching. In addition, the system explicitly does not build a syntactic representation of a sentence; this makes it all the more difficult to bootstrap a match. In contrast, the system described in this chapter makes explicit use of syntactic constraints to drive matching. It also directly uses thematic role constraints. Finally, the causally based description says that what causes what is more important to a word's meaning than any primitive decomposition into a series of "physical transfers."

Both models benefit from a parameter-setting approach. First, the essentially combinatoric character of parameter setting immediately accounts for the productivity of word meanings. If recursive devices are some kind are allowed, then an indefinite number of word meanings can be accommodated. Second, a theory of parameter setting translates directly into a learning theory. "All" that must be learned is the settings of the various parameters. Berwick (1985) presented one model of parameter setting under the restriction that only positive examples are given. The key idea is that if positive-only examples are used, then at no time should the learner make a general guess that leaves open the possibility that the correct answer is a subset of the guess just made. The reason for this constraint should be apparent. If a subset of the hypothesis can be the correct "target," then there is now no way for the learner to find this out. Why? If the correct target is indeed a subset of the hypothesis, then any positive example will be compatible with the now overly general hypothesis. In fact, the constraint that one cannot interpolate a subset between one guess and the next turns out to be necessary and sufficient for learning using positive examples.

### Learning AKO Hierarchies

The second weak point of the learning model is that it depends on a preexisting set of AKO descriptors. It would be better if these could themselves be learned. Why, for instance, should be class *RULER* or *HUMAN* be given rather than *POLITICAL*? Yet the choice of these descriptors figures crucially in what the matcher does and what difference pointers are created.

In the remainder of this section we outline one possible method of inducing the AKO categories themselves. The central idea rests on

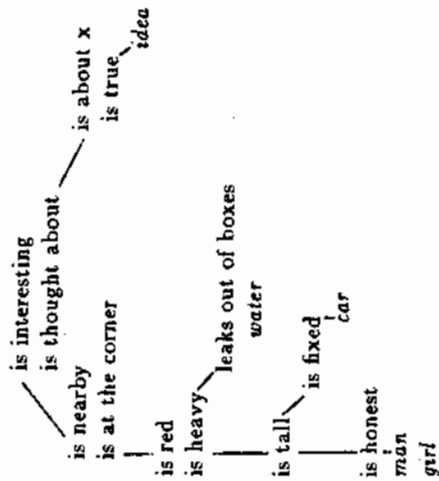
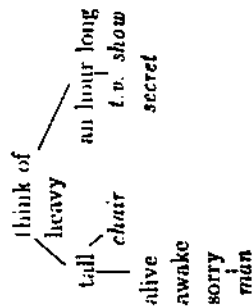


Figure 3.6. The M constraint

an observation of Sommers (1971) about semantic networks, explored in detail by Keil (1979).

Briefly, Sommers has suggested that if one arranges terms such as *dog*, *cat*, *love*, and so on into a graph structure whose terminal leaves are terms and where each node is an item that can sensibly be predicated of all terms below it in the graph, then one never, or rarely, finds human intuitions of sensibility resulting in M shaped graphs—the so-called "M" constraint. Rather, the graph structures take the form of hierarchical trees. For instance, the tree in Figure 3.6 (reproduced from Keil 1979, p. 15, Figure 1) is typical of one that comports with human intuitions; an idea can be predicated as being true, hence lies under a node labeled true in a predication tree, but cannot be at the corner, or red, hence does not lie under nodes labelled with these predicates. In contrast, a car can be nearby, at the corner, red, tall, and fixed, but not true. The resulting tree of predicates is called a *predicability tree*.

The "M" constraint relates to acquisition in the following way. Keil found through empirical studies of children that the hierarchical tree structures demanded by the "M" constraint developed by the foliation of the trees—that is, children formed new categories by splitting old ones, rather than creating entirely different tree structures. For instance, at the earliest ages studied (5–6 years), some children's predicability trees looked like that in Figure 3.7.



(This is a single individual's tree.)

Figure 3.7. Tree at early age

When second graders were tested, their trees were foliated versions of initial trees of this kind. Figure 3.8 shows the result.

We can use the foliation constraint to learn new AKO categories as follows. Assume at step  $n$  that the system has built some predicability tree. For concreteness, we shall use the tree given in Figure 3.9.

The root node corresponds to a universal predicate  $P_0$  that applies to all things. Several layers below it is a chain of predicates that apply Duncan, Macbeth, Romeo, and Juliet. That is, these predicates apply to all these items as a class; we could call this the class HUMAN. Call the last predicate node dominating all of these  $P_i$ .<sup>3</sup>

Now suppose that the system acquires a network representation of assassinate. The network indicates explicitly that Duncan can be assassinated, and the difference pointer notes this as distinct from Romeo's murder. To abide by the "M" constraint, the predicability tree should be bifurcated at  $P_i$ , carving Duncan away from its old class. Now we have the situation as depicted in Figure 3.10.

We could name the new category "political entity," but actually there is no need to name it explicitly. The class is defined by the predicates that can apply to it. A "ruler," for example, is simply something to which one can apply the predicates ALIVE, CAN REIGN, CAN BE ASSASSINATED, and so forth. Class formation follows the general principle of acquisition from positive evidence cited earlier. Suppose we define a predicability tree as a set of direct domination relations among predicates  $P_i$ . We say that  $\text{dom}(P_i, P_j)$  if and only if

<sup>3</sup>One remark in passing: this way of defining a noun in terms of the predicates that can apply to it is quite close to Montague's (1973) analysis.

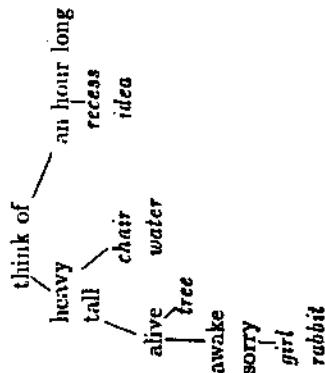


Figure 3.8. Foliated tree

$P_i$  directly dominates  $P_j$ . Then according to the class refinement scheme defined earlier, any new predicate is not permitted to destroy existing  $\text{dom}$  relations; it can only add new  $\text{dom}$  relations (between it and old  $P_j$ 's).

The justification for this monotonic refinement follows from the general principle of acquisition from positive-only evidence cited earlier.<sup>4</sup> Recall that if a learner is not to overgeneralize, then a new hypothesis should not allow the possibility of a correct target that is a proper subset of the new hypothesis. But this precisely the situation previously described: If a correct target predicability tree could include interpolated  $P_j$ 's then such trees would be subsets of the hypothesized trees (see Figure 3.11). The criterion of learning from positive-only examples would be violated.

To a certain extent, then, the empirical fact that the evolution of children's predicability trees does not allow for interpolated new predicates constitutes evidence that the constraint of acquisition from positive evidence has played a role. To make predicability trees learnable, they are constrained to obey the "M" principle. By following this constraint, the learning procedure too can make use of positive-only example, without relying on explicit negative correction. This is just a particular illustration of the general point that if negative examples are not allowed, then additional constraints are required.

<sup>4</sup>For examples of monotonic refinement in other domains of learning, see Berwick (1985).



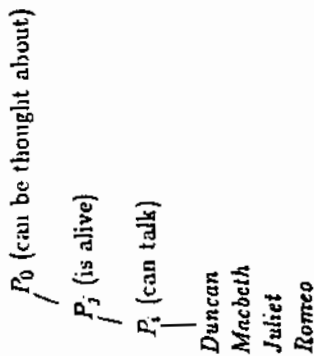


Figure 3.3. A predicability tree fragment for the Shakespeare world

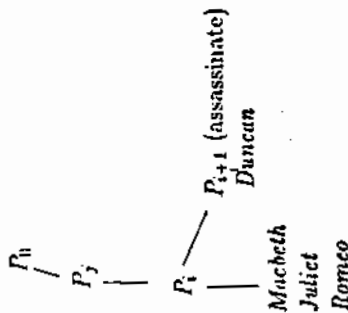


Figure 3.10. New predicability tree

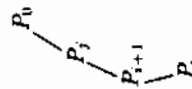


Figure 3.11. An impossible predicability tree

CONCLUSIONS

To sum up, the acquisition theory for words described here seems to meet Wittgenstein's challenge for a theory of word meaning that does not depend on dictionary definitions, but instead on a "family resemblance" model of word relationships. It successfully acquires causally based network models of verbs by the analogical analysis of story contexts containing them. Some extensions of the method for noncausally based verb descriptions were also presented. A method for learning AKO hierarchies (class descriptions) using positive-only evidence was presented. All of these learning procedures work because they are tightly constrained. The verb-learning model uses only causal links and syntactic filtering to guide it. The extension to other verbs depends on a limited set of descriptors closely linked to a spatial vocabulary. Finally, AKO learning depends on monotonic tree refinement obeying the "M" constraint. Future success of natural-language learning hinges on the discovery of constraints like these that make learning possible.

REFERENCES

Berwick, R. (1979). Learning structural descriptions of grammar rules from examples. *Proceedings of the 5th International Joint Conference on Artificial Intelligence* (pp. 56-58). Tokyo, Japan.

Berwick, R. (1980). Computational analogs of constraints on grammars. *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics* (pp. 49-53).

Berwick, R. (1982). Locality principles and the acquisition of syntactic knowledge. Unpublished doctoral dissertation, Department of Electrical Engineering and Computer Science, MIT, Cambridge, Massachusetts.

Berwick, R. (1985). *The acquisition of syntactic knowledge*. Cambridge, MA: MIT Press.

Carnap, R. (1952). Meaning postulates. *Philosophical Studies*, 3, 65-73.

Fillmore, C. (1968). The case for case. In E. Bach & R. Harms (Eds.), *Universals in linguistic theory* (pp. 1-88). New York: Holt, Rinehart, and Winston.

Fodor, J., Garret, M., Walker, E., & Parkes, C. (1980). Against definitions. *Cognition*, 8, 263-367.

Gelman, R., & Gallistel, C. (1979). *The young child's understanding of numbers*. Cambridge, MA: Harvard University Press.

Granger, R. (1977). Foul-up: A program that figures out meanings of words from context. *Proceedings of the 5th International Joint Conference on Artificial Intelligence* (pp. 172-178). Tokyo, Japan.

- Cruber, J. (1965). Studies in lexical relations. Unpublished doctoral dissertation, Department of Linguistics, MIT, Cambridge, MA.
- Hayes-Roth, B., & Hayes-Roth, F. (1977). The prominence of lexical information in memory representations of meaning. *Journal of Verbal Learning and Verbal Behavior*, 16, 119-136.
- Jackendoff, R. (1972). *Semantic interpretation in generative grammar*. Cambridge, MA: MIT Press.
- Jackendoff, R. (1977). *X syntax: A study of phrase structure*. Cambridge, MA: MIT Press.
- Jackendoff, R. (1983). *Semantics and cognition*. Cambridge, MA: MIT Press.
- Katz, B., & Winston, P. (1982). Parsing and generating English text using commutative transformations. MIT A.I. Lab Memo 677.
- Keil, F. (1979). *Semantic and conceptual development*. Cambridge, MA: Harvard University Press.
- Landau, B., & Gleitman, L. (1985). *Language and experience: Evidence from the blind child*. Cambridge, MA: Harvard University Press.
- Lehnert, W. (1981). Plot units and narrative summarization. *Cognitive Science*, 4, 293-331.
- Levin, B. (1983). On the nature of ergativity. Unpublished doctoral dissertation, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA.
- Macaluso, M. (1984). Learning word meanings from examples. Unpublished Bachelor's thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA.
- Marcus, M. (1980). *A theory of syntactic recognition for natural language*. Cambridge, MA: MIT Press.
- Miller, G. (1984). How to misread a dictionary. Proceedings of the 10th International Conference on Computational Linguistics (p. 462). Stanford, CA.
- Miller, G. & Johnson-Laird, P. (1976). *Language and perception*. Cambridge, MA: Harvard University Press.
- Montague, R. (1973). The proper treatment of quantification in ordinary English. In R. Thomason (Ed.), *Formal philosophy* (pp. 188-221). New Haven: Yale University Press.
- Salveter, S.C. (1982). Inferring building blocks for knowledge representation. In W. Lehnert & M. Ringle (eds.), *Strategies for natural language processing*, (pp. 327-344). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Schank, R. (1972). Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3(4), 552-631.
- Schank, R. (1973). Identification of conceptualizations underlying natural language. In R. Schank & K. Colby (Eds.), *Computer models of thought and language*. San Francisco: W. H. Freeman.
- Sommers, F. (1971). Structural optology. *Philosophia*, 1, 21-42.
- Winston, P. (1975). Learning structural descriptions from examples. In P. Winston (Ed.), *The psychology of computer vision* (pp. 157-210). New York: McGraw-Hill.
- Winston, P. (1980). Learning and reasoning by analogy. *Communications of the Association for Computing Machinery*, 23, 12.

## APPENDIX 1: THE STORIES

```

::This is for the Shakespeare story file - slavish attention to consistent
::usage was avoided on the ground the programs should be robust enough
::to deal with some variety and some error -- much of the initial
::stuff sets up demons and establishes the A-KIND-OF tree

::initialize

(reset)

::do not insert PART relation automatically

(setfv *insert-part* nil)

::read new-style demons

(fread "<analog>demons.lsp")

::read demons for Macbeth-WX analogy

::((fread "<analog>xcross.lsp")

::create some demons

(make-if-affirmed murder
 (affirm reference 'cause (affirm subject 'kill object))
 (affirm subject 'hate object))

(make-if-affirmed kill
 (affirm reference 'cause (affirm object 'is 'dead)))

(make-if-affirmed cruel (when (eq relation 'is) (affirm subject 'is
 'evil)))

(make-if-affirmed loyal (when (eq relation 'is) (affirm subject 'is
 'good)))

(make-if-affirmed honest (when (eq relation 'is) (affirm subject 'is
 'good)))

(make-if-affirmed married

(cond ((eq (relation reference) 'is)
 (let ((partner (get-one-object reference 'to)))
 (cond ((and partner (true? (consider reference 'to partner))))))

```

```
(affirm (affirm partner 'is 'married)
  'to
  (subject reference))))))
```

```
:::build ako tree
```

Person is an instance of a people.

Man woman and masochist are instances of a person.

Father son boy gentleman and bastard are instances of a man.

Mother daughter girl lady.hag and witch are instances of a woman.

Shrew is an instance of a bitch.

General and colonel are instances of a soldier.

Emperor Empress King and Queen are instances of a ruler.

Prince Noble Ruler Emperor and King are instances of a man.

Prince is an instance of a noble.

Empress Queen and Princess are instances of a Woman.

Knife and sword are instances of a weapon.

```
:::insert PART relation automatically
```

```
(setf *insert-part* t)
```

```
(make-if-affirmed ako
```

```
(loop for class in (get-classes subject)
```

```
do (affirm subject 'ako class)))
```

## APPENDIX 2: THE MATCHING FUNCTION

The following code is from Macalusa (1984).

```
(p '|Matching | verb '| with | rel '| scores | score))
  (setq *given-matches current-matches)
)
)
)
```

```
::: Is the given verb in story1 "close in meaning" to any verb in story2?
```

```
(defun word-match (word1 word2)
```

```
:: MATCHES contains all the matching objects; NEXT-MATCHES contains
:: the matching objects at each level, including ones which are
:: already in MATCHES; NEW-MATCHES contains all the matching objects.
:: not already in MATCHES, at each level.
(declare (special *nesting-level))
```

## 3. LEARNING WORD MEANINGS FROM EXAMPLES

```
(if (not (matching? word1 word2) (sub-obj-match? word1 word2)))
()
)
```

```
(add-to-matches word1 word2) :: Assume the two words match.
(let ((matches (list (list word1 word2)))
      (new-matches (list (list word1 word2)))
      (new-match? nil))
```

```
(loop for i from 1 to *nesting-level
```

```
with relations and next-matches
```

```
do (let ((rels-and-objs (get-next-matches new-matches)))
```

```
(add-matches (car rels-and-objs) 'relations)
```

```
(setq next-matches (cadr rels-and-objs))
```

```
(loop for match? nil) (setq new-matches nil)
```

```
(setq for match in next-matches
```

```
do (if (add-match match 'matches)
```

```
(progn
```

```
(setq new-match? t)
```

```
(setq new-matches (cons match new-matches))))))
```

```
if (and (not new-match?) (> i 1) relations)
```

```
::do (p '|***No matches past level| i) and return relations
```

```
return nil
```

```
finally (return relations)
```

```
))
```

```
))
```

```
(defun get-next-matches (old-matches)
```

```
:: (print old-matches)
```

```
(loop for match in old-matches
```

```
with relations and next-matches
```

```
do (let ((rels-and-objs (do-match (car match) (cadr match))))
```

```
:: (p '|rels:| (car rels-and-objs) '|objs:| (cadr rels-and-objs))
```

```
(setq relations (append relations (car rels-and-objs)))
```

```
(setq next-matches (append next-matches (cadr rels-and-objs))))
```

```
finally (return (list relations next-matches))
```

```
))
```

```
(defun do-match (word1 word2)
```

```
:: Works with all relations one link away from word1 and word2;
```

```
:: Adds the matching relations to *matching-relations;
```

```
:: Adds the matching objects to *matching-objects;
```

```
:: Returns an alist of the matching objects at this level.
```

```
(declare (special *uninteresting-relations *uninteresting-objects))
```

```
(let ((rels1 (get-relations word1))
```

```
(rels2 (get-relations word2))
```

```
(matching-rels1) (matching-rels)
```

```
(matching-objs1) (matching-objs))
```

```
(setq matching-rels (lists-match
```

```

(list-difference rels1 'uninteresting-relations)
(list-difference rels2 'uninteresting-relations))
(loop for rels in matching-rels
  with objs1 and objs2
do (let ((obj1 (get-objects word1 (car rels)))
        (obj2 (get-objects word2 (cadr rels))))
    (setq matching-objs
      (lists-match
        (list-difference objs1 'uninteresting-objects)
        (list-difference objs2 'uninteresting-objects)))
    ;; MATCHING-RELS1 will contain the actual instantiations
    ;; of the matching relations. MATCHING-OBJS1 will contain
    ;; a list of objects matched at this level.
    (loop for objs in matching-objs
      do (let ((rel1 (consider word1 (car rels) (car objs)))
              (rel2 (consider word2 (cadr rels) (cadr objs))))
          (if (not (subj-obj-match? rel1 rel2)) ()
              (add-match (list rel1 rel2) 'matching-rels1)
              (add-match objs 'matching-objs1))
          ))
    (list matching-rels1 matching-objs1)))

(defun add-match (item list-name)
  (let ((fcn-value t)
        (list (eval list-name)))
    (loop for item1 in list
      if (or (equal (car item) (car item1))
            (equal (cadr item) (cadr item1)))
      do (setq fcn-value nil) and return fcn-value
    finally (set list-name (cons item list)))
  fcn-value))

(defun add-to-matches (list1 list-name)
  (loop for item in list1
    do (add-match item list-name)
  ))

(defun add-to-matches (word1 word2)
  (declare (special *given-matches*))
  (add-match (list word1 word2) *given-matches)
  (let ((root1 (get word1 'root))
        (root2 (get word2 'root)))
    (if (and root1 root2)
        (add-match (list root1 root2) *given-matches))
  ))

```

```

(defun apply-filters (verb relations)
  (let ((root (get verb 'root)))
    (properties (list 'transitive))
    (pattern) (rels))
    (loop for prop in properties
      do (setq pattern (cons (get root prop) pattern)))
    (setq rels (&sort-relations relations properties pattern))
    (car rels)
  ))

(defun lists-match (list1 list2)
  (let ((match-list))
    (loop for element1 in list1
      do (loop for element2 in list2
          when (matching? element1 element2)
          do (setq list2 (delq element2 list2)) and
            return (setq match-list
                      (cons (list element1 element2) match-list)
                    ))
    match-list))

(defun subj-obj-match? (node1 node2)
  (let* ((subj1 (get node1 'subject)) (obj1 (get node1 'object))
         (subj2 (get node2 'subject)) (obj2 (get node2 'object)))
    (and (matching? subj1 subj2) (matching? obj1 obj2))
  ))

(defun matching? (object1 object2)
  (let ((answer (matching1? object1 object2)))
    (if answer ()
        (setq answer (matching1? (get object1 'root) (get object2 'root))))
  ))

(defun matching1? (object1 object2)
  (declare (special *given-matches*))
  (cond ((or (null object1) (null object2)) nil)
        ((eq object1 object2) T)
        ((or (member (list object1 object2) *given-matches)
              (member (list object2 object1) *given-matches)) T)
        ((a-kind-of-join object1 object2) T)
        (t nil)))

(defun sublist? (list1 list2)
  (let ((b1 (sublist1? list1 list2))
        (b2 (sublist1? list2 list1)))
    (if (and b1 b2)

```

```

nil ;; return nil if the two sets are "equal"
bl) ;; otherwise, return the value of the first boolean
))

(defun sublist1? (list1 list2)
  (cond ((null list1) t)
        ((member (car list1) list2) (sublist? (cdr list1) list2))
        (t nil)
        ))

(defun scoring-fcn (matching-rels)
  (let ((true-matches (eliminate-inverses matching-rels)))
    (length true-matches)))

(defun eliminate-inverses (matching-rels)
  (let* ((sorted-rels (sort-rels matching-rels))
         (cause-rels (car sorted-rels))
         (caused-by-rels (cadr sorted-rels))
         (other-rels (caddr sorted-rels)))
    (loop for cb-rel in caused-by-rels
          do (let ((cb-rel (car cb-rels)))
              (loop for c-rels in cause-rels
                    do (let ((c-rel (car c-rels)))
                        (if (inverse? c-rel cb-rel) (del c-rels 'cause-rels))
                          ))
              (append cause-rels (append caused-by-rels other-rels))
              ))
    ))

(defun sort-rels (matching-rels)
  (loop for rels in matching-rels
        with cause-rels and caused-by-rels and other-rels
        do (let* ((rel (car rels))
                 (root (get rel 'root)))
            (cond ((eq root 'cause) (add rels 'cause-rels))
                  ((eq root 'caused-by) (add rels 'caused-by-rels))
                  (t (add rels 'other-rels))
                  ))
    finally (return (list cause-rels caused-by-rels other-rels))
    ))

(defun inverse? (rel1 rel2)
  (and (eq (get c-rel 'subject)
           (get cb-rel 'object))
        (eq (get c-rel 'object)
           (get cb-rel 'subject))
        ))

```

## CHAPTER 4

### An Introduction to Plot Units

WENDY G. LEHNERT  
CYNTHIA L. LOISELLE

Department of Computer and Information Science  
University of Massachusetts

#### UNDERSTANDING AND REPRESENTATION

##### Representational Systems for Text Understanding

If a computer is to be said to understand a story, we must demand of it the same demonstrations of understanding that we require of people. When a person reads a story, an internal representation for that story is constructed in memory. For a computer to read and understand a story, it too must represent the story's content in memory. We can test both human and computer understanding by using various natural-language tasks such as answering questions or summarization. Each task will help us examine a different piece of the understanding process and the underlying representation. Question answering provides us with a method for examining the contents of the memory representation, but tells us very little about how it is structured. We can only guess at how the various pieces fit together. Summarization, on the other hand, requires concentration on the central elements of a story while ignoring peripheral information. As such it provides an excellent tool for investigating the global structure of a memory representation.