Second Edition

ENCYCLOPEDIA OF ARTIFICIAL INTELLIGENCE

Volume 2
M-Z

Stuart C. Shapiro
Editor-in-Chief

ENCYCLOPEDIA OF ARTIFICIAL INTELLIGENCE SECOND EDITION

VOLUME 2

Stuart C. Shapiro, *Editor-in-chief*

Wiley-Interscience Publication
John Wiley & Sons, Inc.
New York / Chichester / Brisbane / Toronto / Singapore

Library of Congress Cataloging in Publication Data:

Encyclopedia of artificial intelligence / Stuart C. Shapiro, editor-in
-chief. — 2nd ed.
p. cm.
"A Wiley-Interscience publication."
Includes bibliographical references and index.
ISBN 0-471-50307-X (set). — ISBN 0-471-50305-3 (v. 1). — ISBN
0-471-50306-1 (v. 2)
1. Artificial intelligence—Encyclopedias. I. Shapiro, Stuart
Charles.
Q335.E53 1992
006.3'03—dc20 91-37272

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1
Printed and bound by Courier Companies, Inc.

ENCYCLOPEDIA OF ARTIFICIAL INTELLIGENCE SECOND EDITION

VOLUME 2

MACHACK 6

Also known as The Greenblatt Chess Program, MACHACK 6 was written by Richard Greenblatt, Donald Eastlake, and Stephen Crocker at MIT. It was the first computer chess program to play well against humans in a chess tournament. It uses a method of forward pruning to variable depths along with the alpha-beta algorithm to determine its next move (see R. D. Greenblatt, D. E. Eastlake, and S. D. Crocker, The Greenblatt Chess Program, *Proceedings of the Fall Joint Computer Conference, 1967*, AFIPS Press, Montvale, N.J., 1967, pp. 801–810; M. Newborn, *Computer Chess*, Academic Press, New York, 1975).

J. Rosenberg
Kilchberg, Switzerland

MACHINE TRANSLATION

Machine translation is typically composed of the following three steps: analysis of a source language sentence, transfer (word selection and structural mapping) from one language to another, and generation of a target language sentence. It incorporates the methods of natural language understanding (qv) and sentence generation. Natural language understanding is composed of parsing technology (see Parsing; Parsing, word expert) and knowledge representation (qv). The question then arises as to whether machine translation has a particular content of its own, besides these mentioned above, which are all basic natural language technologies. The answer follows. First, there is a problem of drawing a correspondence between source language words and phrases to those of the target language. Further problems include determining what factors should decide such word selection and what cultural backgrounds of the countries where these languages are used have decisive influence on the selection of translation words and sentential styles. These problems are interesting to attack because all are central to artificial intelligence. Human translators are supposed to be doing the same in their translations works; however, research in the behavior of human translators is not advanced enough for the results to be utilized in practical machine translation systems.

There has been tremendous advancement in linguistic theories since the 1970s, and there are a few theories which are interesting from the standpoint of machine translation. However, they do not cover varieties of exceptional expressions which practical machine translation systems have to handle. A machine translation system, which is still imperfect and will never be completed, is exposed to very crude tests when the system construction reaches a certain stage. At that stage of development, the system is given a comparatively simple sentence for translation, with structures that can be analyzed by a grammar given to the system. After completion, people other than those who developed the system are asked to translate a variety of texts such as newspaper articles, science magazines, patent documents, contract documents, and commercial letters. Because the documents have not been adequately tested at the development stage, users are disappointed by the poor translation results produced by the system. Many of the failures of the system come from the fact that the dictionary and the grammar are not sufficient to accept such unexpected input sentences. There are almost infinite varieties of sentential expressions, and a few years of effort in the improvement of grammar and dictionary is not enough.

Natural language has a property of infinity in sentential structures and meanings. Some ten thousand words are normally used in everyday life. Each word has a variety of meanings and word combinations create even more meanings. Many of these cannot be explained by a standard grammar and typical meaning interpretation. Natural language can be regarded as a huge set of exceptional expressions, and as many such expressions as possible must be collected in the dictionary. It is an endless job.

There is enthusiasm for such grammar theories as lexical functional grammar and head-driven phrase structure grammar, and many experimental laboratory systems are constructed by using one of these grammar formalisms. However, such systems will not be successful as practical machine translation systems. Natural language is very complex. It is not a phenomena of natural science, and cannot be explained by a single theory. Natural language must be analyzed by the proper combination of several different principles each of which must be applied to a specific kind of linguistic phenomena. The vague total world of natural language cannot be clarified by a sharp, elegant tool. The combination of several, not sharp, but substantially powerful tools must be used. A combination of the phrase structure grammar formalism and case grammar formalism, with large and precise dictionaries that have case frame information, modality information, and many exceptional linguistic expressions, is a good model in this respect.

DIFFICULTIES OF MACHINE TRANSLATION

One of the difficulties of translation, whether it is done by a human or a machine, is that the translation of an input sentence is not unique. Various translations are possible for an input sentence, and it is very hard to decide which one is the best. The task requires constructing a box (transducer) whose input is a source language sentence and whose output is a target language sentence which conveys the same meaning. If there is a definite one-to-one correspondence between input and output, the design and construction of the box will not be so difficult. In machine translation, the input-to-output mapping is ambiguous, and the design and construction of the box cannot be achieved by a normal engineering methodology, namely a bottom-up approach from translation pairs. A top-down model-theoretic approach to machine translation must be considered. Natural language understanding systems and natural language dialogue systems consider how the human brain makes inferences and decisions, represents inside information out, and uses available information in such inference and decision making. An ideal machine translation system will be a kind of human brain system. It is only realizable by integrating research results from neurophysics, linguistics, cognitive science (qv), knowledge representation (qv), problem solving (qv), and supporting computer software.

The present-day machine translation system, unfortunately, is not advanced enough to simulate human translation activities. It performs a series of rather simple transformations on an input sentence and obtains an output sentence. The process includes some undeterministic and feedback stages, but generally speaking, is purely mechanical. The system has no world knowledge, but only a certain amount of linguistic knowledge. Almost no part of it can be properly called artificial intelligence, so they cannot produce expressions and phrases which are not explicitly given in an input sentence. The process is based on the compositionality principle, that is, a translation sentence is composed of an appropriate combination of the words or phrases whose source language words or phrases exist in the source language sentence. Many improper translations and, sometimes, failures result, particularly with complex and long input sentences. However, the systems are useful and cost-beneficial when used wisely and when weak points of the system are avoided. There are more than ten commercial systems marketed in Japan, all of which translate between English and Japanese. There are several commercial systems in the United States and Europe (Dostert and co-workers, 1978; Hutchins, 1988; King, 1987; Japan Electronic Industry Development Association, 1989).

The real difficulty in machine translation is not only in the problem of word sense; ambiguity is very often cited with regard to the inability of machine translation systems; eg, when the word "bank" is used in the sentence "I went to the bank," it is unclear which meaning is meant: a river bank, or a bank to deposit money. This problem is not solved completely in the present-day machine translation system, but will be solved by utilizing contextual information and domain knowledge (qv) about the text to be translated. In practical machine translation systems, when the ambiguity is not solvable by such information, a post-editor selects a proper solution from all the possible alternative translations. Post-editing is an obligatory step in the present-day machine translation. It is necessary not only in machine translation, but also in human translation, which goes through two steps: rough translation and post-editing. This supports the post-editing step in machine translation, which is generally considered inferior to human translation (although this is not always true).

Another difficulty in machine translation exists in the complex combination of the above ambiguity phenomena with the additional ambiguity in syntactic structure. Almost all the words in a sentence have several meanings, and the phrases in a sentence can modify several other phrases from a syntactic point of view. Therefore, the number of possible syntactic structures for a sentence of, say, twenty words, exceeds several hundred. There is no reliable method or information available to select only one right sentential structure from the many alternatives. Semantic information, contextual information, and world knowledge will play important roles in the selection of a proper solution, but there are no reliable theories which are applicable for the selection.

Current machine translation systems can analyze and translate sentences composed of less than ten words, but almost always fail to analyze and translate the sentences of more than thirty words. A reason for such failure is the ambiguity mentioned above. This is inevitable because sometimes even a human cannot understand the meaning of a long sentence at the first reading. We may feel we understand the structure and the meaning of a long sentence at the first reading, but when accurately checked we often cannot decide which phrases modify which.

To examine the real cause of the translation failure is important, although it is difficult in a large system which has over a thousand grammatical rules and tens of thousands of lexical entries. Often the cause stems from incompleteness of grammatical rules and dictionary information. A particular phrase containing unexpected adverbial phrases may sometimes put a translation system out of order. The condition for the application of a syntactic rule is not always well specified, and many unnecessary analysis trees disturb the analysis process to obtain the best result. Similar erroneous results appear when the necessary information is lacking in the dictionary and a good selection is not made for a translation word.

When causes of errors are located, the necessary improvement is taken for grammatical rules and dictionary contents, then the translation will be done correctly for the particular sentences for which the system improvements are achieved. However the improvement of a system for some sentences is sometimes harmful for other sentences which were translated correctly before the improvement. The improvement of grammar and dictionary must be done very carefully, considering varieties of possible side-effects, otherwise there will be unexpected effects for many other sentences. Improvement is difficult in a big system because the detailed behavior of a system cannot be seen and the effect of a change at a local portion to other portions cannot be estimated.

THEORIES AND METHODOLOGIES

As the above discussion demonstrates, a machine translation system is a kind of expert system. It is probably the hardest expert system to construct because a language is a monstrous system, and machine translation has to build a bridge between two such complex systems. Grammatical rules in machine translation can be regarded as production rules that find out a specific linguistic structure and transform it to another linguistic structure. However, the conditional parts of grammatical rules (the left side of a rewriting rule), that is, the conditions for the application of a particular rule, are rather simple in an ordinary grammar formalism, particularly in a context-free phrase structure grammar. Surrounding situations such as adjacent words and phrasal structures must be checked for the application of a grammatical rule. This means that a tree-to-tree transformation formalism is required for the analysis, transfer, and generation of a sentence. Even contextual information and knowledge must be used as such a condition. A difference between an ordinary expert system and a machine translation system is that the former has no good theoretical background in general but is just the accumulation of intuitive experiences, while the latter has theories such as syntactic theories and semantic theories (Grishman, 1986), although these are still very simple and imprecise. There is active research in such areas as discourse theory, situation semantics (qv), mental space theory, etc, besides the traditional linguistic theories (Brady and Berwick, 1983; Fauconnier, 1985). These theories will help the construction of machine translation systems greatly and are making rapid progress. They are, however, quite young. Therefore, it is difficult to incorporate these recent linguistic research results into existing machine translation systems at present. It will probably take another ten years to integrate all of the recent linguistics theories to achieve the next-generation machine translation systems.

Again, these theories cannot cover the whole activity of natural language because a language is a large set of exceptional expressions, and general theories are hopeless for such exceptional phenomena. Only the dictionary, which collects all these exceptions, is effective for the system, and is therefore drawing the attention of many researchers in this area (see DICTIONARY/LEXICON), as is knowledge representation in artificial intelligence. There are several researches going on in this domain. Examples are the CYC project at MCC (Lenat and Guha, 1990), and the Electronic Dictionary Project at EDR. The former aims at encoding encyclopedic knowledge in the computer in a specific representation framework. The latter involves encoding linguistic knowledge of words in a bilingual dictionary as well as monolingual dictionaries of Japanese and English. The efforts must be carefully evaluated before it is known whether the current frameworks of the encyclopedic knowledge base and the dictionaries are sufficient for various applications.

One of the research fields which has only recently gained serious attention is the comparative study of languages from the standpoint of translation. There are so many complex problems in this field that there is no methodology established yet to compare two languages. There are word level comparisons, phrase level comparisons, comparisons of grammatical structures, and comparisons of semantic, pragmatic, and culture-dependent phenomena, to mention a few, which are closely related to translation. It is well known that the French and Italian languages are very similar, as are the Japanese and Korean languages; they are close in the language family as well as in the cultural traditions. Between these closely related languages, translation is comparatively easy because word-to-word and phrase-to-phrase correspondences are simple. Also, the sentential constructions used to express a certain idea about a phenomenon or an object are almost the same, so that a deep language analysis process for machine translation is unnecessary. Shallow analysis and simple word replacement will give better translation results than the deep analysis, complex transformation, and understanding process for such language pairs.

Translation must convey the explicit and implicit information in a source language to a target language. In the case of language pairs such as the above, the simple replacement of words from one language to another and some additional simple processing will better retain delicate information from a source language sentence than does heavy analysis, which inevitably loses delicate information contained in the original sentence. Machine translation has been successful in such language pairs and has been used in everyday operations. English–French, French–Italian, English–German, and French–Spanish are successful language pairs used by the European Community, Siemens, and others. The Pan American Health Organization (PAHO) has developed their own translation systems between English and Spanish, and is using them very successfully (Vasconcellos and Leon, 1985). Speed, cost, and translation quality are said to be fairly good. Evaluation of machine translation quality must always be done by comparison with average human translation done by translators not specialized in the particular field of documents. Evaluation will become an important topic as machine translation becomes more common.

TOWARDS HUMANISTIC MACHINE TRANSLATION

Machine translation is difficult between dissimilar language families, with different cultural backgrounds, eg, English and Japanese. English is an SVO language. Many noun phrases and verb phrases have structures such as N · PP and V · PP or V · O · PP. Japanese is an SOV language and has no such structures as PP attachment from the right. Modifiers always come to the left of modifiees. Word order is rather free in Japanese, and subject and object words are often omitted. Therefore, translation necessarily entails big structural transformations and the recovery of omitted words or the deletion of redundant words, which is difficult to estimate. Because of the cultural differences, the same content is often presented in totally different expressions in these two languages, such as "I will come soon" in English, vs "I will go soon" in Japanese, and "You don't . . . No, I don't" in English, vs "You don't . . . Yes, I don't" in Japanese. Contrastive studies are essential as a basis of machine translation between two such languages.

In a sentence two components can be recognized: the core fact that a speaker wants to convey, and the speaker's attitude to the fact and to the hearer (see MENTAL MODELS). For example, the sentence "I would like to send it to you" is composed of "I send it to you" and "I would like to," with the latter expressing the attitude of the speaker toward the hearer. This part of the speaker's attitude is strongly influenced by the cultural background and by the mentality of the speaker at the time of the utterance.

Japanese people usually recognize or estimate extralinguistic information from various factors such as the speaker's sex and age, the hearer's relative position and attitude to the speaker, and official or casual situations. If the essential part of a sentence is not the factual content but rather the speaker's mentality, and if this is expressed in the way of indirect speech acts, the translation is quite difficult (McKeown, 1985). This is because the direct translation of the expression does not necessarily convey the same indirect speech act in another language and cultural world. For example, "I will consider it seriously" in Japanese sometimes means polite denial (eg, "I don't think I can" or "I don't think it is possible").

Compared to the translation of the sophisticated mentality of a sentence, it may seem that the translation of the essential fact in a sentence is not as difficult. This is not so. As mentioned, grammatical structures are completely different between different language families. Western languages such as English and French often have complex embedded structures in a sentence. The Japanese language seldom uses such structures but connects several simpler sentences (embedded phrases) in a sequence with some connectives to express the relation between these sentences. Metaphorically, the Western languages have a three-dimensional or cubic structure, and the Japanese language has linear or flat structure. Heavy structural transformations are required to bridge this structural gap. For this, a context-free phrase structure model is inadequate. A framework which transforms a tree structure to another tree structure must be introduced (Nagao and co-workers, 1985; Nagao, 1987). The Japanese language has no gender or plural and has many ellipses (eg, subject and object omissions). Good algorithms must be developed to estimate and recover this information. Contextual and situational information is important for these problems. Some studies have been done on these subjects. Particularly the usage of pronouns such as *this, that,* and *it,* in real conversational situations in Japanese have been fairly clearly distinguished by a recent study which utilizes the mental space theory (Kinsui and Takubo, 1990).

The selection of the most suitable translation word or phrase in a target language for a word or a phrase in a source language is a difficult problem. The problem lies not in choosing a word for "bank" in the sense of river bank or money bank, but in selecting a word appropriate to a context (such as the selecting among *think, consider, imagine, study, meditate,* etc). It is mapping from a source language word with vague meaning to a target language word with vague meaning whose coverage is somewhat different from the meaning of the original word. The selection is influenced not only by the word meaning, but also by the sentential structure in which the word is used. It is also affected by the sophisticated mental state of the speaker and his or her relation to the hearer. This problem, particularly in the context of contrastive study between, eg, English and Japanese, has not been studied adequately in the linguistics world.

Some trials have begun on part of the above problem, namely, the selection of a translation word according to the phrasal–sentential structure in which the word is used. One of the most promising involves storing many pairs of example phrases and their translations to compare an input partial phrase with these examples, and to extract the most similar example phrase. Then, the translation of the input phrase is done in reference to the translation of the example phrase. This principle is called *translation by analogy.* (Nagao, 1984). Humans perform this operation by consulting the dictionary for example phrases and their translations. Translation by analogy has been used in some recent research (Sumita and co-workers, 1990; Sadler, 1989). For example, it is applied to the translation of "of" in the phrase type, "*A* of *B*," in which the interpretation is quite difficult. There are many meanings in "of," and the distinction depends on the combination of the lexical meanings of *A* and *B*. The combinations are enormous, and the semantic distinction of lexical meanings of *A* and *B* and the determination of the function of "of" is difficult by the semantic primitive approach. Instead, different examples of "*A* of *B*" and their translations are stored in computer memory for comparison with the input phrase "*a* of *b*." The thesaurus is useful for determining the similarity of *a* and *A* and of *b* and *B*. When the most similar example is determined using a similarity measure based on the thesaurus nearness, the translation is produced according to the example translation.

Metaphorical interpretation of an expression is essential to the understanding and translation of a sentence. To properly perform such an interpretation, huge amounts of human knowledge ranging from academic theories to everyday customs must be accumulated, and strong associative inference processes must be performed. For example, in the sentence "I read Shakespeare," "Shakespeare" must be interpreted not as a writer but as the literary works of Shakespeare. This interpretation is performed by the inference sequence from Shakespeare to the writer to the literary works (which can be an object of "read"). However, there is still no clear definition of the conditions of the proper interpretation. It may depend on circumstances which are difficult to specify for every possible situation. This will be an interesting topic of future research.

In the future of machine translation, there will be many more problems in which artificial intelligence factors are involved. Sentence generation and dialogue interpretation and translation are typical problems that require information about the speaker's and hearer's mental states. This is called a *user model* (Kobsa, 1989), and it depends heavily on the discourse, the speaker's and hearer's intentions, speaker–hearer relations, and cultural background. Intensive future research is expected.

FROM TOY PROGRAMS TO ROBUST PRACTICAL SYSTEMS

Present-day machine translation systems are almost wholly dependent on phrase structure grammar (qv), which merges several linguistic components into one, and on case grammar (see GRAMMAR, CASE) for the recognition of core sentential structures. Case grammar includes some semantic information. One of the most widely used grammars in the academic circle today is a unification grammar which merges two elements into one. This, however, is not sufficient, because many words in a sentence influence each other at the same time. For practical systems which must process a variety of expressions, three or more elements must be checked and merged at the same time by a grammatical rule, or, tree-to-tree transformation must be introduced to cope with varieties of sentential structures. It is important for linguists to be able to write the grammatical relations of three or more elements at the same level, even though the actual checking and merging operations are done two elements at a time by the computer. Simple standard sentences will be handled by a unification grammar, but thirty years of effort in machine translation proves that such a simple-minded grammar cannot conquer the real language.

Case grammar is commonly adopted as a model to interpret natural language sentences; it is also adopted as a model of internal representation of a sentence. That is, there is an assumption that any language sentences which represent the same content can be analyzed by case grammer into the same internal representation. Case structure representation is widely considered a kind of neutral or pivot representation of the same content expressed in different languages. It is regarded as the core of a multilingual machine translation system, and hence is called the *pivot language* (Nirenburg, 1987). Ongoing debates continue as to whether the pivot method is better than the transfer method, which includes a stage that connects the internal representations of two language sentences. However, the discussion is no longer fruitful, because it has been made clear that if a multi-lingual system should handle such information as tense, modality, and other culture-dependent components of translation, there is little intersection among many different languages. Such sophisticated information can be treated only in the domain of two specific languages. Therefore, if high-quality machine translation is to be achieved, the transfer method must be adopted.

Present-day theoretical linguistics is making advances in a variety of directions, and many of these will be very useful for machine translation in the future. However, at present few of the results are in use in practical machine translation systems. At least another ten years will be necessary to construct a completely new machine translation system which will integrate all the reliable linguistic theories of morphology, syntax, semantics, and discourse, and which has the capabilities of problem solving, inferencing, and culture-dependent translation.

BIBLIOGRAPHY

J. Barwise and J. Perry, *Situations and Attitudes*, MIT Press, Cambridge, Mass., 1983.

M. Brady and R. Berwick, eds., *Computational Models of Discourse*, MIT Press, Cambridge, Mass., 1983.

B. H. Dostert, R. R. McDonald, and M. Zarechnak, *Machine Translation*, Mouton Publishers, 1978.

G. Fauconnier, *Espaces Mentaux, Edition de Minuit, 1984*, English translation: Mental Spaces, MIT Press, Cambridge, Mass., 1985.

R. Grishman, *Computational Linguistics, An Introduction*, Cambridge University Press, UK, 1986.

W. J. Hutchins, "Recent Developments in Machine Translation—A Review of the Last Five Years," *Proceedings of New Directions in Machine Translation*, Budapest, Aug. 1988.

Japan Electronic Industry Development Association, *A Japanese View of Machine Translation in Light of the Considerations and Recommendations Reported by ALPAC, U.S.A.*, July 1989.

M. King, ed., *Machine Translation Today; The State of the Art*, Edinburgh University Press, Edinburgh, UK, 1987.

S. Kinsui and Y. Takubo, "A Discourse Management Analysis of the Japanese Demonstrative Expressions," in *Advances in Japanese Cognitive Science*, Vol. 3, Kodan-sha Publishers, 1990.

A. Kobsa and W. Wahlster, eds., *User Models in Dialog Systems*, Springer-Verlag, New York, 1989.

D. B. Lenat and R. V. Guha, *Building Large Knowledge Based Systems*, Addison-Wesley, Reading, Mass., 1990.

K. R. McKeown, *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*, Cambridge University Press, UK, 1985.

M. Nagao, "A Framework of a Mechanical Translation Between Japanese and English by Analogy Principle," in A. Elithorn and R. Banerji, eds., *Artificial and Human Intelligence*, North-Holland, Amsterdam, 1984.

M. Nagao, "Role of Structural Transformation in a Machine Translation System," in S. Nirenburg, ed., *Machine Translation, Theoretical and Methodological Issues*, Cambridge University Press, UK, 1987.

M. Nagao, J. Tsujii, and J. Nakamura, "The Japanese Government Project for Machine Translation," in J. Slocum, ed., *Machine Translation Systems*, Cambridge University Press, UK, 1985.

S. Nirenburg, V. Raskin, and A. Tucker, "The Structure of Interlingua in TRANSLATOR," in S. Nirenburg, ed., *Machine Translation*, Cambridge University Press, UK, 1987.

V. Sadler, *Working with Analogical Semantics—Disambiguation Techniques in DLT*, Foris Publishers, 1989.

E. Sumita, H. Iida, and H. Kohyama, "Translating with Examples; A New Approach to Machine Translation," *Proceedings of the Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, June 11–13, 1990, Austin, Tex.

M. Vasconcellos and M. Leon, "SPANAM and ENGSPAN; Machine Translation at the Pan American Health Organization," in J. Slocum, ed., *Machine Translation Systems*, Cambridge University Press, UK, 1985.

Makoto Nagao
Kyoto University

MACHINES, SELF-ORGANIZING. See Robotics.

MACLISP. See LISP.

MACSYMA

A system for solving problems in symbolic mathematics, such as integration and algebraic manipulation, MACSYMA was designed in 1968 by J. Moses and co-workers and written in 1971 by Martin and Fateman at MIT (see Lambda calculus; W. A. Martin and R. J. Fateman, "The MACSYMA System," *Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation*, ACM SIGSAM, New York, 1971, pp. 59–75; see also J. Moses, "A MACSYMA Primer," Mathlab Memo No. 2, Computer Science Laboratory, MIT, Cambridge, Mass., 1975).

M. Taie
AT&T Bell Laboratories

MANIPULATORS. See Robot hands and end-effectors.

MANUFACTURING, AI IN

Computer-integrated manufacturing (CIM) is, basically, the technology that embraces the full range of the unique ability possessed by the digital computer and related computer technology to enhance the capabilities of the manufacturing process. That ability has three main elements. The first is the ability of the computer to provide online, variable-program (flexible) automation of manufacturing activities and equipment. The second is the ability to provide online, moment-by-moment optimization of manufacturing activities and operations. With respect to both of these elements, it should be noted that the computer can accomplish them not only with the "hard" components of manufacturing (the manufacturing machinery and equipment, etc) but also with the "soft" components (the information flow, the handling of databases, etc). However, as is becoming more and more widely recognized, the third element of the computer's unique ability is by far the most important and powerful. This is its ability to integrate all of the constituents of the entire manufacturing process into a system which can, because of the first two elements mentioned, be flexibly automated and optimized as a whole moment by moment. This powerful ability of the computer to function as a systems tool therefore results, in the case of its application to manufacturing, in what is called the computer-integrated-manufacturing system (Merchant, 1973).

That generic CIM system is defined as a closed-loop feedback system in which the prime inputs are product requirements (needs) and product concepts (creativity) and the prime outputs are finished products (fully assembled inspected and ready for use). It is comprised of a combination of software and hardware, the elements of which include product design (for production), production planning (programming), production control (feedback, supervisory and adaptive optimizing), production equipment (including machine tools), and production processes (removal, forming, and consolidative). It is amenable to being realized by application of systems engineering and has the potential of being fully automated by means of versatile automation and of being made fully self-optimizing (adaptively optimizing); the present major resources for accomplishing this are the computer-related technologies.

The general concept of this system can be more fully appreciated by reference to Figure 1. In this particular characterization of the system, five main elements are represented by the five boxes. There is nothing hard and fast about this particular characterization of the elements of the manufacturing system. The important concept to recognize is that all of the types of activities, equipment, and processes represented by the terms in the boxes are, and must be, integral parts of any manufacturing system that is to be automated, optimized, and integrated by applying the computer to these tasks if the full benefits of CIM are to be realized.

The second point to note from Figure 1 is that the CIM system is a closed-loop system. In other words, data and information relative to what is happening downstream in the system must be fed back upstream constantly, and in real time, to continuously condition the operations and activities going on there. Without such feedback, online real-time optimization and integrated, coordinated, flexible automation become impossible. Two of the more critical feedback loops, labeled "cost and capabilities" and "performance" in Figure 1, illustrate the general nature of the two types of data and information that must be fed back to provide overall flexible automation and real-time

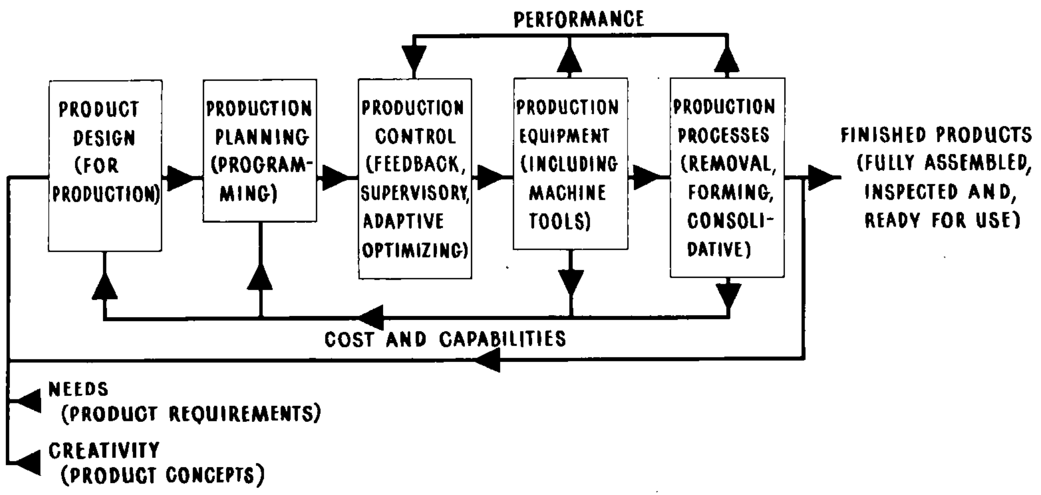
PERFORMANCE
PRODUCT DESIGN (FOR PRODUCTION)
PRODUCTION PLANNING (PROGRAMMING)
PRODUCTION CONTROL (FEEDBACK, SUPERVISORY, ADAPTIVE OPTIMIZING)
PRODUCTION EQUIPMENT (INCLUDING MACHINE TOOLS)
PRODUCTION PROCESSES (REMOVAL, FORMING, CONSOLIDATIVE)
FINISHED PRODUCTS (FULLY ASSEMBLED, INSPECTED AND, READY FOR USE)
COST AND CAPABILITIES
NEEDS (PRODUCT REQUIREMENTS)
CREATIVITY (PRODUCT CONCEPTS)

Figure 1. The CIM system.

optimization. Obviously, all data and information originating within any of the elements of the system must be able to be fed either forward or back to any of the other elements of the system where it is required.

The foregoing concept of computer-integrated manufacturing evolved during the 1960s and gradually increasing effort was devoted to its realization in practice during the 1970s, although progress was painfully slow. Nevertheless, such attention and progress led gradually toward a broadening of the concept of the CIM system that had evolved in the 1960s. By the early 1980s, it was recognized that the concept represented by Figure 1 involved primarily the technological operations of a company's system of manufacturing and that, to take full advantage of the enormous potential of computer technology to improve manufacturing capability and cost effectiveness, the concept and its application should be broadened to encompass both the technological and the business operations of a company, eg, a manufacturing enterprise. This concept was quickly espoused by the Computer and Automated Systems Association (CASA) of the Society of Manufacturing Engineers (SME) and by 1985 had evolved into the concept of the computer-integrated-manufacturing enterprise illustrated in Figure 2. As illustrated here, the current concept of CIM aims at integration of the corporate enterprise's technological manufacturing system and its business operations system, both within themselves and with each other, to accomplish overall computer integration of the corporate manufacturing enterprise. This concept of the computer-integrated manufacturing enterprise provides guidance for the major ongoing technological and managerial efforts being focused on development and implementation of full computer-automated, computer-optimized, and computer-integrated manufacturing, collectively called computer-integrated manufacturing. Despite such efforts, however, *full* computer-integrated manufacturing has not yet been realized in practice anywhere in the world. Although at this stage it is possible to integrate many parts of the system with each other, the technology is not yet sufficiently advanced to accomplish overall, closed-loop integration of the total corporate enterprise from conceptual design of the product to its delivery to the customer in finished, ready-to-use form. In particular, the greatest difficulty is being experienced in closed-loop integration of the engineering design of the product with the remainder of the technological system, and of integration of the technological system with the business operations system.

THE ROLE OF AI IN CIM

Background

The fact that CIM has not yet been fully realized anywhere is due in large part to the fact that the CIM system is not yet an intelligent system. At this stage, artificial

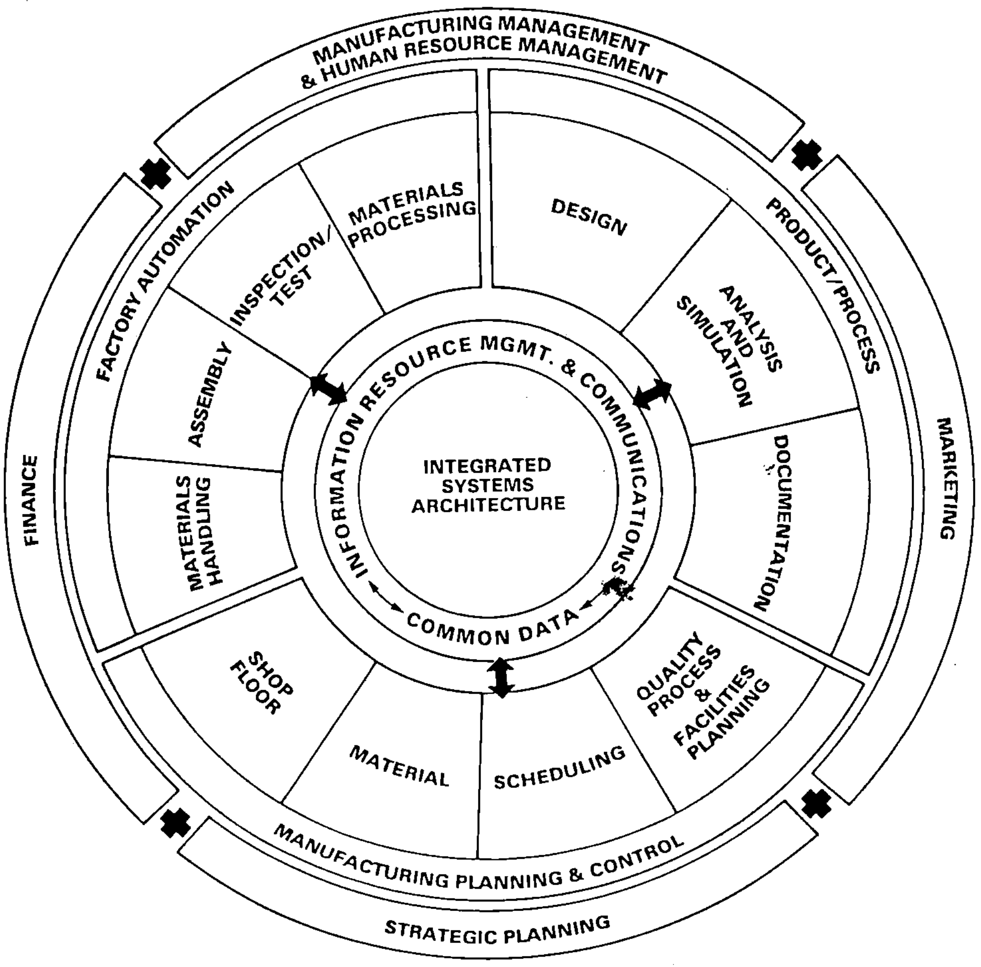

Figure 2. The CIM enterprise. CIM "wheel" developed by the Technical Council of the Society of Manufacturing Engineers' Computer and Automated Systems Association (CASA/SME, 1985). Courtesy of the Society of Manufacturing Engineers.

schema combines knowledge with the processes for using it, schemas are more like actors than like frames or systems with unitary blackboards.

As a point of comparison, consider the original form of the HEARSAY (qv) system for speech understanding, where various hypotheses generated as the interpretation of the input proceeds are placed on a single data structure called the blackboard. The blackboard has various levels to hold the hypotheses, starting from the raw spectrogram, to the phonemic level, the lexical level (the word level) and the phrasal level (binding words together into phrases with some specified grammatical and semantic structure). Each hypothesis represents the spoken input during a specific time interval, and for any time interval there may be more than one hypothesis in a level. Each hypothesis is linked to the other hypotheses that support it, and each hypothesis has a confidence level associated with it, giving a rough measure of how strong the support had so far been shown to be. Computing agents called knowledge sources act on hypotheses at one level to try to come up with a hypothesis at another level. Knowledge sources in HEARSAY include the lexicon, various processes embodying grammatical knowledge, and so on. The lexicon is like a dictionary, so that the knowledge of an individual word would have much finer granularity, more like that of schemas. Knowledge sources were brought in one at a time by a serial scheduler in HEARSAY. By contrast, schemas describe units in a network of active processes communicating with each other. Nonetheless, it is possible to use distributed blackboards in implementing schemas (see below).

Working within the perspective of schema theory, Arbib (1989a) offers an account of action and perception, Arbib and Hesse (1986) developed philosophical analyses linking the schemas of the individual to the social construction of reality (including an account of consciousness and free will), and Arbib and co-workers (1987) study language perception, acquisition, and generation. Moreover, much work in BT and AI contributes to schema theory, even though the scientists involved do not use this term. For example, Minsky (1985), in a theory that seems to subsume rather than replace his frame theory of 1977, espouses a society-of-mind analogy in which members of society, the agents, are analogous to schemas. Brooks (1986) controls robots with layers made up of asynchronous modules that can be considered as a version of schemas. This work shares with schema theory, with its mediation of action through a network of schemas, the point that no single, central, logical representation of the world need link perception and action (Arbib, 1972), while sharing with Walter (1953) and Braitenberg (1965, 1984) the study of the evolution of simple creatures with increasingly sophisticated sensorimotor capacities.

There is no consensus view as to what constitutes schema theory. The schema theory presented here owes much to the work of McCulloch. McCulloch and Pitts's (1943) formal theory of neural networks laid the basis for automata theory and, via such papers as those in Shannon and McCarthy (1956), artificial intelligence. Pitts and McCulloch (1947) gave a study of neural networks for pattern recognition that showed how visual input could control motor output via the distributed activity of a layered neural network without the intervention of executive control, perhaps the earliest example of cooperative computation. Kilmer and co-workers (1969) foreshadowed the schema level of analysis by showing how to analyze activity in a brain region that could set the organism's overall mode of behavior through the cooperative computation (again, no executive control) of modules that aggregate the activity of many neurons. Arbib and Didday (1971) developed the slide-box metaphor for visual perception and started the work on *Rana computatrix* (Arbib, 1989b), which models data on visuomotor coordination in the frog and toad to study the integration of action and perception in distributed systems in such a way as to contribute both to brain theory and perceptual robotics. This, plus the work of Bernstein (1967) on synergies as units of motor control, led to the analysis of visual perception and motor control in terms of slides and output feature clusters (Arbib, 1972), which was then refined and renamed perceptual schemas and motor schemas, respectively (Arbib, 1975, 1981a).

THE RS APPROACH

There is no formalism that captures all aspects of current and future work in schema theory (any more than Turing machines provide a good formal model of Pascal-like programs, let alone recursive or concurrent object-oriented programs). Nonetheless, it may be useful to outline one formal approach to schemas that embodies many of the criteria listed above for schemas for DAI. The Robot Schema language (RS) (Lyons and Arbib, 1989) is a language designed to facilitate sensory-based task-level robot programming. Computation is performed in a distributed manner by the interaction of a number of concurrent computing agents, known as schema instances. A schema constitutes the long-term memory of a perceptual or motor skill or the structure coordinating such skills, whereas the process of perception or action is controlled by active copies of schemas, called *schema instances*. For certain behaviors, there may be no distinction between schema and instance, a single neural network may embody the skill memory and provide the processor that implements it. However, in more complex behaviors, the different mobilizations of a given skill-unit must be carefully distinguished.

In the RS formalism, each schema instance (SI) has a set of input and output ports through which it can communicate with other schema instances. The behavioral description of a schema defines how an instance of that schema will behave in response to communication. An assemblage is a network of schema instances, and its characteristics are similar to that of a single schema. Instantiation and deinstantiation operations capture the notion that, as action and perception progress, certain schema instances need no longer be active (they are deinstantiated), while new ones are added as new objects are perceived and new plans of action are elaborated (schemas are instantiated as new schema instances). Two more points in preparation for what follows: a schema assemblage formed as a network of schema instances may itself schema combines knowledge with the processes for using it, schemas are more like actors than like frames or systems with unitary blackboards.

As a point of comparison, consider the original form of the HEARSAY (qv) system for speech understanding, where various hypotheses generated as the interpretation of the input proceeds are placed on a single data structure called the blackboard. The blackboard has various levels to hold the hypotheses, starting from the raw spectrogram, to the phonemic level, the lexical level (the word level) and the phrasal level (binding words together into phrases with some specified grammatical and semantic structure). Each hypothesis represents the spoken input during a specific time interval, and for any time interval there may be more than one hypothesis in a level. Each hypothesis is linked to the other hypotheses that support it, and each hypothesis has a confidence level associated with it, giving a rough measure of how strong the support had so far been shown to be. Computing agents called knowledge sources act on hypotheses at one level to try to come up with a hypothesis at another level. Knowledge sources in HEARSAY include the lexicon, various processes embodying grammatical knowledge, and so on. The lexicon is like a dictionary, so that the knowledge of an individual word would have much finer granularity, more like that of schemas. Knowledge sources were brought in one at a time by a serial scheduler in HEARSAY. By contrast, schemas describe units in a network of active processes communicating with each other. Nonetheless, it is possible to use distributed blackboards in implementing schemas (see below).

Working within the perspective of schema theory, Arbib (1989a) offers an account of action and perception, Arbib and Hesse (1986) developed philosophical analyses linking the schemas of the individual to the social construction of reality (including an account of consciousness and free will), and Arbib and co-workers (1987) study language perception, acquisition, and generation. Moreover, much work in BT and AI contributes to schema theory, even though the scientists involved do not use this term. For example, Minsky (1985), in a theory that seems to subsume rather than replace his frame theory of 1977, espouses a society-of-mind analogy in which members of society, the agents, are analogous to schemas. Brooks (1986) controls robots with layers made up of asynchronous modules that can be considered as a version of schemas. This work shares with schema theory, with its mediation of action through a network of schemas, the point that no single, central, logical representation of the world need link perception and action (Arbib, 1972), while sharing with Walter (1953) and Braitenberg (1965, 1984) the study of the evolution of simple creatures with increasingly sophisticated sensorimotor capacities.

There is no consensus view as to what constitutes schema theory. The schema theory presented here owes much to the work of McCulloch. McCulloch and Pitts's (1943) formal theory of neural networks laid the basis for automata theory and, via such papers as those in Shannon and McCarthy (1956), artificial intelligence. Pitts and McCulloch (1947) gave a study of neural networks for pattern recognition that showed how visual input could control motor output via the distributed activity of a layered neural network without the intervention of executive control, perhaps the earliest example of cooperative computation. Kilmer and co-workers (1969) foreshadowed the schema level of analysis by showing how to analyze activity in a brain region that could set the organism's overall mode of behavior through the cooperative computation (again, no executive control) of modules that aggregate the activity of many neurons. Arbib and Didday (1971) developed the slide-box metaphor for visual perception and started the work on *Rana computatrix* (Arbib, 1989b), which models data on visuomotor coordination in the frog and toad to study the integration of action and perception in distributed systems in such a way as to contribute both to brain theory and perceptual robotics. This, plus the work of Bernstein (1967) on synergies as units of motor control, led to the analysis of visual perception and motor control in terms of slides and output feature clusters (Arbib, 1972), which was then refined and renamed perceptual schemas and motor schemas, respectively (Arbib, 1975, 1981a).

THE RS APPROACH

There is no formalism that captures all aspects of current and future work in schema theory (any more than Turing machines provide a good formal model of Pascal-like programs, let alone recursive or concurrent object-oriented programs). Nonetheless, it may be useful to outline one formal approach to schemas that embodies many of the criteria listed above for schemas for DAI. The Robot Schema language (RS) (Lyons and Arbib, 1989) is a language designed to facilitate sensory-based task-level robot programming. Computation is performed in a distributed manner by the interaction of a number of concurrent computing agents, known as schema instances. A schema constitutes the long-term memory of a perceptual or motor skill or the structure coordinating such skills, whereas the process of perception or action is controlled by active copies of schemas, called *schema instances*. For certain behaviors, there may be no distinction between schema and instance, a single neural network may embody the skill memory and provide the processor that implements it. However, in more complex behaviors, the different mobilizations of a given skill-unit must be carefully distinguished.

In the RS formalism, each schema instance (SI) has a set of input and output ports through which it can communicate with other schema instances. The behavioral description of a schema defines how an instance of that schema will behave in response to communication. An assemblage is a network of schema instances, and its characteristics are similar to that of a single schema. Instantiation and deinstantiation operations capture the notion that, as action and perception progress, certain schema instances need no longer be active (they are deinstantiated), while new ones are added as new objects are perceived and new plans of action are elaborated (schemas are instantiated as new schema instances). Two more points in preparation for what follows: a schema assemblage formed as a network of schema instances may itself

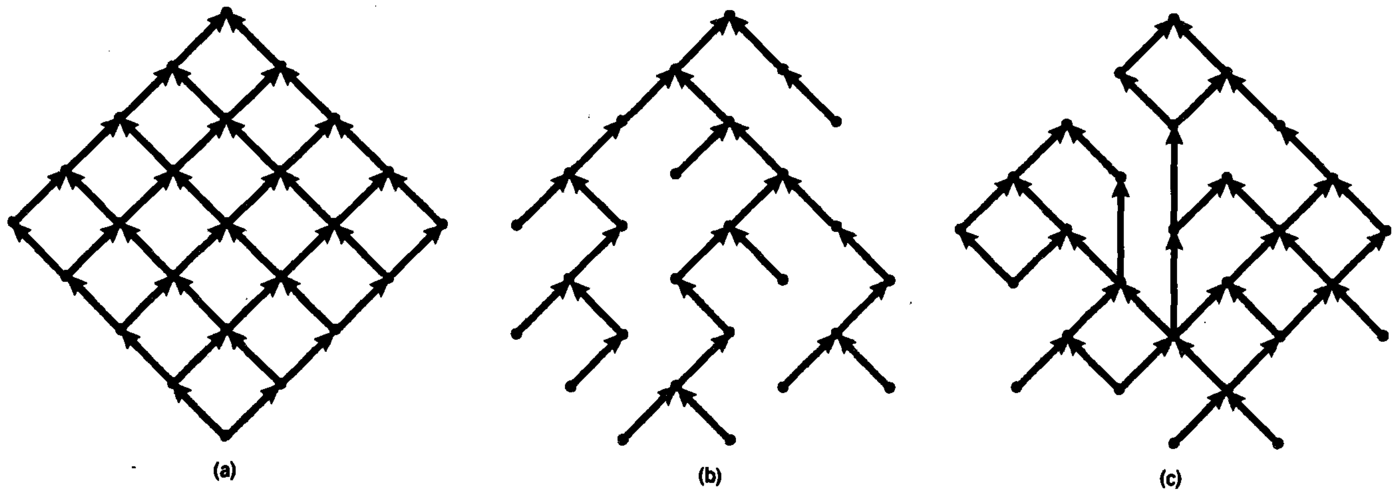
Figure 12. Three kinds of hierarchies: (*a*) lattice; (*b*) tree; and (*c*) arbitrary.

In Figure 12, the arrows extend from subtypes to supertypes. Unlike trees, a lattice allows types to have multiple immediate supertypes. An arbitrary acyclic graph like the one on the right is not a lattice because some pairs of nodes have no common supertype, some pairs have no common subtype, and some pairs have more than one minimal common supertype. Following are the characteristics of these kinds of hierarchies:

1. *Acyclic Graph.* Every partial ordering can be drawn as an acyclic graph, a graph with no cycles. Although an acyclic graph has no cycles, it may have branches that separate and come back together again, permitting some nodes to have more than one parent. Such graphs are sometimes called tangled hierarchies.
2. *Tree.* The simplest hierarchy is a *rooted tree.* It is an acyclic graph with further constraints that untangle the hierarchy: There is a single general type at the top, and every other type has exactly one immediate supertype.
3. *Lattice.* Unlike trees, lattices may have nodes with multiple parents. But they impose other constraints: every pair of types x and y must have a unique minimal common supertype $x \cup y$ and a unique maximal common subtype $x \cap y$. These constraints cause a lattice to look like a tree from both ends. At the top, it has a maximal node $\top$ that is a supertype of all others, and at the bottom, it has a minimal node $\bot$ that is a subtype of all others. (It is also possible to have infinite lattices with no top or bottom.)

Lattices are general enough to solve Aristotle's problem, since they allow different subdivisions to coexist. The type ANIMAL, for example, could have subtypes SEA-ANIMAL and LAND-ANIMAL as well as subtypes BIRD, MAMMAL, and FISH. Then SEA-BIRD would be an immediate subtype of both BIRD and SEA-ANIMAL.

The first mechanized type lattice was Leibniz's universal characteristic (1903). He assigned integers to each type: the number 1 would represent $\top$ or the supreme genus; each property or *differentia,* that distinguishes a subtype from its immediate supertype would be represented by a prime number. Then each type below $\top$ would be represented by the product of the primes for each of its differentiae. The set of all these numbers forms a lattice: For any types x and y, x is a subtype of y if y divides x; the minimal common supertype $x \cup y$ is the greatest common divisor of x and y; and the maximal common subtype $x \cap y$ is the least common multiple. One of Leibniz's reasons for designing the first calculator to do multiplication and division was to automate this system of reasoning. Masterman's original semantic network (1961) was also organized as a lattice. Since then, lattices have been widely used to support multiple inheritance through different paths in the type hierarchy.

INHERITANCE

A major use for a type hierarchy is to allow properties to be inherited from supertypes to subtypes: Whatever is true for any ANIMAL is true for any subtype such as MAMMAL, BIRD, or FISH. The original inheritance system is Aristotle's theory of syllogisms. According to Lukasiewicz (1957), the basic form of an Aristotelian syllogism is a conditional:

If *A* is predicated of all *B*,
and *B* is predicated of all *C*,
then *A* is predicated of all *C*.

This is the pattern of the first *mood*, which the medieval Scholastics named Barbara (the three *a*'s in "Barbara" indicate three universal affirmative clauses). Aristotle systematically analyzed all 24 valid moods with four kinds of clauses: universal affirmative (A); particular affirmative (I); universal negative (E); and particular negative (O). The syllogism Celarent for example, has three clauses of type E, A, and E:

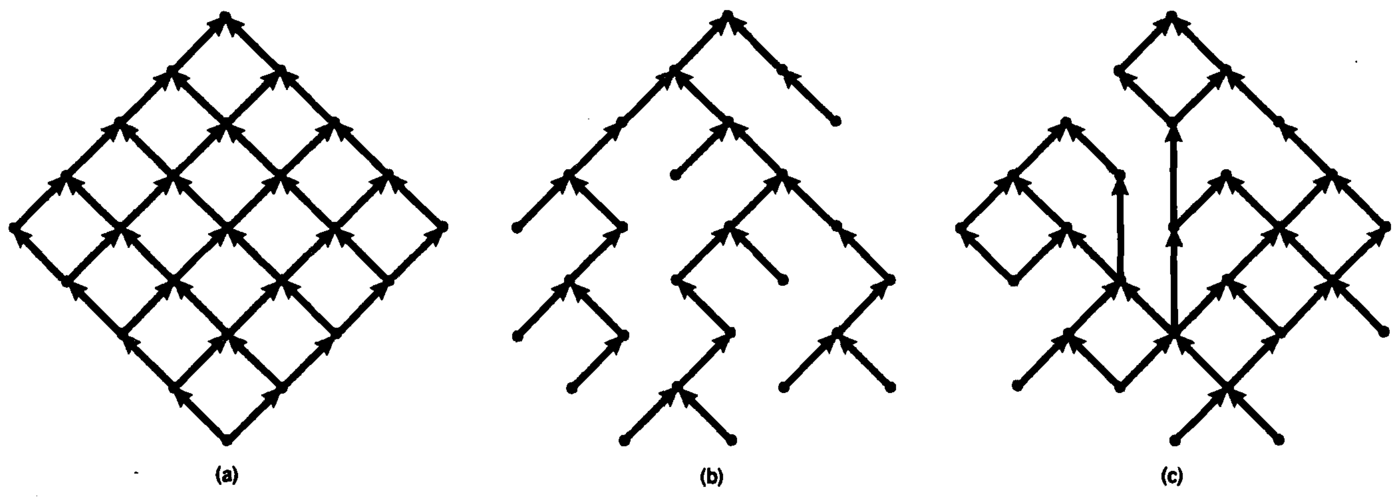

Figure 12. Three kinds of hierarchies: (*a*) lattice; (*b*) tree; and (*c*) arbitrary.

In Figure 12, the arrows extend from subtypes to supertypes. Unlike trees, a lattice allows types to have multiple immediate supertypes. An arbitrary acyclic graph like the one on the right is not a lattice because some pairs of nodes have no common supertype, some pairs have no common subtype, and some pairs have more than one minimal common supertype. Following are the characteristics of these kinds of hierarchies:

1. *Acyclic Graph.* Every partial ordering can be drawn as an acyclic graph, a graph with no cycles. Although an acyclic graph has no cycles, it may have branches that separate and come back together again, permitting some nodes to have more than one parent. Such graphs are sometimes called tangled hierarchies.
2. *Tree.* The simplest hierarchy is a *rooted tree.* It is an acyclic graph with further constraints that untangle the hierarchy: There is a single general type at the top, and every other type has exactly one immediate supertype.
3. *Lattice.* Unlike trees, lattices may have nodes with multiple parents. But they impose other constraints: every pair of types x and y must have a unique minimal common supertype $x \cup y$ and a unique maximal common subtype $x \cap y$. These constraints cause a lattice to look like a tree from both ends. At the top, it has a maximal node $\top$ that is a supertype of all others, and at the bottom, it has a minimal node $\perp$ that is a subtype of all others. (It is also possible to have infinite lattices with no top or bottom.)

Lattices are general enough to solve Aristotle's problem, since they allow different subdivisions to coexist. The type ANIMAL, *for* example, could have subtypes SEA-ANIMAL and LAND-ANIMAL as well as subtypes BIRD, MAMMAL, and FISH. Then SEA-BIRD would be an immediate subtype of both BIRD and SEA-ANIMAL.

The first mechanized type lattice was Leibniz's universal characteristic (1903). He assigned integers to each type: the number 1 would represent $\top$ or the supreme genus; each property or *differentia,* that distinguishes a subtype from its immediate supertype would be represented by a prime number. Then each type below $\top$ would be represented by the product of the primes for each of its differentiae. The set of all these numbers forms a lattice: For any types x and y, x is a subtype of y if y divides x; the minimal common supertype $x \cup y$ is the greatest common divisor of x and y; and the maximal common subtype $x \cap y$ is the least common multiple. One of Leibniz's reasons for designing the first calculator to do multiplication and division was to automate this system of reasoning. Masterman's original semantic network (1961) was also organized as a lattice. Since then, lattices have been widely used to support multiple inheritance through different paths in the type hierarchy.

INHERITANCE

A major use for a type hierarchy is to allow properties to be inherited from supertypes to subtypes: Whatever is true for any ANIMAL is true for any subtype such as MAMMAL, BIRD, or FISH. The original inheritance system is Aristotle's theory of syllogisms. According to Lukasiewicz (1957), the basic form of an Aristotelian syllogism is a conditional:

> If A is predicated of all B,
> and B is predicated of all C,
> then A is predicated of all C.

This is the pattern of the first *mood,* which the medieval Scholastics named Barbara (the three *a*'s in "Barbara" indicate three universal affirmative clauses). Aristotle systematically analyzed all 24 valid moods with four kinds of clauses: universal affirmative (A); particular affirmative (I); universal negative (E); and particular negative (O). The syllogism Celarent for example, has three clauses of type E, A, and E:

M. Weck, "Development and Application of a Flexible, Modular Monitoring and Diagnosis System," *Comput. Ind.* **7**, 45–51 (1986).

M. Weck and G. Kiratli, "Applicability of Expert Systems to Flexible Manufacturing," in *Robot. Comput. Integr. Manufact.* **3**, 97–103 (1987).

W. J. Zdeblick and B. E. Barkocy, "Manufacturing Planning Evolution with Artificial Intelligence and Applications Toward Machining Operations," *Proceedings of the PROLAMAT Sixth International Conference,* AFCET, Paris, 1985, pp. 99–108.

M. Eugene Merchant
Institute of Advanced Manufacturing Sciences, Inc.

MATHEMATICAL INDUCTION. See Induction, mathematical.

MATHEMATICAL KNOWLEDGE REPRESENTATION

One of the intended applications of the MKRP (qv) system is to prove all the lemmas and theorems of a standard textbook of a particular mathematical field. With the present system, this is possible in principle (one third of a text book on semigroups and automata has, in fact, been encoded and proved) but there are problems: it is cumbersome to express typical mathematical notions in plain first-order predicate calculus, which is more like a "machine language" of the MKRP system. Encoding a text book requires a user-friendly input language that offers some higher order constructs which can be automatically translated into the "machine language" of the logical engine.

The factual knowledge of a mathematical field is highly structured, and this structure influences the search for a proof; eg, a human mathematician knows when and down to which level to expand a definition. An adequate representation for such knowledge was developed and a substantial part of the mathematical knowledge of a text book was encoded into this representation.

J. H. Siekmann
Universität Kaiserslautern

MEANS-ENDS ANALYSIS

Means-ends analysis is a term that is quite descriptive. In the context of problem-solving (qv), it refers to the process of comparing what is given or known to what is desired and, on the basis of this comparison, selecting a "reasonable" thing to do next. This definition is deliberately informal and general because it is intended to capture the essential nature of a number of different, but similar, problem-solving methods.

The use of means-ends analysis in computer programs that solve problems dates back to 1957 when it was first used in GPS (qv) (Newell, Shaw, and Simon, 1960). Since then, mean-ends analysis has been the topic of considerable research, only some of which was part of the GPS effort. This article describes most of this research; it starts with a description of GPS, partly for historical reasons, but also because it is quite easy to describe other programs given a technical description of GPS. Amazingly enough, the variant of means-ends analysis used in GPS is still one of the more elaborate and subtle problem-solving methods reported in the literature (ie, methods that have applicability in several different domains, unlike a method that can only be used, for example, in chess).

Over the years, research on GPS had at least three distinct goals: One was empirical explorations into problem solving and generality. This was important in the earlier years because little was known about how to get a computer to behave intelligently. The final version of GPS, which is the culmination of this research, is described in Ernst and Newell (1969). A second goal was the simulation of cognitive processes for the purpose of understanding the extent to which GPS can be used as a model of human problem solving. A good reference to this research is Newell and Simon (1972), which also contains several other models of human problem solving. The remaining contribution of research on GPS is its problem-solving method; it is the only one described in this entry.

Another variant of means-ends analysis is used in FDS (Quinlan and Hunt, 1968), which is described after GPS. This problem solver was designed for a certain class of theorem-proving problems. In the early seventies there was a significant research effort at Stanford Research Institute on the use of problem solving in robotics. This work was based on STRIPS (Fikes and Nilsson, 1971), a computer program that used means-ends analysis. It and one of its successors, ABSTRIPS (Sacerdoti, 1974) is described after FDS.

The last variant of means-ends analysis in this entry is MPS (Korf, 1985). This research differs from the others in that it focuses on learning good strategies for solving problems. The strategies that are learned use mean-ends analysis and are sufficiently powerful for MPS to solve difficult puzzles, such as Rubik's cube. The entry closes with a discussion of how to choose good differences for GPS. The differences are problem-dependent parameters whose purpose is to guide the search for a solution.

GPS

GPS is an acronym for general problem-solver. This name stems from the fact that it was the first problem-solving program that separated the problem-dependent and the problem-independent parts of the system in a reasonably clean way. GPS was designed to solve state space problems (Nilsson, 1971) in which there is an initial state, a set of goal states, and a set of operators. Each operator *f* is a partial function on states; dom(*f*) denotes its domain. A solution to a problem is a sequence of operators that transforms the initial state into a goal state. Each intermediate state produced by one of these operators must be in the domain of the next operator in the sequence.

To solve a problem, GPS creates a hierarchy of goals;

the first goal is to transform the initial state into a goal state. Assuming that the initial state is not a goal state. GPS detects differences between them and then attempts to reduce one of these differences. For this kind of goal, GPS selects an operator that is relevant to reducing the difference and creates the goal of applying the operator. A separate goal is used for this because the initial state may not be in the domain of the operator that gives rise to a difference and the goal of reducing it.

This very brief description of how GPS works contains the three different kinds of goals that GPS uses:

- Transform a state into a set of states,
- Reduce a difference possessed by a state, and
- Apply an operator to a state.

The method for achieving a transform goal tests if the state is in the set of states. If not, the goal of reducing the largest difference between them is created, followed by the goal of transforming its result into the set of states. GPS requires the differences to be totally ordered. The method for achieving a reduce goal selects a relevant operator and creates the goal of applying it. GPS requires a table that indicates which operators are relevant to which differences. The method for achieving an apply goal tests if the state is in the domain of the operator. If not, the goal of reducing the largest difference between them is created, followed by the goal of applying the operator to the result of the reduce goal. To summarize, GPS uses three different kinds of goals, and for each there is a single method for achieving it. This is a slightly simplified picture of GPS. There was a select type of goal and other methods also. These were required for operators that mapped two states into a third state. GPS could handle such operators even though they are excluded by the state space paradigm; the details are in Ernst and Newell (1969). This entry only describes how GPS solved state space problems.

Information about differences is a problem-dependent parameter to GPS; its purpose is to make the search more efficient. More problem-solving methods designed for more than a single problem have such parameters; for example, *h* is such a parameter to the A* problem-solving method (Nilsson, 1971) (see A* ALGORITHM). GPS requires the following information about differences:

- The differences to be used;
- An ordering on these differences; and
- For each difference, the operators relevant to reducing it.

Intuitively, the differences are just properties of states that are appropriate for the given problem. Some of these are more difficult to remove than others, and thus they are ordered according to their difficulty. GPS employs the strategy of removing differences in order of their difficulty, the most difficult first. Any operator will be relevant to removing some differences but not others. These intuitive concepts can be used inside of GPS because it is parameterized by the information about differences, which varies from problem to problem.

A trace of GPS solving a simple problem will make the above description more concrete. The problem is the three-disk Tower of Hanoi whose initial state has three disks of different diameters stacked on the first peg in ascending order; the other two pegs are empty. There is only one goal state in which all the disks are on the third peg. The operators move the top disk from one peg to another, provided that the disk being moved is not placed on a smaller disk.

Figure 1 shows how GPS solves this problem. As always, the first goal is to transform the initial state s0 to the goal states G. Comparing the two, GPS detects that all of the disks are in the wrong positions. The second goal is to reduce the largest of these differences; d3 indicates that the position of disk 3 is incorrect, and GPS notes in goal 2 that it should be on peg 3. Goal 3 is to apply the operator that moves disk 3 to peg 3. Of course, this operator cannot be directly applied because s0 is not in its domain since the other two disks are not on peg 2. Therefore, goal 4 is created to reduce d2, the larger of these two differences. Since disk 2 cannot be moved in s0 (goal 5), disk 1 is moved to peg 3 (goals 6 and 7), which results in the new state s1. Disk 2 is then moved in s1 (goal 8) resulting in s2. Goal 9 is another attempt to move disk 3, but this cannot be done in s2, and GPS continues in a similar manner.

The indentation in Figure 1 is important because it shows the hierarchical relationship among the goals. Although this relationship is obvious for the first seven goals, note that goal 9 is a subgoal of goal 3. This is impor-

```
 1. Transform s0 into G.
    2. Reduce d3 of s0 to peg-3.
       3. Apply disk-3 → peg-3 to s0.
          4. Reduce d2 of s0 to peg-2.
             5. Apply disk-2 → peg-2 to s0.
                6. Reduce d1 of s0 to peg-3.
                   7. Apply disk-1 → peg-3 to s0.
                      s1 = ((2 3) () (1))
                8. Apply disk-2 → peg-2 to s1.
                   s2 = ((3) (2) (1))
          9. Apply disk-3 → peg-3 to s2.
             10. Reduce d1 of s2 to peg-2.
                 11. Apply disk-1 → peg-2 to s2.
                     s3 = ((3) (1 2) ())
             12. Apply disk-3 → peg-3 to s3.
                 s4 = (() (1 2) (3))
    13. Transform s4 into G.
        14. Reduce d2 of s4 to peg-3.
            15. Apply disk-2 → peg-3 to s4.
                16. Reduce d1 of s4 to peg-2.
                    17. Apply disk-1 → peg-1 to s4.
                        s5 = ((1) (2) (3))
                18. Apply disk-2 → peg-3 to s5.
                    s6 = ((1) () (2 3))
        19. Transform s6 into G.
            20. Reduce d1 of s6 to peg-3.
                21. Apply disk-1 → peg-3 to s6.
                    s7 = (() () (1 2 3))
            22. Transform s7 into G.
                Success.
```

Figure 1. A trace of GPS solving the three-disk Tower of Hanoi problem.