

knowledge prepared by the software "Oracle" of the system, and down-loaded prior to the start of the search.

Hitech has been in preparation for nearly 3 years, during which time graduate student Carl Ebeling designed the hardware and MOSIS manufactured the special purpose VLSI chips. Since the beginning of 1985, a team headed by Hans Berliner has fleshed out the system to be a useful chess playing entity. Involved in the system building were: Carl Ebeling: Hardware design and construction, Gordon Goetsch: system software, Andy Palay: initial concept and search strategies, Murray Campbell: openings and testing; Larry Slomer: hardware construction, and Hans Berliner: chess knowledge.

A SURVEY ON SPECIAL PURPOSE COMPUTER ARCHITECTURES FOR AI

Benjamin W. Wah and Guo-Jie Li
Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1101 W. Springfield Avenue
Urbana, IL 61801¹

ABSTRACT

In this survey, we will provide a short survey and classification on the current work in special purpose architectures to support AI applications. In spite of the growing importance of AI applications, work in the area of designing AI architectures are so diversified that articles were published in other areas besides AI, ranging from psychology, medicine, manufacturing, computer architecture, software engineering, and database management to industrial engineering, operations research, and the list grows. The literature search is also complicated by the fact that with the development of the Fifth-Generation Computer Systems, some work in this area is very recent and was published in many foreign countries. During our literature search to compile this survey, we systematically went through over sixty different journals published in various countries and proceedings from over fifty conferences in the last twenty years and over seventy books.

1. INTRODUCTION

Many of today's computers are single-processor von Neumann machines designed for sequential and deterministic numerical computations[1-4], and are not equipped for AI applications that are mainly parallel non-deterministic symbolic manipulations[5-8]. Consequently, efficient computer architectures for AI applications would be sufficiently different from traditional computers[9-18]. These architectures have the following requirements.

Symbolic Processing: In the microlevel, AI applications require symbolic processing operations such as

¹Research supported by National Science Foundation grant DMC 85-19849.

comparison, selection, sorting, matching, logic set operations (union, intersection, and negation), contexts and partition, transitive closure, pattern retrieval, and recognition. In a higher level, these applications may require the processing of nonnumerical data such as sentences, speech, graphics, and images. Efficient computers designed for these applications should possess hardware for symbolic processing functions[19-21]. The most important ones are tagged mechanisms[20,22] and hardware stacks[23].

Parallel and Distributed Processing: Most AI problems are complex[24,25] and must be evaluated by high-performance computers. Due to technological limitations of physical devices, parallelism is perhaps the only promising mechanism to further improve the performance of computers[8,10,26-28]. To prevent the bottleneck of a centralized controller, intelligence in such a system should be decentralized. In applying multiprocessing and distributed processing to solve problems with exponential complexity, which is typical for problems in AI, one must realize that multiprocessing is useful in improving the computational efficiency, and *not in extending the solvable problem size*[30]. To extend the solvable problem space of such problems, the key is to find better models and more efficient heuristics.

Nondeterministic Processing: Most AI algorithms are nondeterministic[31], that is, it is impossible to plan in advance the procedures to execute and to terminate with the available information. Therefore, dynamic allocation and load balancing of computational resources are essential in AI architectures[10,32]. Further, an efficient interconnection network is needed to disseminate information for the scheduler. The tradeoff between the overhead of distributing the scheduling information and the overhead for the extra work needed without the scheduling information must be made. Moreover, efficient garbage collection is important for AI architectures owing to the dynamically allocated storage[32-34].

Knowledge Base Management: Since a very large amount of information have to be stored and retrieved in AI applications, large knowledge bases are inevitable[35-38]. An implementation using a common memory is inappropriate due to access conflicts. A decentralized memory system with distributed intelligence and capabilities for pattern matching and proximity search is required.

Software Oriented Computer Architectures: The efficiency of a computer system for an AI application depends strongly on its knowledge representation and the language used. An efficient AI architecture should be designed around the knowledge representations of the problems to be solved and the high-level AI languages to be supported. Further, the designed architectures should adapt to changes in granularity and data formats of various applications. Examples of these architectures are the dataflow machines[39,40], object-oriented architectures[41,42], Lisp machines[18,22], and Prolog-like machines, such as the Fifth Generation Computer System[14].

Currently, extensive research are carried out in designing efficient AI architectures. Many existing concepts in computer architecture, such as dataflow processing[43,44], stack machines[23], tagging[20], pipelining[27], direct execution of high-level languages[45-47], database machines[48], multiprocessing, and distributed processing, can be incorporated into future

AI architectures. New concepts in computer architectures are also expected.

2. ARTIFICIAL INTELLIGENCE LANGUAGES AND PROGRAMMING

One goal of computer scientists working in the field of AI is to produce programs that imitate intelligent behavior of human beings[49-53]. Von-Neumann-style programming that uses imperative languages, such as Fortran and Pascal, is inadequate due to its inability to specify parallel tasks and its unacceptable complexity[54-56]. To enhance programmers' productivity, a type of problem-oriented languages called declarative languages have been developed and widely applied in AI programming[57]. *Functional programming*[55,58,59] and *logic programming*[60-64] are the major programming paradigms of declarative languages.

Functional programming does not contain any notion of the present state, program counter, or storage. Rather, the "program" is a function in the true mathematical sense: it is applied to the input of the program, and the resulting value is the program's output. The terms *functional language*, *applicative language*, *dataflow language*, and *reduction language* have been used somewhat interchangeably[65-68]. Examples of functional languages are pure Lisp[69-73], Backus' FP[54], Hope[74], Val[75] and Id[76]. Interest in functional programming is steadily growing because it is one of the few approaches that offer a real hope of relieving the twin crises of AI-oriented computing today: the absolute necessity to reduce the cost of programming, and the need to find computer designs that make much better use of the power of VLSI and parallelism.

In its modest form, a logic program refers to the procedural interpretation of Horn clauses predicate logic[63,64]. The computer language Prolog[77-82] is based on logic programming. Generally speaking, logic programming is a reasoning-oriented or deductive programming. In fact, some ideas of logic programming, like automatic backtracking, have been used in early AI languages QA3[49], PLANNER, MICROPLANNER, and CONNIVER[51,83]. Logic programming has recently received considerable attention because of its choice by the Japanese as the core computer language for the Fifth Generation Computer System Project[84]. Although it seems on the surface that logic programming is an independent and somewhat separate notion from function programming, an ideal AI-programming style should combine the features of both languages and may be called "assertional programming"[61].

New languages and programming systems are being developed to simplify AI programming radically. It is expected that *object-oriented programming*[85] will be important in the 1980's as structured programming was in the 1970's. The language Smalltalk[86,87] is an example of object-oriented programming. Some ideas of object-oriented programming have been used in existing languages and systems, such as Simula, 85000, Lisp-AI notion of frame, ADA, and CLU. Other new object-oriented programming systems have also been developed[41,42,88-90].

AI programming languages have had a central role in the history of AI research. Frequently, new ideas in AI are accompanied by a new language that is natural for the ideas to be applied. Except for the widely used language Prolog, Lisp and its dialects, MacLisp[91], InterLisp[92,93],

Clisp[94], Common Lisp[95], Franz Lisp[96], etc., many other AI language have been designed and implemented. Examples include IPL[97,98] PLANNER[99], CONNIVER[83], KRL[100], NETL[101], SAIL[102], POP-2[103], FUZZY[104] and first-order logic. In general, three capabilities, namely, action, description, and reasoning, are needed for an AI language. Historically, languages strong in one of these capacities tended to be relatively weak in others. Prolog is a reasoning-oriented language that is limited by its inefficiency of description and action. Lisp, the second oldest programming language in present widespread use retains some features of von Neumann programming. Some new languages, such as Loglisp[61] and QUTE[105], which amalgamate Prolog and Lisp in natural ways, have been developed. On the other hand, to explore parallelism, the parallel versions of Prolog and Lisp, such as Parlog[106], Concurrent Prolog[107,108], and Concurrent Lisp[109,110], have been proposed. Recent efforts are aimed at automatic programming that will allow the program to be generated from a simple specification of the problem[111-115].

Besides programming languages, it became apparent to the AI community since the mid-1960s that inference alone, even those augmented with heuristics, were often inadequate to solve real-life problems. To enhance the performance of AI programs, they must be augmented with knowledge of the problem domain rather than formal reasoning methods. This realization gave birth to *knowledge engineering* or *knowledge-based system*, the field of applied AI[116,117].

A *knowledge-based expert system*, or in short, expert system, is a knowledge-intensive program that solves problems in a specific domain normally requiring human expertise[17,118-124]. An expert system consists of two parts: knowledge base and inference procedure. The knowledge base contains the facts and heuristics, while the inference procedure consists of the processes that search the knowledge base to infer solutions to problems, form hypotheses, and so on. What distinguishes an expert system from an ordinary computer application is that, in a conventional computer program, pertinent knowledge and the methods for utilizing it are all intermixed, while in an expert system, the knowledge base is separated from the inference procedure, and new knowledge can be added to the system without reprogramming.

Contemporary expert-system development techniques are shifting towards the use of software development tools that resemble a programming language, but include internal user-accessible databases and other high-level strategies for using knowledge to solve a class of problems[119,122,125,126]. Each tool suggests some additional design properties, such as rule-base and backward reasoning, for the knowledge-system architecture. Three of the most popular families of expert-system tools are: (1) EMYCIN[127,128], KS300, and S.1; (2) HEARSAY-III[129] and AGE[130]; and (3) OPS that incorporates the MYCIN, HEARSAY-II, and R1 (XCON) expert-system families[131]. Other expert-system tools include LOOPS[132], ROSIE[133], RLL[134], MRS, and KMS. Some of these tools aim to provide a mixture of representations and inference techniques. Knowledge-acquisition tools such as TEIRESIAS[135], EXPERT[136], KAS[137], and learning tools such as META-DENDRAL[138] and EURISKO[139], have also been developed.

3. MICRO AND MACRO LEVEL AI ARCHITECTURES

The VLSI (Very-Large-Scale-Integration) technology flourished in the past ten years[140-144] and resulted in the development of advanced microprocessors[145], semiconductor memories[146], and systolic arrays[147-151].

The microlevel architectures consist of architectural designs that are fundamental to applications in AI. In the design of massively parallel AI machines[152], some of the basic computational problems recognized are set intersection, transitive closure, contexts and partitions, best-match recognition, Gestalt recognition, and recognition under transformation. These operations may not be unique in AI and may exist in many other applications as well. Due to the simplicity of some of these operations, they can usually be implemented directly in hardware, especially in systolic arrays using the VLSI technology. Many other basic operations can also be implemented in VLSI. Examples include sorting[153-162] and selection[163,164], computing transitive closure[147,165], string and pattern matching[19,166-172], selection from secondary memories[173,174], dynamic programming evaluations[165,175,176], proximity searches[177], and unification[178-182].

Some AI languages such as Lisp differ from traditional machine languages in that the program/data storage is conceptually an unordered set of linked record structures of various sizes, rather than an ordered, indexable vector of numbers or bit fields of a fixed size. The instruction set must be designed according to the storage structure[183]. Additional concepts that are well suited for list processing are the tagged-memory[184,20] and stack architectures[23].

The macrolevel is an intermediate level between the microlevel and the system level. In contrast to the microlevel architectures, the macrolevel architectures are (possibly) made up of a variety of microlevel architectures and perform more complex operations. However, they are not considered as a complete system that can solve problems in AI applications, but can be taken as more complex supporting mechanisms for the system level. The architectures can be classified into dictionary machines, database machines, architectures for searching, and architectures for managing data structures.

A dictionary machine is an architecture that supports the insertion, deletion, and searching for membership, extremum, and proximity of keys in a database[185-192]. Most designs are based on binary-tree architectures; however, design using radix trees and a small number of processors have been found to be preferable when keys are long and clustered[187].

A database machine is an architectural approach that distributes the search intelligence into the secondary and mass storage, and relieves the workload of the central processor. Extensive research has been carried out in the past decade on optical and mass storage[193,194], backend storage systems[195], and database machines[196-205]. Earlier database machines developed were mainly directed towards general-purpose relational database management systems. Examples include the DBC, DIRECT, RAP, CASSM, associative array processors, text retrieval systems[196,197], and CAFS[202]. Nearly all current research on database machines to support knowledge databases assume that the knowledge database is relational, hence research is directed towards solving the disk paradox[198] and enhancing previous relational database

machines by extensive parallelism[206-208,38]. Commercially available database and backend machines have also been applied in knowledge management[209-211].

Searching is an essential to many applications, although unnecessary combinatorial searches should be avoided. The suitability of parallel processing to searching depends on the problem complexity, the problem representation, and the corresponding search algorithms. Problem complexity should be low enough such that a serial computer can solve the problem in a reasonable amount of time. Problem representations are very important because they are related to the search algorithms. Parallel algorithms have been found to be able to dramatically reduce the average-time behavior of search problems, the so-called combinatorially implosive algorithms[212-214].

A search problem can be represented as searching an acyclic graph or a search tree. According to the functions of nodes in the graph, the problem is transformed into one of the following paradigms: (a) AND-tree (or graph) search: all nonterminal nodes are AND nodes, (b) OR-tree (or graph) search: all nonterminal nodes are OR nodes, and (c) AND/OR-tree (or graph) search: the nonterminal nodes are either AND or OR nodes. A divide-and-conquer algorithm is an example algorithm to search AND trees; a branch-and-bound algorithm is used to search OR trees; and an alpha-beta algorithm is used to search (AND/OR) game trees. Parallel algorithms for divide-and-conquer[215], branch-and-bound[216-223], and AND/OR-graph search[224-226] have been developed. Various parallel architectures to support divide-and-conquer algorithms[227,228] and branch-and-bound algorithms[229-238,30] have been proposed.

Extensive research has been carried out in supporting dynamic data structures in a computer with a limited memory space. Garbage collection is an algorithm that periodically reclaims memory space no longer needed by the users[32-34,239-251]. This is usually transparent to the users and could be implemented in hardware, software, or a combination of both. For efficiency reasons, additional hardware such as stacks and reference counters are usually provided.

4. FUNCTIONAL-PROGRAMMING-ORIENTED ARCHITECTURES

The origin of functional languages as a practical class of computer languages can perhaps be traced to the development of Lisp by McCarthy[70] in the early 60's, but their ancestry went directly back to the lambda calculus developed by Church in the 1930's. The objective of writing a functional program is to define a set of (possibly recursive) equations for each function[252]. Data structures are handled by introducing a special class of functions called constructor functions. This view allows functional languages to deal directly with structures that would be termed "abstract" in more conventional languages. Moreover, functions themselves can be passed around as data objects. The design of the necessary computer architecture to support functional languages thus centers around the mechanisms of efficient manipulation of data structures (list-oriented architectures) and the parallel evaluation of functional programs (function-oriented architectures).

List-oriented architectures are architectures designed to efficiently support the manipulation of

data structures and objects. Lisp, a mnemonic for list processing language, is a well known language to support symbolic processing. There are several reasons why Lisp and list-oriented computers are really needed. First, to relieve the burden on the programmers, Lisp was designed as an untyped language. The computer must be able to identify the types of data, which involves an enormous amount of data-type checking and the use of long strings of instructions at compile and run times. Conventional computers cannot do these efficiently in software. Second, the system must periodically perform garbage collection and reclaim unused memory at run time. This amounts to around ten to thirty percents of the total processing time in a conventional computer. Hardware implementation of garbage collection is thus essential. Third, due the nature of recursion, a stack-oriented architecture is more suitable for list processing. Lastly, list processing usually requires an enormous amount of space, and the data structures are so dynamic that the compiler cannot predict how much space to allocate at compile time. Special hardware to manage the data structures and the large memory space would make the system more cost effective and efficient[253-255,18].

The earliest implementation of Lisp machines were the PDP-6 computer and its successors the PDP-10 and PDP-20 made by the Digital Equipment Corporation[70]. The half-word instructions and the stack instructions of these machines were developed with Lisp's requirements in mind. Extensive work has been done for the DEC-system 10's and 20's on garbage collection to manage and reclaim the memory space used.

The design of Lisp machines was started at MIT's AI Laboratory in 1974. CONS, designed in 1976[256-259], was superseded in 1978 by a second-generation Lisp machine, the CADR. This machine was a model for the first commercially available Lisp machines[260-262], including the Symbolics LM2, the Xerox 1100 Interlisp work station, and the Lisp Machine Inc. Series III CADR, all of them delivered in 1981. The third-generation machines were based on additional hardware to support data tagging and garbage collection. They are characterized by the Lisp Machines Inc. Lambda supporting Zetalisp and LMLisp[260,261,22], the Symbolics 3600 supporting Zetalisp, Flavors, and Fortran 77[20,263-265], the Xerox 1108 and 1132 supporting Interlisp-D and Smalltalk[266-268], and the Fujitsu FACOM Alpha Machine, a backend Lisp processor supporting MacLisp[269,270]. Most of the Lisp machines support networking using Ethernet. The LMI Lambda has a NuBus developed at MIT to produce a modular, expandable Lisp machine with multiprocessor architecture.

A single-chip computer to support Lisp has been implemented in the MIT SCHEME-79 chip[271,272,183]. Other experimental computers to support Lisp and list-oriented processing have been reported[273-282]. These machines usually have additional hardware tables, hashing hardware, tag mechanisms, and list processing hardware, or are microprogrammed to provide macroinstructions for list processing. Experimental multiprocessing systems have been proposed to execute Lisp programs concurrently[110,283-288]. Dataflow processing is suitable for Lisp as these programs are generally data driven[289,290]. Other multiprocessing and dataflow architectures to support list processing have been proposed and developed[291-296].

Besides specialized hardware implementations, software implementations on general-purpose computers are also popular. The earliest Lisp compilers were

developed on the IBM 704 and later extended to the IBM 7090, 360, and 370. Various strategies for implementing Lisp compilers have been proposed[71,21,297,93,72,298], and conventional microcomputers have been used to implement Lisp compilers[299-302]. Lisp is also available on various general- and special-purpose work stations, typically based on multiple 68000 processors[299,302]. Lisp has been developed on Digital Equipment Corp. VAXstation 100, a MC68000-based personal graphics work station, and clusters of 11/782s running several dialects of Lisp and Common Lisp[303]. One dialect of Lisp, Franz Lisp, developed at the University of California, Berkeley, was written in C and runs under Unix and is available on many general-purpose work stations.

Architectures have also been developed to support object-oriented programming languages which have been extended from functional languages to additionally implement operations such as creating an object, sending and receiving messages, modifying an objects' state, and forming class-superclass hierarchies[304,305,41]. Smalltalk, first developed in 1972 by the Xerox Corp., is recognized as a simple but powerful way of communicating with computers. At MIT, the concept was extended to become the Flavors system. Special hardware and multiprocessors have been proposed to directly support the processing of object-oriented languages[306,42,89,90].

In *function-oriented architectures*, the design issues center on the physical interconnection of processors, the method used to "drive" the computation, the representation of programs and data, the method to invoke and control parallelism, and the optimization techniques[307]. Desirable features of such architectures should include a multiprocessor system with a rich interconnection structure, the representation of list structures by balanced trees, and hardware supports for demand-driven execution, low-overhead process creation, and storage management.

Architectures to support functional-programming languages can be classified as uniprocessor architectures, tree-structured machines, data-driven machines, and demand-driven machines. In a uniprocessor architecture, besides the mechanisms to handle lists, additional stacks to handle function calls and optimization for redundant calls and array operations may be implemented[272,308-310,67]. Tree-structured machines usually employ lazy evaluations, but suffer from the bottleneck at the root of the tree[311-316]. Dataflow machines are also natural candidates for executing functional programs and have tremendous potential for parallelism. However, the issue of controlling parallelism remains unresolved. A lot of the recent work is concentrated on demand-driven machines which are based on reduction machines on a set of load-balanced (possibly virtual) processors[282,317-326].

Owing to the different motivations and objectives of various functional-programming-oriented architectures, each machine has its own distinct features. For example, the Symbolics 3600[265] was designed for an interactive program development environment where compilation is very frequent and ought to appear instantaneous to the user. This requirement simplified the design of the compiler and results in only a single-address instruction format, no indexed and indirect addressing modes, and other mechanisms to minimize the number of nontrivial choices to be made. On the other hand, the aim in developing SOAR[90] was to demonstrate that a Reduced Instruction Set Computer could provide high performance in an ex-

platory programming environment. Instead of microcode, SOAR relied on software to provide complicated operations. As a result, more sophisticated software techniques were used.

5. LOGIC AND KNOWLEDGE ORIENTED ARCHITECTURES

In logic and knowledge oriented architectures, the ideal goal is for the user to specify the problem in terms of the properties of the problem and the solution (logic or knowledge), and the architecture exercises the control on how the problem is to be solved. This goal is not fully achieved yet, and users still need to provide small but undue amounts of control information in logic programs, partly by ordering the clauses and goals in a program, and partly by the use of extra-logical "features" in the language.

Knowledge and logic oriented architectures can be classified according to the knowledge representation schemes. Besides incorporating knowledge into a program written in a functional programming language, some of the well-known schemes are logic programs and semantic networks. According to the search strategy, logic programs can further be classified into production systems and logical inference systems[15-17,327,178,179].

Substantial research has been carried out on parallel computational models of utilizing AND parallelism, OR parallelism, and stream parallelism in logical inference systems[39,328-342], production systems[343-345], and others[346]. The basic problem on the exponential complexity of logic programs remains open at this time.

Sequential Prolog machines using software interpretation[347,348], emulation[349,350], and additional hardware support such as hardware unification and backtracking[351,352,180] have been proposed. Single-processor systems for production systems using additional data memories[353] and a RISC architecture[179] have been studied.

New logic programming languages suitable for parallel processing have been investigated[354]. In particular, the use of predicate logic[355], extensions of Prolog to become Concurrent Prolog[356-362], Parlog[363], and Delta-Prolog[364], and parallel production systems[365] have been developed. Concurrent Prolog has also been extended to include object-oriented programming[360] and has been applied as a VLSI design language[361]. One interesting parallel language is that of systolic programming, which is useful as an algorithm design and programming methodology for high-level-language parallel computers[366].

Several prototype multiprocessor systems for processing inference programs and Prolog have been proposed, some of which are currently under construction. These systems include multiprocessors with a shared memory[358], ZMOB, a multiprocessor of Z80's connected by a ring network[367-371], AQUARIUS, a heterogeneous multiprocessor with a crossbar switch[372], and MAGO, a cellular machine implementing a Prolog compiler that translates a Prolog program into a formal functional program[373]. Techniques for analyzing Prolog programs such that they can be processed on a dataflow architecture have been derived[374-376,40]. DADO is a multiprocessor system with a binary-tree interconnection network that implements parallel production systems[377-380]. An associative processor has been

proposed to carry out propositional and first-order predicate calculus[381].

It has been recognized that a combination of Lisp, Prolog, and an object-oriented language such as Smalltalk may be a better language for AI applications[382]. Computer of this type that implements a combination of the AI languages may use microprogramming to emulate the various functions. Prolog is also available as a secondary language on some Lisp machines. A version of Prolog interpreter with a speed of 4.5 klips has been developed for Lisp Machine's Lambda[260]. Some of the prototype multiprocessors, such as ZMOB[367-371] and MAGO[373], were developed with a flexible architecture that can implement object-oriented, functional, and logic languages. FAIM-1, a multiprocessor connected in the form of a twisted hex-plane topology, implements the features of object-oriented, functional, and logic programming in the OIL programming language[383].

Besides representing knowledge in logic, it can also be represented in terms of semantic nets. Proposed and experimental architectures have been developed. NETL[384,101,385], and its generalization to THISTLE[152], consists of an array of simple cells with marker-passing capability to perform searches, set-intersections, inheritance of properties and descriptions, and multiple-context operations on semantic nets. Thinking Machine's Connection Machine is a cellular machine with 65,536 processing elements. It implements marker passing and virtually reconfigures the processing elements to match the topology of the application semantic nets[386,387]. Associative processors for processing semantic nets have also been proposed[388,389].

Some AI architectures are based on frame representations and may be called object-oriented architectures. For example, the *Apiary* developed at MIT is a multiprocessor actor system[286]. Actor is an object that contains a small amount of state and can perform a few primitive operations: sending a message, creating another actor, making a decision, and changing its local state. An efficient AI architecture also depends on the problem-solving strategy. The basic idea of the *Boltzmann machine* developed at the Carnegie-Mellon University is the application of statistical mechanics to constraint-satisfaction searches in a parallel network[390]. The most interesting aspect of this machine lies in its domain-independent learning algorithm[391].

With the inclusion of control into stored knowledge, the resulting system becomes a distributed problem solving system. These systems are characterized by the relative autonomy of the problem solving nodes, a direct consequence of the limited communication capability[392-394]. With the proposed formalism of the Contract Net, contracts are used to express the control of problem solving in a distributed processor architecture[395-397]. Related work in this area include Petri-net modeling[398], distributed vehicle-monitoring testbed[399,400], distributed air-traffic control system[401], and modeling the brain as a distributed system[402,403].

6. FIFTH GENERATION COMPUTER SYSTEM

The Fifth-Generation-Computer-System, or FGCS, project was a project started in Japan in 1982 to further the research and development of the next generation of computers. It was conjectured that computers of the next decade will be used increasingly for nonnumeric data

processing such as symbolic manipulation and applied AI. The goals of the FGCS project are

1. to implement basic mechanisms for inference, association, and learning in hardware;
2. to prepare basic AI software in order to utilize the full power of the basic mechanisms implemented;
3. to implement the basic mechanisms for retrieving and managing a knowledge base in hardware and software;
4. to use pattern recognition and AI research achievements in developing user-oriented man-machine interfaces; and
5. to realize supporting environments for resolving the "software crisis" and enhancing software production.

The FGCS project is a marriage between the implementation of a computer system and the requirements specified by applications in AI, such as natural-language understanding and speech recognition. Specific issues studied include the choice of logic programming over functional programming, the design of the basic software systems to support knowledge acquisition, management, learning, and the intelligent interface to users, the design of highly parallel architectures to support inferencing operations, and the design of distributed-function architectures that integrates VLSI technology to support knowledge databases[14,84,404-407].

A first effort in the FGCS project is to implement a sequential inference machine, or SIM[408,409]. Its first implementation is a medium-performance machine known as a personal sequential inference, or PSI, machine[410,411]. The current implementation is on the parallel inference machine, or PIM[207,412-416,40]. Another architectural development is on the knowledge-base machine, Delta[412,207,208,417,38]. Lastly, the development of the basic software system acts as a bridge to fill the gap between a highly parallel computer architecture and knowledge information processing[418-420]. Currently, all the projects are progressing well; however, the struggle is still far from over[421].

The Japanese FGCS project has stirred intensive responses from other countries[319,320,422-430]. The British project is a five-year \$550 million cooperative program between government and industry that concentrates on software engineering, intelligent knowledge-based systems, VLSI circuitry, and man-machine interfaces. Hardware development has focused on ALICE, a Parlog machine using dataflow architectures and implementing both Hope, Prolog, and Lisp[319,320,426-429]. The European Commission has started the \$1.5 billion five-year European Strategic Program for Research in Information Technologies (Esprit) in 1984[423]. The program focuses on microelectronics, software technology, advanced information processing, computer-integrated manufacturing, and office automation. In the United States, the most direct response to the Japanese FGCS project was the establishment of the Microelectronics and Computer Technology Corp. in 1983[430]. The project has an annual budget of \$50 million to \$80 million per year. It has a more evolutionary approach than the revolutionary approach of the Japanese and should yield technology that the corporate sponsors can build into advanced products in the next 10 to 12 years. Meanwhile, other research organizations have formed to develop future computer technologies of the United States in a broader sense. These include DARPA's Strategic Computing and Survivability, the semiconductor industry's Semiconductor Research Corporation, and the Microelectronics Center of North Carolina[430].

7. CONCLUSIONS

This survey briefly summarizes the state of the art in AI architectures. Conventional von Neumann computers are unsuitable for AI applications because they are designed mainly for deterministic numerical processing. To cope with the increasing inefficiency and difficulty in coding algorithms in artificial intelligence, declarative languages have been developed. Lambda-based and logic-based languages are two popular classes of declarative languages.

One of the architect's starting point in supporting applications in artificial intelligence is the language. This approach has been termed the *language-first approach*. A possible disadvantage of this approach is that each language may lead to a quite distinct architecture which is unsuited to other languages, a dilemma in high-level-language computer architectures. In artificial intelligence applications, the lambda-based and logic-based languages have been considered seriously by novel architects. Recent research lies in integrating the logic and lambda languages, and the work on lambda and logic oriented architectures provides useful guidelines for parallel architectures that support more advanced languages. On the other hand, AI architectures are also related to knowledge representations. This approach has been called the *knowledge-first approach*. Several architectures have been designed to support multiple knowledge representations.

An appropriate methodology to design an AI architecture should combine the top-down and bottom-up design approaches. That is, we need to develop functional requirements based on the AI problem requirements and map these requirements into architectures based on technological requirements. Parallel processing is a great hope to increase the power of AI machines. However, parallel processing is not a way to overcome the difficulty of combinatorial explosion. It cannot significantly extend the solvable problem space on problems that we can solve today. Hence the problem complexity is an important consideration in designing AI machines. Problems of lower complexity may be solved by sequential computation; problems of moderate complexity may be solved by parallel processing; while problems of high complexity should be solved by heuristic and parallel processing. Since the complexities of most AI problems are high, an appropriate approach should start by first designing good heuristics to reduce the serial-computational time and using parallel processing to pursue a near-linear speedup.

Although many AI architectures have been built or proposed, the Lisp machines are the only architecture that have had widespread use for solving real AI problems. Most underlying concepts in AI architectures are not new and have been used in conventional computer systems. For example, hardware stack and tagged memory were proposed before they were used in Lisp machines. On the other hand, some popular architectural concepts in current supercomputers will have restricted use in some AI applications. For example, the large amount of branch and symbolic processing operations in AI programs reduce stream parallelism in pipelining.

The question of how AI programs can be executed directly in hardware efficiently is still largely unanswered. The following are some key issues in designing AI architectures:

1. identification of parallelism in AI programs;
2. tradeoff between the benefit and the overhead on

the use of heuristic information;

3. efficient interconnection structure to distribute heuristic-guiding and pruning information;
4. granularity of parallelism;
5. dynamic scheduling and load balancing;
6. architecture to support the acquisition and learning of heuristic information;
7. predication of performance and linear scaling; and
8. management of the large memory space.

Due to the space limitation, special architectures for computer vision, speech processing, and natural language understanding are not included in this survey.

- [1] J.-L. Baer, *Computer Systems Architecture*, Computer Science Press, 1980.
- [2] J. L. Baer, "Computer Architecture," *Computer*, vol. 17, no. 10, pp. 77-87, IEEE, Oct. 1984.
- [3] H. S. Stone, *Introduction to Computer Architecture*, 2nd Edition, Science Research Associates, 1980.
- [4] G. Myer, *Advances in Computer Architecture*, Wiley, 1978.
- [5] H. Boley, "A Preliminary Survey of AI Machines," *SIGART Newsletter*, no. 72, pp. 21-28, ACM, July 1980.
- [6] S. Fahlman, "Computing Facilities for AI: A Survey of Present and Near-Future Options," *AI Magazine*, pp. 16-23, AAAI, Winter 1980-81.
- [7] Arvind and R. A. Iannucci, "A Critique of Multiprocessing von Neumann Style," *Proc. 10th Annual Int'l Symp. on Computer Architecture*, pp. 426-436, IEEE/ACM, June 1983.
- [8] P. C. Treleaven, "The New Generation of Computer Architecture," *Proc. 10th Annual Int'l Symp. on Computer Architecture*, pp. 402-409, IEEE/ACM, June 1983.
- [9] M. F. Deering, "Hardware and Software Architectures for Efficient AI," *Proc. Nat'l Conf. on AI*, pp. 73-78, AAAI, Aug. 1984.
- [10] C. Hewitt and H. Lieberman, "Design Issues in Parallel Architectures for AI," *Proc. COMPCON Spring*, pp. 418-423, IEEE, Feb. 1984.
- [11] E. A. Feigenbaum, F. Hayes-Roth, D. Waltz, R. Reddy, and V. Zue, "The Building Blocks," *Spectrum*, pp. 77-87, IEEE, Nov. 1983.
- [12] D. I. Moldovan, *Survey of Computer Architectures for AI*, Technical Report PPP-84-6, Univ. of Southern California, Los Angeles, CA, July 1984.
- [13] T. Moto-oka, et al., "Challenge for Knowledge Information Processing Systems," *Proc. Int'l Conf. on 5th Generation Systems*, pp. 3-89, ICOT, Tokyo, Japan, and North-Holland, 1981.
- [14] K. Fuchi, "The Direction the FGCS Project Will Take," *New Generation Computing*, vol. 1, no. 1, pp. 3-9, OHMSHA Ltd. and Springer-Verlag, 1983.
- [15] H. Boley, "AI Languages and AI Machines: An Overview," *Proc. German Workshop on AI*, Springer-Fachberichte, 1981.
- [16] D. Schaefer and J. Fischer, "Beyond the Supercomputer," *Spectrum*, vol. 19, no. 3, pp. 32-37, IEEE, March 1982.
- [17] D. A. Waterman and F. Hayes-Roth, *Pattern-Direct Inference Systems*, Academic Press, 1978.
- [18] M. F. Deering, "Architectures for AI," *Byte*, pp. 193-206, McGraw-Hill, April 1985.
- [19] A. Mukhopadhyay, "Hardware Algorithms for Non-numeric Computation," *Trans. on Computers*, vol. C-28, no. 6, pp. 384-394, IEEE, June 1979.
- [20] A. Hirsch, "Tagged Architecture Supports Symbolic Processing," *Computer Design*, PennWell, June 1984.
- [21] J. Campbell and J. Fitch, "Symbolic Computing With and Without Lisp," *Conf. Record of Lisp Conf.*, Stanford Univ., Menlo Park, CA, 1980.
- [22] M. Creeger, "Lisp Machines Come Out of the Lab.," *Computer Design*, pp. 132-137, PennWell, Nov. 1983.
- [23] R. Doran, "Architecture of Stack Machine," in *High-Level Language Computer Architecture*, ed. Y. Chu, Academic Press, 1975.
- [24] S. A. Cook, "An Overview of Computational Complexity," *Comm. of the ACM*, vol. 26, no. 6, pp. 401-408, ACM, June 1983.
- [25] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, 1984.
- [26] L. S. Haynes, R. L. Lau, D. P. Siewiorek, and D. W. Mizell, "A Survey of Highly Parallel Computing," *Computer*, vol. 15, no. 1, pp. 9-24, IEEE, Jan. 1982.
- [27] K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*, McGraw-Hill, 1984.
- [28] N. R. Lincoln, "Technology and Design Tradeoffs in the Creation of a Modern Supercomputer," *Trans. on Computers*, vol. C-31, no. 5, pp. 349-362, IEEE, May 1982.
- [29] J. P. Riganati and P. B. Schneck, "Supercomputing," *Computer*, vol. 17, no. 10, pp. 97-113, IEEE, Oct. 1984.
- [30] B. W. Wah, G.-J. Li, and C. F. Yu, "Multiprocessing of Combinatorial Search Problems," *Computer*, vol. 18, no. 6, pp. 93-108, IEEE, June 1985.
- [31] J. Cohen, "Non-Deterministic Algorithms," *Computing Surveys*, vol. 11, no. 2, pp. 79-94, ACM, June 1979.
- [32] H. G. Baker Jr, "Optimizing Allocation and Garbage Collection of Spaces," in *AI: An MIT Perspective*, ed. P. H. Winston and R. H. Brown, vol. 1, pp. 391-396, MIT Press, 1979.
- [33] J. Cohen, "Garbage Collection of Linked Data Structures," *Computing Surveys*, vol. 13, no. 3, pp. 341-367, ACM, Sept. 1981.
- [34] H. Lieberman and C. Hewitt, "A Real-Time Garbage Collector Based on the Lifetimes of Objects," *Comm. of the ACM*, vol. 26, no. 6, pp. 419-429, ACM, June 1983.
- [35] E. Babb, "Functional Requirements for Very Large Knowledge Bases," *Proc. ACM'84*, pp. 55-56, ACM, Oct. 1984.
- [36] M. Bartschi, "An Overview of Information Retrieval Subjects," *Computer*, vol. 18, no. 5, pp. 67-84, IEEE, May 1985.
- [37] G. Wiederhold, "Knowledge and Database Management," *Software*, vol. 1, no. 1, pp. 63-73, IEEE, Jan. 1984.

- [38] H. Sakai, K. Iwata, S. Kamiya, M. Abe, A. Tanaka, S. Shibayama, and K. Murakami, "Design and Implementation of Relational Database Engine," Proc. 5th Generation Computer Systems, pp. 419-426, ICOT and North-Holland, 1984.
- [39] L. Bic, "A Data-Driven Model for Parallel Interpretation of Logic Programs," Proc. Int'l Conf. on 5th Generation Computer Systems, pp. 517-523, ICOT and North-Holland, 1984.
- [40] N. Ito, H. Shimizu, M. Kishi, E. Kuno, and K. Rokusawa, "Data-Flow Based Execution Mechanisms of Parallel and Concurrent Prolog," New Generation Computing, vol. 3, pp. 15-41, OHMSHA Ltd. and Springer-Verlag, 1985.
- [41] M. Tokoro and Y. Ishikawa, "An Object-Oriented Approach to Knowledge Systems," Proc. Int'l Conf. on 5th Generation Computer Systems, pp. 623-632, ICOT and North-Holland, 1984.
- [42] Y. Ishikawa and M. Tokoro, "The Design of an Object-Oriented Architecture," Proc 11th Int'l Symp. on Computer Architecture, pp. 178-187, IEEE/ACM, 1984.
- [43] J. B. Dennis, "Data Flow Supercomputers," Computer, vol. 13, no. 11, pp. 48-56, IEEE, Nov. 1980.
- [44] P. C. Treleaven and I. G. Lima, "Future Computers: Logic, Data Flow, ... Control Flow?," Computer, vol. 17, no. 3, pp. 47-55, IEEE, March 1984.
- [45] Y. Chu, "Direct-Execution Computer Architecture," in Information Processing 77, ed. B. Gilchrist, pp. 18-23, North-Holland, 1977.
- [46] M. Yamamoto, "A Survey of High-Level Language Machines in Japan," Computer, vol. 14, no. 7, pp. 68-78, IEEE, July 1981.
- [47] M. J. Flynn, "Directions and Issues in Architecture and Language," Computer, vol. 13, no. 10, pp. 5-22, IEEE, Oct. 1980.
- [48] G. G. Langdon Jr., "Database Machines: An Introduction," Trans. on Computers, vol. C-28, no. 6, pp. 381-384, IEEE, June 1979.
- [49] A. Barr and E. A. Feigenbaum, The Handbook on AI, 2, William Kaufmann, Los Altos, CA, 1982.
- [50] E. Charniak, C. Riesbeck, and D. McDermott, AI Programming, Lawrence Erlbaum Press, 1980.
- [51] D. Bobrow and B. Paphael, "New Programming Languages for AI research," Computing Surveys, vol. 6, no. 3, pp. 153-174, ACM, 1974.
- [52] E. Rich, "The Gradual Expansion of AI," Computer, vol. 17, no. 5, pp. 4-12, IEEE, May 1984.
- [53] P. H. Winston and B. Horn, Lisp, Second Edition, Addison Wesley, 1984.
- [54] J. Backus, "Can Programming be Liberated from the von Neumann Style? A Functional Style and Algebra of Programs," Comm. of the ACM, vol. 21, no. 8, pp. 613-641, ACM, 1978.
- [55] T. Winograd, "Beyond Programming Languages," Comm. of the ACM, vol. 22, no. 7, pp. 391-401, ACM, July 1979.
- [56] B. D. Kornman, "Pattern Matching and Pattern-Directed Invocation in Systems Programming Languages," J. of Systems and Software, vol. 3, pp. 95-102, Hayden Publishing, 1983.
- [57] S. Eisenbach and C. Sadler, "Declarative Languages: an Overview," Byte, pp. 181-197, McGraw-Hill, Aug. 1985.
- [58] J. Backus, "Function-Level Computing," Spectrum, vol. 19, no. 8, pp. 22-27, IEEE, Aug. 1982.
- [59] P. Henderson, Function Programming, Application and Implementation, Prentice-Hall, 1980.
- [60] R. Kowalski, "Logic Programming," in IFIP Information Processing, ed. R. E. A. Mason, pp. 133-145, Elsevier, 1983.
- [61] J. A. Robinson, "Logic Programming--Past, Present and Future," New Generation Computing, vol. 1, no. 2, pp. 107-124, OHMSHA Ltd. and Springer-Verlag, 1983.
- [62] K. L. Clark and S-A. Tarnlund, ed., Logic Programming, Academic Press, 1982.
- [63] R. Kowalski, "Predicate Logic as a Programming Language," IFIP Information Processing, pp. 569-574, North-Holland, 1974.
- [64] R. Kowalski, Logic for Problem Solving, North-Holland, 1979.
- [65] R. Burstall, "Programming with Modules as Typed Functional Programming," Proc. Int'l Conf. on 5th Generation Computers System, pp. 103-112, ICOT and North-Holland, 1984.
- [66] T. Ida and J. Tanaka, "Functional Programming with Streams--Part II," New Generation Computing, vol. 2, no. 3, pp. 261-275, OHMSHA Ltd. and Springer-Verlag, 1984.
- [67] D. A. Turner, "A New Implementation Technique for Applicative Languages," Software--Practice and Experience, vol. 9, no. 1, pp. 31-49, John Wiley & Son, 1979.
- [68] A. L. Davis and R. M. Keller, "Data Flow Program Graphs," Computer, vol. 15, no. 2, pp. 26-41, IEEE, 1982.
- [69] J. McCarthy, P. Abrahams, D. Edwards, T. Hart, and M. Levin, Lisp 1.5 Programmer's Manual, MIT Press, 1962.
- [70] J. McCarthy, "History of Lisp," SIGPLAN Notices, vol. 13, no. 8, pp. 217-223, ACM, 1978.
- [71] E. Sandewall, "Programming in an Interactive Environment: the Lisp Experience," Computing Surveys, vol. 10, no. 1, pp. 35-71, ACM, March 1978.
- [72] M. L. Griss and E. Benson, "Current Status of a Portable Lisp Compiler," Proc. SIGPLAN Symp. on Compiler Construction, pp. 276-283, ACM, June 1982.
- [73] H. G. Baker, Jr., "Shallow Binding in Lisp 1.5," Comm. of the ACM, vol. 21, no. 7, pp. 565-569, ACM, July 1978.
- [74] R. M. Burstall, D. B. MacQueen, and D. T. Sannella, "HOPE: An Experimental Applicative Language," Conf. Record of Lisp Conf., pp. 136-143, Stanford Univ., Menlo Park, CA, 1980.
- [75] J. R. McGraw, "Data Flow Computing: Software Development," Trans. on Computers, vol. C-29, no. 12, pp. 1095-1103, IEEE, 1980.

- [76] Arvind, K. Gostelow, and W. Plouffe, *An Asynchronous Programming Language and Computing Machine*, Tech. Rep. 114a, Univ. of California, Irvine, CA, Dec. 1978.
- [77] A. Colmerauer, "Prolog in 10 Figures," *Proc. 8th IJCAI*, pp. 488-499, William Kaufman, Los Altos, CA, 1983.
- [78] W. F. Clocksin and C. S. Mellish, *Programming in Prolog*, Springer-Verlag, 1981.
- [79] K. L. Clark and F. G. McCabe, "Prolog: A Language for Implementing Expert Systems," in *Machine Intelligence 10*, ed. J. Hayes, D. Michie, and Y. H. Pao, pp. 455-471, Ellis Horwood Ltd., Chichester, England, 1982.
- [80] A. Colmerauer, H. Kanoui, and M. Van Caneghem, "Last Steps Towards an Ultimate Prolog," *Proc. 7th IJCAI*, pp. 947-948, William Kaufman, Los Altos, CA, Aug. 1981.
- [81] B. Domolki and P. Szeredi, "Prolog in Practice," in *IFIP Information Processing*, ed. R. E. A. Mason, pp. 627-636, Elsevier, 1983.
- [82] D. H. Warren, L. M. Pereira, and F. Perelra, "Prolog--The Language and its Implementation Compared with Lisp," *Proc. Symp. on Aland Programming Languages*, also *SIGART Newsletter*, vol. 64, pp. 109-115, ACM, Aug. 1977.
- [83] G. J. Sussman and D. V. McDermott, "From PLANNER to CONNIVER--A Genetic Approach," *FJCC*, vol. 41, pp. 129-137, AFIPS Press, 1972.
- [84] T. Moto-oka and H. S. Stone, "5th-Generation Computer Systems: A Japanese Project," *Computer*, vol. 17, no. 3, pp. 6-13, IEEE, March 1984.
- [85] T. Rentsch, "Object Oriented Programming," *SIGPLAN Notices*, vol. 17, no. 9, pp. 51-57, ACM, Sept. 1982.
- [86] A. J. Goldberg and D. Robson, *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley, 1983.
- [87] "Special Issue on Smalltalk," *Byte*, McGraw-Hill, Aug. 1981.
- [88] F. Mizoguchi, H. Ohwada, and Y. Katayama, "LOOKS: Knowledge Representation System for Designing Expert Systems in a Logic Programming Framework," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 606-612, ICOT and North-Holland, 1982.
- [89] N. Suzuki, K. Kubota, and T. Aoki, "SWORD32: A Bytecode Emulating Microprocessor for Object-Oriented Languages," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 389-397, ICOT and North-Holland, 1984.
- [90] D. Ungar, R. Blau, P. Foley, D. Samples, and D. Patterson, "Architecture of SOAR: Smalltalk on RISC," *Proc. 11th Annual Int'l Symp. on Computer Architecture*, pp. 188-197, IEEE/ACM, 1984.
- [91] D. Moon, *MacLisp Reference Manual*, MIT Press, 1974.
- [92] B. Sheil, "Power Tools for Programmers," *Datamation*, pp. 131-144, Technical Publishing, Feb. 1983.
- [93] W. Teitelman and L. Masinter, "The Interlisp Programming Environment," *Computer*, vol. 14, no. 4, pp. 25-33, IEEE, April 1981.
- [94] E. D. Sacerdoti, R. E. Fikes, R. Reboh, D. Sagalowicz, R. J. Waldinger, and B. M. Wilber, "Ollisp--A Language for the Interactive Development of Complex Systems," *Proc. NCC*, pp. 139-146, AFIPS Press, 1976.
- [95] G. L. Steele, Jr., "An Overview of Common Lisp," *Conf Record of the 1982 Symp. on Lisp and Function Programming*, pp. 98-107, ACM, 1982.
- [96] R. Wilensky, *Lispcraft*, W. W. Norton and Co., New York, N.Y., 1984.
- [97] A. Newell, J. C. Shaw, and H. A. Simon, "Programming the Logic Theory Machine," *Prof. 1957 WJCC*, pp. 230-240, IRE, 1957.
- [98] A. Newell, J. C. Shaw, and H. A. Simon, "Empirical Explorations with the the Logic Theory Machine," in *Computers and Thought*, ed. E. A. Feigenbaum and J. Feldman, pp. 109-133, 1963.
- [99] C. Hewitt, *Description and Theoretical Analysis (Using Schemas) of PLANNER: A Language for Proving Theorems and Manipulating Models in Robots*, Doctoral Dissertation, AI Lab., MIT, 1971.
- [100] D. G. Bobrow and T. Winograd, "An Overview of KRL--A Knowledge Representation Language," *Cognitive Science*, vol. 1, no. 1, pp. 3-46, Ablex Publishing, 1976.
- [101] S. Fahlman, *NETL: A System for Representing and Using Real-World Knowledge*, Series on AI, MIT Press, 1979.
- [102] J. F. Reiser, ed., *SAIL*, Technical Report STAN-CS-76-574, Computer Science Dept. Stanford Univ., Menlo Park, CA, 1976.
- [103] D. Davies, et al., *POPLER 1.5 Reference Manual*, Univ. of Edingburgh, Edinburgh, England, 1973.
- [104] R. A. Le Faivre, *FUZZY Reference Manual*, Computer Science Dept., Rutgers Univ., New Brunswick, NJ, 1977.
- [105] M. Sato and T. Sakurai, "QUTE: A Prolog/Lisp Type Language for Logic Programming," *Proc. 8th IJCAI*, pp. 507-513, William Kaufman, Los Altos, CA, Aug. 1983.
- [106] K. Clark and S. Gregory, "Note on System Programming in PARLOG," *Proc. Int'l Conf. 5th Generation Computer System*, pp. 299-306, ICOT and North-Holland, 1984.
- [107] E. Y. Shapiro, *A Subset of Concurrent Prolog and its Interpreter*, Technical Report TR-003, ICOT, Tokyo, Japan, 1984.
- [108] E. Y. Shapiro, "Object Oriented Programming in Concurrent Prolog," *New Generation Computing*, vol. 1, no. 1, pp. 25-48, OHMSHA Ltd. and Springer-Verlag, 1983.
- [109] K. Tabata, S. Sugimoto, and Y. Ohno, "Concurrent Lisp and its Interpreter," *J. of Information Processing*, vol. 4, no. 4, Information Processing Society of Japan, Feb. 1982.
- [110] S. Sugimoto, K. Tabata, K. Agusa, and Y. Ohno, "Concurrent Lisp on a Multi-Micro-Processor System," *Proc. 7th IJCAI*, pp. 949-954, William Kaufman, Los Altos, CA, Aug. 1981.
- [111] D. Barstow, "A Perspective on Automatic Programming," *Proc. 8th IJCAI*, pp. 1170-1179, William Kaufman, Los Altos, CA, Aug. 1983.
- [112] E. J. Lerner, "Automating Programming," *Spectrum*, vol. 19, no. 8, pp. 28-33, IEEE, Aug. 1982.

- [113] D. R. Barstow, "An Experiment in Knowledge-Based Automatic Programming," in *Readings in AI*, ed. B. L. Webber and N. J. Nilsson, pp. 289-312, Tioga, 1981.
- [114] Z. Manna and R. Waldinger, "A Deductive Approach to Program Synthesis," *Proc. 6th IJCAI*, pp. 542-551, William Kaufman, Los Altos, CA, 1979.
- [115] D. R. Smith, "A Design for an Automatic Programming System," *Proc. 7th IJCAI*, pp. 1027-1029, William Kaufman, Los Altos, CA, Aug. 1981.
- [116] E. A. Feigenbaum, "Knowledge Engineering: The Applied Side," in *Intelligent Systems: The Unprecedented Opportunity*, ed. J. E. Hayes and D. Michie, pp. 37-55, Ellis Horwood Ltd., Chichester, England, 1983.
- [117] D. B. Lenat, "Computer Software for Intelligent Systems," *Scientific American*, vol. 251, no. 3, pp. 204-213, Scientific American Inc., Sept. 1984.
- [118] F. Hayes-Roth, "The Knowledge-Based Expert System: A Tutorial," *Computer*, vol. 17, no. 9, pp. 11-28, IEEE, Sept. 1984.
- [119] F. Hayes-Roth, "Knowledge-Based Expert Systems," *Computer*, vol. 17, no. 10, pp. 263-273, IEEE, Oct. 1984.
- [120] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, *Building Expert Systems*, Addison-Wesley, 1983.
- [121] B. G. Buchanan, "New Research on Expert Systems," in *Machine Intelligence 10*, ed. J. Hayes, D. Michie, and Y.-H. Pao, pp. 269-299, Ellis Horwood Ltd., Chichester, England, 1982.
- [122] W. B. Gevarter, *An Overview of Expert Systems*, Tech. Rep. NBSIR 82-2505, Nat'l Bureau of Standards, Washington, DC, 1982.
- [123] D. S. Nau, "Expert Computer System," *Computer*, vol. 16, no. 2, pp. 63-85, IEEE, Feb. 1983.
- [124] R. Davis, "Expert Systems: Where Are We? And Where Do We Go From Here?," *The AI Magazine*, pp. 3-22, AAAI, Spring 1982.
- [125] A. S. Cromarty, "What Are Current Expert System Tools Missing?," *Proc. COMPCON Spring*, pp. 411-418, IEEE, 1985.
- [126] V. P. Kobler, "Overview of Tool for Knowledge Base Construction," *Proc. Data Engineering Conf.*, pp. 282-285, IEEE, 1984.
- [127] Van Melle, E. H. Shortliffe, and B. G. Buchanan, "EMYCIN: A Domain-Independent System that Aids in Constricting Knowledge-Based Consultation Programs," *Machine Intelligence: Infotech State of the Art Report 9*, Infotech International, London, England, 1981.
- [128] Van Melle, A. C. Scott, J. S. Bennett, and M. Peairs, *The EMYCIN Manual*, Tech. Report HPP-81-16, Computer Science Dept., Stanford Univ., Menlo Park, CA, 1981.
- [129] L. Erman, P. London, and S. Fickas, "The Design and Example Use of HEARSAY-III," *Proc. 7th IJCAI*, pp. 409-415, William Kaufman, Los Altos, CA, 1981.
- [130] H. P. Nii and N. Aiello, "AGE (Attempt to Generalize): A Knowledge-Based Program for Building Knowledge-Based Programs," *Proc. 6th IJCAI*, pp. 645-655, William Kaufman, Los Altos, CA, Aug. 1979.
- [131] C. Forgy and J. McDermott, "OPS--A Domain-Independent Production Systems Language," *Proc. 5th IJCAI*, pp. 933-939, William Kaufman, Los Altos, CA, 1977.
- [132] M. Stefik, D. Bobrow, S. Mittal, and L. Conway, "Knowledge Programming in LOOPS: Report on an Experimental Course," *The AI Magazine*, pp. 20-30, AAAI, Fall 1983.
- [133] J. Fain and F. Hayes-Roth, et al., *Programming in ROSIE: An Introduction by Means of Examples*, Tech. Note N-1646-ARPA, Rand Corp., Santa Monica, CA, 1982.
- [134] R. Greiner and D. Lenat, "A Representation Language," *Proc. First Nat'l Conf. on AI*, pp. 165-169, William Kaufman, Los Altos, CA, 1980.
- [135] R. Davis and B. Buchanan, "Meta-level Knowledge: Overview and Applications," *Proc. 5th IJCAI*, pp. 920-928, William Kaufman, Los Altos, CA, 1977.
- [136] S. M. Weiss and C. A. Kulikowski, "EXPERT: A System for Developing Consulting Models," *Proc. 6th IJCAI*, pp. 942-947, William Kaufman, Los Altos, CA, 1979.
- [137] R. O. Duda, J. G. Gaschnig, and P. E. Hart, "Model Design in the PROSPECTOR Consultant System for Mineral Exploration," in *Expert System in the Micro-Electronics Age*, Edingburgh Univ. Press, Edingburgh, England, 1979.
- [138] B. G. Buchanan and E. A. Feigenbaum, "Dendral and Meta-Dendral: Their Applications Dimension," *AI*, vol. 11, no. 1-2, pp. 5-24, North-Holland, 1978.
- [139] D. B. Lenat and J. S. Brown, "Why AM and EURISKO Appear to Work?," *AI*, vol. 23, no. 3, pp. 269-294, North-Holland, 1984.
- [140] C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.
- [141] B. P. Treleaven and C. Phillip, ed., *VLSI Architectures*, Prentice-Hall, 1983.
- [142] C. L. Seitz, ed., *Proc. Caltech Conf. on Very Large Scale Integration*, Caltech, Pasadena, CA, Jan. 1979.
- [143] *Proc. 2nd Caltech Conf. on Very Large Scale Integration*, Computer Science Press, 1981.
- [144] R. Bryant, ed., *Proc. 3rd Caltech Conf. on Very Large Scale Integration*, Computer Science Press, 1983.
- [145] P. C. Treleaven, "VLSI Processor Architectures," *Computer*, vol. 15, no. 6, pp. 33-45, IEEE, June 1982.
- [146] T. Williams, "Semiconductor Memories: Density and Diversity," *Computer Design*, pp. 105-116, PennWell, Aug. 1984.
- [147] H. T. Kung, "Let's Design Algorithms for VLSI Systems," in *Proc. Caltech Conf. on VLSI*, ed. C. L. Seitz, pp. 65-90, Caltech, Pasadena, CA, Jan. 1979.
- [148] M. J. Foster and H. T. Kung, "The Design of Special-Purpose VLSI Chips," *Computer*, vol. 13, no. 1, pp. 26-40, IEEE, Jan. 1980.
- [149] J. Grinberg, G. R. Nudd, and R. D. Etchells, "A Cellular VLSI Architecture," *Computer*, vol. 17, no. 1, pp. 69-81, IEEE, Jan. 1984.

- [150] C. L. Seitz, "Concurrent VLSI Architectures," *Trans. on Computers*, vol. C-33, no. 12, pp. 1247-1265, IEEE, Dec. 1984.
- [151] J. A. B. Fortes, K. S. Fu, and B. W. Wah, "Systematic Approaches to the Design of Algorithmically Specified Systolic Arrays," *Proc. Int'l Conf. on Acoustics, Speech, and Signal Proc.*, pp. 8.9.1-8.9.4, IEEE, 1985.
- [152] S. E. Fahlman and G. E. Hinton, "Massively Parallel Architectures for AI: NETL, THISTLE, and BOLTZMANN Machines," *Proc. Nat'l Conf. on AI*, pp. 109-113, AAAI, 1983.
- [153] D. Bitton, D. J. DeWitt, D. K. Hsiao, and J. Menon, "A Taxonomy of Parallel Sorting," *Computing Surveys*, vol. 16, no. 3, pp. 287-318, ACM, Sept. 1984.
- [154] J. D. Ullman, "Some Thoughts About Supercomputer Organization," *Proc. COMPCON Spring*, pp. 424-432, IEEE, Feb. 1984.
- [155] C. D. Thompson and H. T. Kung, "Sorting on a Mesh-Connected Parallel Computer," *Comm. of the ACM*, vol. 20, no. 4, pp. 263-271, ACM, April 1977.
- [156] C. D. Thompson, "The VLSI Complexity of Sorting," *Trans. on Computers*, vol. C-32, no. 12, pp. 1171-1184, IEEE, Dec. 1983.
- [157] C. C. Hsiao and L. Snyder, "Omni-Sort: A Versatile Data Processing Operation for VLSI," *Proc. Int'l Conf. on Parallel Processing*, pp. 222-225, IEEE, 1983.
- [158] L. E. Winslow and Y. C. Chow, "The Analysis and Design of Some New Sorting Machines," *Trans. on Computers*, vol. C-32, no. 7, pp. 677-683, IEEE, July 1983.
- [159] M. A. Bonuccelli, E. Lodi, and L. Pagli, "External Sorting in VLSI," *Trans. on Computers*, vol. C-33, no. 10, pp. 931-934, IEEE, Oct. 1984.
- [160] G. Baudet and D. Stevenson, "Optimal Sorting Algorithms for Parallel Computers," *Trans. on Computers*, vol. C-27, no. 1, pp. 84-87, IEEE, Jan. 1978.
- [161] F. P. Preparata, "New Parallel-Sorting Schemes," *Trans. on Computers*, vol. C-27, no. 7, pp. 669-673, IEEE, July 1978.
- [162] C. P. Kruskal, "Searching, Merging, and Sorting in Parallel Computation," *Trans. on Computers*, vol. C-32, no. 10, pp. 942-946, IEEE, Oct. 1983.
- [163] B. W. Wah and K. L. Chen, "A Partitioning Approach to the Design of Selection Networks," *Trans. on Computers*, vol. C-33, no. 3, pp. 261-268, IEEE, March 1984.
- [164] A. C. C. Yao, "Bounds on Selection Networks," *SIAM J. on Computing*, vol. 9, no. 3, pp. 566-582, SIAM, Aug. 1980.
- [165] L. J. Guibas, H. T. Kung, and C. D. Thompson, "Direct VLSI Implementation of Combinatorial Algorithms," *Proc. Caltech Conf. on VLSI*, pp. 509-525, Caltech, Pasadena, CA, 1979.
- [166] P. A. V. Hall and G. R. Dowling, "Approximate String Matching," *Computing Surveys*, vol. 12, no. 4, pp. 381-402, ACM, Dec. 1980.
- [167] C. Hoffmann and M. O'Donnell, "Pattern Matching in Trees," *J. of the ACM*, vol. 21, no. 1, pp. 68-95, ACM, 1982.
- [168] A. Apostolico and A. Negro, "Systolic Algorithms for String Manipulations," *Trans. on Computers*, vol. C-33 no. 4, pp. 361-364, IEEE, April 1984.
- [169] J. Aoe, Y. Yamamoto, and R. Shimada, "A Method for Improving String Pattern Matching Machines," *Trans. on Software Engineering*, vol. SE-10, no. 1, pp. 116-120, IEEE, Jan. 1984.
- [170] S. R. Ahuja and C. S. Roberts, "An Associative/Parallel Processor for Partial Match Retrieval Using Superimposed Codes," *Proc. 7th Annual Symp. on Computer Architecture*, pp. 218-227, IEEE/ACM, May 1980.
- [171] K. Goser, C. Foelster, and U. Rueckert, "Intelligent Memories in VLSI," *Information Sciences*, vol. 34, no. 1, pp. 61-82, Elsevier, 1984.
- [172] S. L. Tanimoto, "A Boolean Matching Operator for Hierarchical Cellular Logic," *Proc. Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pp. 253-256, IEEE, Oct. 1983.
- [173] R. Gonzalez-Rubio, J. Rohmer, and D. Terral, "The Schuss Filter: A Processor for Non-Numerical Data Processing," *Proc. 11th Annual Int'l Symp. on Computer Architecture*, pp. 64-73, IEEE/ACM, June 1984.
- [174] H. C. Du, "Concurrent Disk Accessing for Partial Match Retrieval," *Proc. Int'l Conf. on Parallel Processing*, pp. 211-218, IEEE, 1982.
- [175] D. P. Bertsekas, "Distributed Dynamic Programming," *Trans. on Automatic Control*, vol. AC-27, no. 3, pp. 610-616, IEEE, June 1982.
- [176] J. Casti, M. Richardson, and R. Larson, "Dynamic Programming and Parallel Computers," *J. of Optimization Theory and Applications*, vol. 12, no. 4, pp. 423-438, Plenum Press, Nov. 1973.
- [177] P. N. Yianilos, "Dedicated Comparator Matches Symbol Strings Fast and Intelligently," *Electronics*, pp. 113-117, McGraw-Hill, 1983.
- [178] R. J. Douglass, "A Qualitative Assessment of Parallelism in Expert Systems," *Software*, vol. 2, no. 2, pp. 70-81, IEEE, May 1985.
- [179] C. Forgy, A. Gupta, A. Newell, and R. Wedig, "Initial Assessment of Architectures for Production Systems," *Proc. Nat'l Conf. on AI*, pp. 116-120, AAAI, Aug. 1984.
- [180] E. Tick and D. H. D. Warren, "Towards a Pipelined Prolog Processor," *New Generation Computing*, vol. 2, no. 4, pp. 323-345, OHMSHA Ltd. and Springer-Verlag, 1984.
- [181] J. S. Vitter and R. A. Simons, "Parallel Algorithms for Unification and Other Complete Problems," *Proc. ACM'84*, pp. 75-84, ACM, Oct. 1984.
- [182] J. S. Vitter and R. A. Simons, "Parallel Algorithms for Unification and Other Complete Problems," *Trans. on Computers*, IEEE, to appear 1986.
- [183] G. L. Steele Jr. and G. J. Sussman, "Design of a Lisp-Based Microprocessor," *Comm. of the ACM*, vol. 23, no. 11, pp. 628-645, ACM, Nov. 1980.
- [184] E. A. Feustel, "On the Advantages of Tagged Architecture," *Trans. on Computers*, vol. C-22, no. 7, pp. 644-656, IEEE, 1973.

- [185] M. J. Atallah and S. R. Kosaraju, "A Generalized Dictionary Machine for VLSI," *Trans. on Computers*, vol. C-34, no. 2, pp. 151-155, IEEE, Feb. 1985.
- [186] H. Schmeck and H. Schroder, "Dictionary Machines for Different Models of VLSI," *Trans. on Computers*, vol. C-34, no. 5, pp. 472-475, IEEE, May 1985.
- [187] A. L. Fisher, "Dictionary Machines with a Small Number of Processors," *Proc. 11th Annual Int'l Symp. on Computer Architecture*, pp. 151-156, IEEE/ACM, June 1984.
- [188] A. K. Somani and V. K. Agarwal, "An Efficient VLSI Dictionary Machine," *Proc. 11th Annual Int'l Symp. on Computer Architecture*, pp. 142-150, IEEE/ACM, June 1984.
- [189] T. A. Ottmann, A. L. Rosenberg, and L. J. Stockmeyer, "A Dictionary Machine (for VLSI)," *Trans. on Computers*, vol. C-31, no. 9, pp. 892-897, IEEE, Sept. 1982.
- [190] M. J. Carey and C. D. Thompson, "An Efficient Implementation of Search trees on $O(\log N)$ Processors," *Tech. Rep. UCB/CSD 82/101*, Computer Science Division Univ. of California, Berkeley, CA, April 1982.
- [191] C. E. Leiserson, "Systolic Priority Queues," *Proc. Caltech Conf. on VLSI*, Caltech, Jan. 1979.
- [192] J. L. Bentley and H. T. Kung, "A Tree Machine for Searching Problems," *Proc. Int'l Conf. on Parallel Processing*, pp. 257-266, IEEE, 1979.
- [193] S. W. Miller, ed., "Special Issue on Mass Storage Systems," *Computer*, vol. 18, no. 7, IEEE, July 1985.
- [194] S. W. Miller, ed., "Special Issue on Mass Storage Systems Evolution of Data Center Architectures," *Computer*, vol. 15, no. 7, IEEE, July 1982.
- [195] H. A. Freeman, ed., "Special Issue on Backend Storage Networks," *Computer*, vol. 13, no. 2, IEEE, Feb. 1980.
- [196] D. K. Hsiao, ed., "Special Issue on Database Machines," *Computer*, vol. 12, no. 3, IEEE, March 1979.
- [197] G. G. Langdon Jr., ed., "Special Issue on Database Machines," *Trans. on Computers*, vol. C-28, no. 6, IEEE, June 1979.
- [198] H. Boral and D. DeWitt, "Database Machine: An Idea Whose Time has Passed?," *Database Machines*, pp. 166-167, Springer-Verlag, 1983.
- [199] F. J. Malabarba, "Review of Available Database Machine Technology," *Proc. Trends and Applications*, pp. 14-17, IEEE, 1984.
- [200] J. Shemer and P. Neches, "The Genesis of a Database Computer," *Computer*, vol. 17, no. 11, pp. 42-56, IEEE, Nov. 1984.
- [201] P. B. Hawthorn and D. J. DeWitt, "Performance Analysis of Alternative Database Machine Architectures," *Trans. on Software Engineering*, vol. SE-8, no. 1, pp. 61-75, IEEE, Jan. 1982.
- [202] E. Babb, "Joined Normal Form: A Storage Encoding for Relational Databases," *Trans. on Database Systems*, vol. 7, no. 4, pp. 588-614, ACM, Dec. 1982.
- [203] D. Gajski, W. Kim, and S. Fushimi, "A Parallel Pipelined Relational Query Processor: An Architectural Overview," *Proc. 11th Annual Int'l Symp. on Computer Architecture*, pp. 134-141, IEEE/ACM, June 1984.
- [204] M. Kitsuregawa, H. Tanaka, and T. Moto-oka, "Application of Hash to Data Base Machine and its Architecture," *New Generation Computing*, vol. 1, no. 1, pp. 63-74, OHMSHA Ltd. and Springer-Verlag, 1983.
- [205] D. E. Shaw, "Knowledge-Based Retrieval on a Relational Database Machine," Ph.D. Dissertation, Stanford Univ., Menlo Park, CA; also as Technical Report, Columbia Univ., New York, NY, Aug. 1980.
- [206] Y. Tanaka, "MPDC-Massive Parallel Architecture for Very Large Databases," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 113-137, ICOT and North-Holland, 1984.
- [207] K. Murakami, T. Kakuta, and R. Onai, "Architectures and Hardware Systems: Parallel Inference Machine and Knowledge Base Machine," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 18-36, ICOT and North-Holland, 1984.
- [208] S. Shibayama, T. Kakuta, N. Miyazaki, H. Yokota, and K. Murakami, "A Relational Database Machine with Large Semiconductor Disk and Hardware Relational Algebra Processor," *New Generation Computing*, vol. 2, no. 2, pp. 131-155, OHMSHA Ltd. and Springer-Verlag, 1984.
- [209] C. Kellogg, "Knowledge Management: A Practical Amalgam of Knowledge and Data Base Technology," *Proc. Nat'l Conf. on AI*, pp. 306-309, AAAI, 1982.
- [210] C. Kellogg, "Intelligent Assistants for Knowledge and Information Resources Management," *Proc. 8th IJCAI*, pp. 170-172, William Kaufman, Los Altos, CA, 1983.
- [211] P. M. Neches, "Hardware Support for Advanced Data Management Systems," *Computer*, vol. 17, no. 11, pp. 29-40, IEEE, Nov. 1984.
- [212] W. A. Kornfeld, "The Use of Parallelism to Implement a Heuristic Search," *Proc. 7th IJCAI*, pp. 575-580, William Kaufman, Los Altos, CA, Aug. 1981.
- [213] W. A. Kornfeld, "Combinatorially Implosive Algorithms," *Comm. of the ACM*, vol. 25, no. 10, pp. 734-738, ACM, Oct. 1982.
- [214] B. W. Weide, "Modeling Unusual Behavior of Parallel Algorithms," *Trans. on Computers*, vol. C-31, no. 11, pp. 1126-1130, IEEE, Nov. 1982.
- [215] E. Horowitz and A. Zorat, "Divide-and-Conquer for Parallel Processing," *Trans. on Computers*, vol. C-32, no. 6, pp. 582-585, IEEE, June 1983.
- [216] M. A. Franklin and N. L. Soong, "One-Dimensional Optimization on Multiprocessor Systems," *Trans. on Computers*, vol. C-30, no. 1, pp. 61-66, IEEE, Jan. 1981.
- [217] S. G. Akl, D. T. Barnard, and R. J. Doran, "Design, Analysis and Implementation of a Parallel Tree Search Algorithm," *Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, no. 2, pp. 192-203, IEEE, March 1982.
- [218] E. Dekel and S. Sahni, "Binary Trees and Parallel Scheduling Algorithms," *Trans. on Computers*, vol. C-32, no. 3, pp. 307-315, IEEE, March 1983.
- [219] J. L. Baer, H. C. Du, and R. E. Ladner, "Binary Search in a Multiprocessing Environment," *Trans. on Computers*, vol. C-32, no. 7, pp. 667-677, IEEE, July 1983.

- [220] T. H. Lai and S. Sahni, "Anomalies in Parallel Branch and-Bound Algorithms," *Comm. of the ACM*, vol. 27, no. 6, pp. 594-602, ACM, June 1984.
- [221] G.-J. Li and B. W. Wah, "Computational Efficiency of Parallel Approximate Branch-and-Bound Algorithms," *Proc. Int'l Conf. on Parallel Processing*, pp. 473-480, IEEE, 1984.
- [222] G.-J. Li and B. W. Wah, "Coping with Anomalies in Parallel Branch-and-Bound Algorithms," *Trans. on Computers*, vol. C-35, no. 4, IEEE, April 1986.
- [223] T. H. Lai and A. Sprague, "Performance of Parallel Branch-and-Bound Algorithms," *Trans. on Computers*, vol. C-34, no. 10, pp. 962-964, IEEE, Oct. 1985.
- [224] D. H. Fishman and J. Minker, "II-Representation: A Clause Representation for Parallel Search," *AI*, vol. 6, no. 2, pp. 103-127, North-Holland, 1975.
- [225] R. A. Finkel and J. P. Fishburn, "Parallelism in Alpha-Beta Search," *AI*, vol. 19, no. 1, pp. 89-106, North-Holland, 1982.
- [226] T. A. Marsland and M. Campbell, "Parallel Search of Strongly Ordered Game Trees," *Computing Surveys*, vol. 14, no. 4, pp. 533-551, ACM, Dec. 1982.
- [227] F. J. Peters, "Tree Machine and Divide-and-Conquer Algorithms," *Lecture Notes CS 111 (CONPAR81)*, pp. 25-35, Springer-Verlag, 1981.
- [228] M. R. Sleep and F. W. Burton, "Towards a Zero Assignment Parallel Processor," *Proc. 2nd Int'l Conf. on Distributed Computing Systems*, pp. 80-85, IEEE, April 1981.
- [229] J. A. Harris and D. R. Smith, "Simulation Experiments of a Tree Organized Multicomputer," *Proc. 6th Annual Symp. on Computer Architecture*, pp. 83-89, IEEE/ACM, April 1979.
- [230] M. Imai and T. Fukumura, "A Parallelized Branch-and-Bound Algorithm Implementation and Efficiency," *Systems, Computers, Controls*, vol. 10, no. 3, pp. 62-70, Scripta Publishing, June 1979.
- [231] B. C. Desai, "A Parallel Microprocessing System," *Proc. Int'l Conf. on Parallel Processing*, p. 136, IEEE, Aug. 1979.
- [232] O. I. El-Dessouki and W. H. Huen, "Distributed Enumeration on Between Computers," *Trans. on Computers*, vol. C-29, no. 9, pp. 818-825, IEEE, Sept. 1980.
- [233] W. M. McCormack, F. G. Gray, J. G. Tront, R. M. Haralick, and G. S. Fowler, "Multi-Computer Parallel Architectures for Solving Combinatorial Problems," in *Multicomputers and Image Processing Algorithms and Programs*, ed. K. Preston Jr. and L. Uhr, pp. 431-451, Academic Press, 1982.
- [234] M. Imai, Y. Tateizumi, Y. Yoshida, and T. Fukumura, "A Multicomputer System Based on the Binary-Tree Structure: DON(2)," *TGEC*, vol. EC83-23, no. 1, pp. 19-30, IECE of Japan, 1983.
- [235] Q. F. Stout, "Sorting, Merging, Selecting and Filtering on Tree and Pyramid Machines," *Proc. Int'l Conf. on Parallel Processing*, pp. 214-221, IEEE, Aug. 1983.
- [236] B. W. Wah, G.-J. Li, and C. F. Yu, "The Status of MANIP--A Multicomputer Architecture for Solving Combinatorial Extremum-Search Problems," *Proc. 11th Annual Int'l Symp. on Computer Architecture*, pp. 56-63, IEEE/ACM, June 1984.
- [237] B. W. Wah and Y. W. E. Ma, "MANIP--A Multicomputer Architecture for Solving Combinatorial Extremum-Search Problems," *Trans. on Computers*, vol. C-33, no. 5, pp. 377-390, IEEE, May 1984.
- [238] R. Finkel and U. Manber, "DIB--A Distributed Implementation of Backtracking," *Proc. 5th Int'l Conf. on Distributed Computing Systems*, pp. 446-452, IEEE, May 1985.
- [239] E. W. Dijkstra, L. Lamport, A. J. Martin, C. S. Scholten, and E. F. M. Steffens, "On-the-Fly Garbage Collection: An Exercise in Cooperation," *Comm. of the ACM*, vol. 21, no. 11, pp. 966-975, ACM, Nov. 1978.
- [240] H. G. Baker Jr., "List Processing in Real Time on a Serial Computer," *Comm. of the ACM*, vol. 21, no. 4, pp. 280-294, April 1978.
- [241] H. C. Baker Jr. and C. Hewitt, "The Incremental Garbage Collection of Processes," *Proc. Symp. on AI and Programming Languages*, also *SIGART Newsletter*, pp. 55-59, ACM, Aug. 1977.
- [242] J. J. Martin, "An Efficient Garbage Compaction Algorithm," *Comm. of the ACM*, vol. 25, no. 8, pp. 571-581, ACM, Aug. 1982.
- [243] D. Spector, "Minimal Overhead Garbage Collection of Complex List Structure," *SIGPLAN Notices*, vol. 17, no. 3, pp. 80-82, ACM, 1982.
- [244] Y. Hibino, "A Practical Parallel Garbage Collection Algorithm and Its Implementations," *Proc. 7th Annual Symp. on Computer Architecture*, pp. 113-120, IEEE/ACM, May 1980.
- [245] R. Fenichel and J. Yochelson, "A Lisp Garbage-Collector for Virtual Memory Computer Systems," *Comm. of the ACM*, vol. 12, no. 11, pp. 611-612, ACM, Nov. 1979.
- [246] J. M. Barth, "Shifting Garbage Collection Overhead to Compile Time," *Comm. of the ACM*, vol. 20, no. 7, pp. 513-518, ACM, July 1977.
- [247] H. Kung and S. Song, "An Efficient Parallel Garbage Collection Systems and Its Correctness Proof," *Technical Report, Department of Computer Science, Carnegie-Mellon Univ., Pittsburgh, PA*, Sept. 1977.
- [248] L. Deutsch and D. Bobrow, "An Efficient Incremental, Automatic Garbage Collector," *Comm. of the ACM*, vol. 19, no. 9, pp. 522-526, ACM, Sept. 1976.
- [249] D. Bobrow and D. Clark, "Compact Encoding of List Structure," *Trans. on Prog. Language and Systems*, vol. 1, no. 2, pp. 266-286, ACM, 1979.
- [250] I. A. Newman and M. C. Woodward, "Alternative Approaches to Multiprocessor Garbage Collection," *Proc. Int'l Conf. on Parallel Processing*, pp. 205-210, IEEE, 1982.
- [251] G. Steele, "Multiprocessing Compactifying Garbage Collection," *Comm. of the ACM*, vol. 18, no. 9, pp. 495-508, ACM, Sept. 1975.
- [252] J. Darlington, "Functional Programming (Chapter 5)," in *Distributed Computing*, ed. F. B. Chambers, D. A. Duce, and G. P. Jones, Academic Press, London, 1984.

- Machine?," SIGSAM Bulletin, vol. 12, no. 4, ACM, 1978.
- [254] J. Fitch, "Do We Really Want a Lisp Machine?," SEAS/SMC Annual Meeting, ACM, Jan. 1980.
- [255] E. Myers, "Machine that Lisp," Datamation, vol. 27, no. 9, pp. 105-108, Technical Publishing, Sept. 1981.
- [256] T. Knight, The CONS Microprocessor, AI Working Paper 80, MIT, Cambridge, MA, Nov. 1974.
- [257] A. Bawden, R. Greenblatt, J. Holloway, T. Knight, D. Moon, and D. Weinreb, "The Lisp Machine," in AI: An MIT Perspective, ed. P. H. Winston and R. H. Brown, vol. 1, pp. 343-373, MIT Press, 1979.
- [258] S. R. Schoichet, "The Lisp Machine," Mini-Micro Systems, pp. 68-74, Cahners Publishing, June 1978.
- [259] R. G. Greenblatt, T. F. Knight, J. T. Holloway, and D. A. Moon, "A Lisp Machine," Proc. 5th Workshop on Computer Architecture for Non-Numeric Processing, pp. 137-138, ACM, March 1980.
- [260] T. Manuel, "Lisp and Prolog Machines are Proliferating," Electronics, pp. 132-137, McGraw-Hill, Nov. 1983.
- [261] W. Myers, "Lisp Machines Displayed at AI Conf.," Computer, vol. 15, no. 11, pp. 79-82, IEEE, Nov. 1982.
- [262] T. Kurokawa, "Lisp Activities in Japan," Proc. 6th IJCAI, pp. 502-504, William Kaufman, Los Altos, CA, 1979.
- [263] D. Weinreb and D. Moon, Flavors, Message Passing in the Lisp Machine, AI Memo 602, MIT Lab, Cambridge, MA, Nov. 1980.
- [264] L. Walker, "Lisp Language Gets Special Machine," Electronics, pp. 40-41, McGraw-Hill, Aug. 25, 1981.
- [265] D. A. Moon, "Architecture of the Symbolics 3600," Proc. 12th Annual Int'l Symp. on Computer Architecture, pp. 76-83, IEEE/ACM, June 1985.
- [266] J. Moore, The Interlisp Virtual Machine Specification, Tech. Rep. CSL 76-5, Xerox PARC, Palo Alto, CA, Sept. 1976.
- [267] D. G. Bobrow, The LOOPS Manual, Tech. Rep. KB-VLSI-81-13, Xerox PARC, Palo Alto, CA, 1982.
- [268] B. Sheil, "Family of Personal Lisp Machines Speeds AI Program Development," Electronics, pp. 153-156, McGraw-Hill, Nov. 1983.
- [269] H. Hayashi, A. Hattori, and H. Akimoto, "ALPHA: A High-Performance Lisp Machine Equipped with a New Stack Structure and Garbage Collection System," Proc. 10th Annual Int'l Symp. on Computer Architecture, pp. 342-348, IEEE/ACM, June 1983.
- [270] H. Akimoto, S. Shimizu, A. Shinagawa, A. Hattori, and H. Hayashi, "Evaluation of the Dedicated Hardware in FACOM Alpha," Proc. COMPCON Spring, pp. 366-369, IEEE, 1985.
- [271] G. J. Sussman, J. Holloway, G. L. Steel Jr., and A. Bell, "Scheme-79--Lisp on a Chip," Computer, vol. 14, no. 7, pp. 10-21, IEEE, July 1981.
- [272] G. Steel and G. Sussman, Design of Lisp-Based Processor, or SCHEME: A Dielectric Lisp or Finite Memories Considered harmful, or LAMBDA: The UI-
- [273] M. Griss and M. Swanson, "MBALM/1700: A Microprogrammed Lisp Machine for the Burroughs B1726," Proc. MICRO-10, ACM/IEEE, 1977.
- [274] K. Taki, Y. Kaneda, and S. Maekawa, "The Experimental Lisp Machine," Proc. 6th IJCAI, pp. 865-867, William Kaufman, Los Altos, CA, Aug. 1979.
- [275] E. Goto, T. Ida, K. Hiraki, M. Suzuki, and N. Inada, "FLATS, A Machine for Numerical, Symbolic and Associative Computing," Proc. 6th IJCAI, pp. 1058-1066, William Kaufman, Los Altos, CA, Aug. 1979.
- [276] P. Deutsch, "Experience with a Microprogrammed Interlisp Systems," Proc. MICRO, vol. 11, ACM/IEEE, Nov. 1978.
- [277] M. Nagao, J. I. Tsujii, K. Nakajima, K. Mitamura, and H. Ito, "Lisp Machine NK3 and Measurement of its Performance," Proc. 6th IJCAI, pp. 625-627, William Kaufman, Los Altos, CA, Aug. 1979.
- [278] N. Greenfeld and A. Jericho, "A Professional's Personal Computer System," Proc. 8th Int'l Symp. on Comp. Architecture, pp. 217-226, IEEE/ACM, 1981.
- [279] J. P. Sansonnet, M. Castan, and C. Percebois, "M3L: A List-Directed Architecture," Proc. 7th Annual Symp. on Computer Architecture, pp. 105-112, IEEE/ACM, May 1980.
- [280] J. Sansonnet, D. Botella, and J. Perez, "Function Distribution in a List-Directed Architecture," Microprocessing and Microprogramming, vol. 9, no. 3, pp. 143-153, North-Holland, 1982.
- [281] J. P. Sansonnet, M. Castan, C. Percebois, D. Botella, and J. Perez, "Direct Execution of Lisp on a List-Directed Architecture," Proc. Symp. on Architectural Support for Programming Languages and Operating Systems, pp. 132-139, ACM, March 1982.
- [282] E. von Puttkamer, "A Microprogrammed Lisp Machine," Microprocessing and Microprogramming, vol. 11, no. 1, pp. 9-14, North-Holland, Jan. 1983.
- [283] R. Williams, "A Multiprocessing System for the Direct Execution of Lisp," Proc. 4th Workshop on Computer Architecture for Non-Numeric Processing, ACM, Aug. 1978.
- [284] D. McKay and S. Shapiro, "MULTI--A Lisp Based Multiprocessing System," Conf. Record of Lisp Conf., Stanford Univ., Menlo Park, CA, 1980.
- [285] M. Model, "Multiprocessing via Intercommunicating Lisp Systems," Conf. Record of Lisp Conf., Stanford Univ., Menlo Park, CA, 1980.
- [286] C. Hewitt, "The Apiary Network Architecture for Knowledgeable Systems," Conf. Record of Lisp Conf., pp. 107-117, Stanford Univ., Menlo Park, CA, 1980.
- [287] A. Guzman, "A Heterarchical Multi-Microprocessor Lisp Machine," Proc. Workshop on Computer Architecture for Pattern Analysis and Image Database Management, pp. 309-317, IEEE, Nov. 1981.
- [288] S. Sugimoto, K. Agusa, K. Tabata, and Y. Ohno, "A Multi-Microprocessor System for Concurrent Lisp," Proc. Int'l Conf. on Parallel Processing, pp. 135-143, IEEE, 1983.

- [289] Y. Yamaguchi, K. Toda, J. Herath, and T. Yuba, "EM-3: A Lisp-Based Data-Driven Machine," Proc. Int'l Conf. on 5th Generation Computer Systems, pp. 524-532, ICOT and North-Holland, 1984.
- [290] Y. Yamaguchi, K. Toda, and T. Yuba, "A Performance Evaluation of a Lisp-Based Data-Driven Machine (EM-3)," Proc. 10th Annual Int'l Symp. on Computer Architecture, pp. 363-369, IEEE/ACM, June 1983.
- [291] W. K. Giloi and R. Gueth, "Concepts and Realization of a High-Performance Data Type Architecture," Int'l J. of Computer and Information Sciences, vol. 11, no. 1, pp. 25-54, Plenum Press, 1982.
- [292] P. C. Treleaven and R. P. Hopkins, "A Recursive Computer Architecture for VLSI," Proc. 9th Annual Symp. on Computer Architecture, pp. 229-238, IEEE/ACM, April 1982.
- [293] M. Amamiya, R. Hasegawa, O. Nakamura, and H. Mikami, "A List-Processing-Oriented Data Flow Machine Architecture," Proc. NCC, pp. 144-151, AFIPS Press, 1982.
- [294] M. Amamiya and R. Hasegawa, "Dataflow Computing and Eager and Lazy Evaluations," New Generation Computing, vol. 2, no. 2, pp. 105-129, OHMSHA Ltd. and Springer-Verlag, 1984.
- [295] H. Diehl, "Concurrent Data Access Architecture," Proc. Int'l Conf. on 5th Generation Computer Systems, pp. 373-388, ICOT and North-Holland, 1984.
- [296] G. Coghill and K. Hanna, "PLEIADES: A Multimicroprocessor Interactive Knowledge Base," Microprocessors and Microsystems, vol. 3, no. 2, pp. 77-82, IPC Business Press, England, March 1979.
- [297] M. Deering, J. Faletti, and R. Wilensky, "PEARL--A Package for Efficient Access to Representations in Lisp," Proc. 7th IJCAI, pp. 930-932, William Kaufman, Los Altos, CA, Aug. 1981.
- [298] H. Samet, "Code Optimization Considerations in List Processing Systems," Trans. on Software Engineering, vol. SE-8, no. 2, pp. 107-113, IEEE, March 1982.
- [299] S. Taff, "The Design of an M6800 Lisp Interpreter," Byte, pp. 132-152, McGraw-Hill, Aug. 1979.
- [300] P. Deutsch, "ByteLisp and its Alto Implementation," Conf. Record of Lisp Conf., Stanford Univ., Menlo Park, CA, 1980.
- [301] S. P. Levitan and J. G. Bonar, "Three Microcomputer Lisps," Byte, pp. 388-412, McGraw-Hill, Sept. 1981.
- [302] T. King, "Expert Systems with 68000 and Lisp," Microprocessors and Microsystems, vol. 8, no. 7, pp. 374-376, IPC Business Press, England, Sept. 1984.
- [303] W. D. Strecker, "Clustering VAX Superminicomputers into Large Multiprocessor Systems," Electronics, pp. 143-146, McGraw-Hill, Oct. 20, 1983.
- [304] C. Hewitt, "Viewing Control Structure as Patterns of Passing Messages," AI, vol. 8, no. 3, pp. 323-364, North-Holland, 1977.
- [305] The Xerox Learning Research Group, "The Smalltalk-80 System," Byte, pp. 36-48, McGraw-Hill, Aug. 1981.
- [306] A. Plotkin and D. Tabak, "A Tree Structured Architecture for Semantic Gap Reduction," Computer Architecture News, vol. 11, no. 4, pp. 30-44, ACM SIGARCH, Sept. 1983.
- [307] S. R. Vegdahl, "A Survey of Proposed Architectures for the Execution of Functional Languages," Trans. on Computers, vol. C-33, no. 12, pp. 1050-1071, IEEE, Dec. 1984.
- [308] P. S. Abrahms, An APL Machine, Ph.D. Dissertation, Stanford Univ., Menlo Park, CA, Feb. 1970.
- [309] K. J. Berkling, "Reduction Languages for Reduction Machines," Proc. Int'l Symp. on Computer Architecture, pp. 133-140, IEEE/ACM, 1975.
- [310] M. Castan and E. I. Organick, "M3L: An HLL-RISC Processor for Parallel Execution of FP-Language Programs," Proc. 9th Annual Symp. on Computer Architecture, pp. 239-247, IEEE/ACM, 1982.
- [311] G. Mago, "A Network of Microprocessors to Execute Reduction Languages, Part I," Int'l J. of Computer and Information Sciences, vol. 8, no. 5, pp. 349-385, Plenum Press, 1979.
- [312] G. Mago, "A Network of Microprocessors to Execute Reduction Languages, Part II," Int'l J. of Computer and Information Sciences, vol. 8, no. 6, pp. 435-471, Plenum Press, 1979.
- [313] G. Mago, "Making Parallel Computation Simple: The FFP Machine," Proc. COMPCON Spring, pp. 424-428, IEEE, 1985.
- [314] R. M. Keller, G. Lindstrom, and S. Patil, "A Loosely-Coupled Applicative Multiprocessing System," Proc. NCC, pp. 613-622, AFIPS Press, 1979.
- [315] A. L. Davis, "A Data Flow Evaluation System Based on the Concept of Recursive Locality," Proc. NCC, pp. 1079-1086, AFIPS Press, 1979.
- [316] J. T. O'Donnell, A Systolic Associative Lisp Computer Architecture with Incremental Parallel Storage Management, Ph.D. Dissertation, Univ. of Iowa, Iowa City, IA, 1981.
- [317] D. P. Friedman and D. S. Wise, "Aspects of Applicative Programming for Parallel Processing," Trans. on Computers, vol. C-27, no. 4, pp. 289-296, IEEE, April 1978.
- [318] W. A. Kornfeld, "ETHER--A Parallel Problem Solving System," Proc. 6th IJCAI, pp. 490-492, William Kaufman, Los Altos, CA, 1979.
- [319] J. Darlington and M. Reeve, ALICE and the Parallel Evaluation of Logic Programs, Preliminary Draft, Dept. of Computing, Imperial College of Science and Technology, London, England, June 1983.
- [320] K. Smith, "New Computer Breed Uses Transputers for Parallel Processing," Electronics, pp. 67-68, McGraw-Hill, Feb. 24, 1983.
- [321] F. Hommes, "The Heap/Substitution Concept--An Implementation of Functional Operations on Data Structures for a Reduction Machine," Proc. 9th Annual Symp. on Computer Architecture, pp. 248-256, IEEE/ACM, April 1982.
- [322] W. E. Kluge, "Cooperating Reduction Machines," Trans. on Computers, vol. C-32, no. 11, pp. 1002-1012, IEEE, Nov. 1983.

- [323] R. M. Keller and F. C. H. Lin, "Simulated Performance of a Reduction-Based Multiprocessor," *Computer*, vol. 17, no. 7, pp. 70-82, IEEE, July 1984.
- [324] R. M. Keller, F. C. H. Lin, and J. Tanaka, "Rediflow Multiprocessing," *Proc. COMPCON Spring*, pp. 410-417, IEEE, 1984.
- [325] T. Clarke, P. Gladstone, C. Maclean, and A. Norman, "SKIM--The S, K, I Reduction Machine," *Conf. Record of Lisp Conf., Stanford Univ., Menlo Park, CA*, 1980.
- [326] P. Treleaven and G. Mole, "A Multi-Processor Reduction Machine for User-Defined Reduction Languages," *Proc. 7th Int'l Symp. Computer Architecture*, pp. 121-130, IEEE/ACM, 1980.
- [327] D. G. Bobrow, "If Prolog Is the Answer, What is the Question?," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 138-145, ICOT and North-Holland, 1984.
- [328] H. Ogawa, T. Kitahashi, and K. Tanaka, "The Theorem Prover Using a Parallel Processing System," *Proc. 6th IJCAI*, pp. 665-667, William Kaufman, Los Altos, CA, Aug. 1979.
- [329] S. I. Nakagawa and T. Sakai, "A Parallel Tree Search Method," *Proc. 6th IJCAI*, pp. 628-632, William Kaufman, Los Altos, CA, Aug. 1979.
- [330] C. Smith, "The Power of Parallelism for Automatic Programming Synthesis," *Proc. 22nd Annual Symp. on Fnd. of Computer Science*, ACM, 1981.
- [331] M. J. Wise, "EPILOG = Prolog + Data Flow: Arguments for Combining Prolog with a Data Driven Mechanism," *SIGPLAN Notices*, vol. 17, no. 12, pp. 80-86, ACM, Dec. 1982.
- [332] S. Umeyama and K. Tamura, "A Parallel Execution Model of Logic Programs," *Proc. 10th Annual Symp. on Computer Architecture*, pp. 349-355, IEEE/ACM, June 1983.
- [333] A. Ciepielewski and S. Haridi, "Execution of Bagof on the OR-Parallel Token Machine," *Proc. Int'l Conf. 5th Generation Computer Systems*, pp. 551-560, ICOT and North-Holland, 1984.
- [334] H. Yasuhara and K. Nitadori, "ORBIT: A Parallel Computing Model of Prolog," *New Generation Computing*, vol. 2, no. 3, pp. 277-288, OHMSHA Ltd. and Springer-Verlag, 1984.
- [335] D. DeGroot, "Restricted AND-Parallelism," *Proc. Int'l Conf. on 5th Generation Computers*, pp. 471-478, ICOT and North-Holland, Nov. 1984.
- [336] T. Khabaza, "Negation as Failure and Parallelism," *Proc. Int'l Symp. on Logic Programming*, pp. 70-75, IEEE, Feb. 1984.
- [337] G. Lindstrom and P. Panangaden, "Stream-Based Execution of Logic Programs," *Proc. Int'l Symp. on Logic Programming*, pp. 168-176, IEEE, Feb. 1984.
- [338] G.-J. Li and B. W. Wah, "MANIP-2: A Multicomputer Architecture for Evaluating Logic Programs," *Proc. Int'l Conf. on Parallel Processing*, pp. 123-130, IEEE, June 1985.
- [339] J. H. Chang, A. M. Despain, and D. DeGroot, "AND-Parallelism of Logic Programs Based on A Static Data Dependency Analysis," *Proc. COMPCON Spring*, pp. 218-225, IEEE, 1985.
- [340] J. S. Conery and D. F. Kibler, "Parallel Interpretation of Logic Programs," *Proc. Conf. on Functional Programming Languages and Computer Architecture*, pp. 163-170, ACM, 1981.
- [341] J. S. Conery and D. F. Kibler, "AND Parallelism and Nondeterminism in Logic Programs," *New Generation Computing*, vol. 3, no. 1, pp. 43-70, OHMSHA Ltd. and Springer-Verlag, 1985.
- [342] D. A. Carlson, "Parallel Processing of Tree-Like Computations," *Proc. 4th Int'l Conf. on Distributed Computing Systems*, pp. 192-198, IEEE, May 1984.
- [343] M. D. Rychener, "Control Requirements for the Design of Production System Architectures," *Proc. Symp. on AI and Programming Languages*, also *SIGART Newsletter*, pp. 37-44, ACM, Aug. 1977.
- [344] K. Oflazer, "Partitioning in Parallel Processing of Production Systems," *Proc. Int'l Conf. on Parallel Processing*, pp. 92-100, IEEE, 1984.
- [345] M. F. M. Tenorio and D. I. Moldovan, "Mapping Production Systems into Multiprocessors," *Proc. Int'l Conf. on Parallel Processing*, pp. 56-62, IEEE, 1985.
- [346] B. V. Funt, "Whisper: A Problem-Solving System Utilizing Diagrams," *Proc. 5th IJCAI*, pp. 459-464, William Kaufman, Los Altos, CA, Aug. 1977.
- [347] J. P. Adam, et al., *The IBM Paris Scientific Center Programming in Logic Interpreter: Overview, Report*, IBM France Scientific Center, Oct., 1984.
- [348] Y. Igawa, K. Shima, T. Sugawara, and S. Takagi, "Knowledge Representation and Inference Environment: KRINE--An Approach to Integration of Frame, Prolog and Graphics," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 643-651, ICOT and North-Holland, 1984.
- [349] M. Yokota, et al., "A Microprogrammed Interpreter for Personal Sequential Inference Machine," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 410-418, ICOT and North-Holland, 1984.
- [350] W. F. Clocksin, "Design and Simulation of a Sequential Prolog Machine," *New Generation Computing*, vol. 3, no. 2, pp. 101-120, OHMSHA Ltd. and Springer-Verlag, 1985.
- [351] M. S. Johnson, "Some Requirements for Architectural Support of Software Debugging," *Proc. SIGPLAN Symp. on Compiler Construction*, pp. 140-148, ACM, June 1982.
- [352] N. Tamura, K. Wada, H. Matsuda, Y. Kaneda, and S. Maekawa, "Sequential Prolog Machine PEK," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 542-550, ICOT and North-Holland, 1984.
- [353] D. B. Lanat and J. McDermott, "Less Than General Production System Architectures," *Proc. 5th IJCAI*, pp. 923-932, William Kaufman, Los Altos, CA, 1977.
- [354] C. J. Hogger, "Concurrent Logic Programming," in *Logic Programming*, ed. S.-A. Tarnlund and K. Clark, pp. 199-211, Academic Press, 1982.
- [355] M. H. van Emden and G. J. de Lucena-Filho, "Predicate Logic as a Language for Parallel Programming," in *Logic Programming*, ed. S.-A.

- Tarnlund and K. Clark, pp. 189-198, Academic Press, 1982.
- [356] E. Y. Shapiro, *Subset of Concurrent Prolog and its Interpreter*, Tech. Rep. TR-003, ICOT, Tokyo, Japan, 1983.
- [357] A. J. Kusalik, "Serialization of Process Reduction in Concurrent Prolog," *New Generation Computing*, vol. 2, no. 3, pp. 289-298, OHMSHA Ltd. and Springer-Verlag, 1984.
- [358] P. Borgwardt, "Parallel Prolog using Stack Segments on Shared-Memory Multiprocessors," *Proc. Int'l Symp. on Logic Programming*, pp. 2-11, IEEE, Feb. 1984.
- [359] K. Ueda and T. Chikayama, "Efficient Stream/Array Processing in Logic Programming Languages," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 317-326, ICOT and North-Holland, 1984.
- [360] E. Shapiro and A. Takeuchi, "Object Oriented Programming in Concurrent Prolog," *New Generation Computing*, vol. 1, no. 1, pp. 25-48, OHMSHA Ltd. and Springer-Verlag, 1983.
- [361] N. Suzuki, "Concurrent Prolog as an Efficient VLSI Design Language," *Computer*, vol. 18, no. 2, pp. 33-40, IEEE, Feb. 1985.
- [362] A. Takeuchi and K. Furukawa, "Bounded Buffer Communication in Concurrent Prolog," *New Generation Computing*, vol. 3, no. 2, pp. 145-155, OHMSHA Ltd. and Springer-Verlag, 1985.
- [363] K. Clark and S. Gregory, *PARLOG: Parallel Programming in Logic*, Research Rep. DOC 84/4, Imperial College, London, England, 1984.
- [364] L. M. Pereira and R. Nasr, "Delta-Prolog, A Distributed Logic Programming Language," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 283-291, ICOT and North-Holland, 1984.
- [365] L. M. Uhr, "Parallel-Serial Production Systems," *Proc. 6th IJCAI*, pp. 911-916, William Kaufman, Los Altos, CA, Aug. 1979.
- [366] E. Shapiro, "Systolic Programming: A Paradigm of Parallel Processing," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 458-470, ICOT and North-Holland, 1984.
- [367] C. Rieger, R. Trigg, and B. Bane, "ZMOB: A New Computing Engine for AI," *Proc. 7th IJCAI*, pp. 955-960, William Kaufman, Los Altos, CA, Aug. 1981.
- [368] R. Trigg, "Software on ZMOB: An Object-Oriented Approach," *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pp. 133-140, IEEE, Nov., 1981.
- [369] U. S. Chakravarthy, S. Kasif, M. Kohli, J. Minker, and D. Cao, "Logic Programming on ZMOB: A Highly Parallel Machine," *Proc. Int'l Conf. on Parallel Processing*, pp. 347-349, IEEE, Aug. 1982.
- [370] M. Weiser, S. Kogge, M. McElvany, R. Pierson, R. Post, and A. Thareja, "Status and Performance of the ZMOB Parallel Processing System," *Proc. COMPCON Spring*, pp. 71-73, IEEE, Feb. 1985.
- [371] S. Kasif, M. Kohli, and J. Minker, "PRISM: A Parallel Inference System for Problem Solving," *Proc. 8th IJCAI*, pp. 544-546, William Kaufman, Los Altos, CA, 1983.
- [372] A. M. Despain and Y. N. Patt, "Aquarius--A High Performance Computing System for Symbolic/Numeric Applications," *Proc. COMPCON Spring*, pp. 376-382, IEEE, Feb. 1985.
- [373] A. Koster, "Compiling Prolog Programs for Parallel Execution on a Cellular Machine," *Proc. ACM'84*, pp. 167-178, ACM, Oct. 1984.
- [374] L. Bic, "Execution of Logic Programs on a Dataflow Architecture," *Proc. 11th Annual Int'l Symp. on Computer Architecture*, pp. 290-296, IEEE/ACM, June 1984.
- [375] R. Hasegawa and M. Amamiya, "Parallel Execution of Logic Programs based on Dataflow Concept," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 507-516, ICOT and North-Holland, 1984.
- [376] K. B. Irani and Y. F. Shih, "Implementation of Very Large Prolog-Based Knowledge Bases on Data Flow Architectures," *Proc. 1st Conf. on AI Applications*, pp. 454-459, IEEE, Dec. 1984.
- [377] S. J. Stolfo and D. E. Shaw, *DADO: A Tree-Structured Machine Architecture for Production Systems*, Technical Report, Columbia Univ., New York, NY, March 1982.
- [378] A. Gupta, "Implementing OPS5 Production Systems on DADO," *Proc. Int'l Conf. on Parallel Processing*, pp. 83-91, IEEE, 1984.
- [379] S. J. Stolfo and D. P. Miranker, "DADO: A Parallel Processor for Expert Systems," *Proc. Int'l Conf. on Parallel Processing*, pp. 74-82, IEEE, Aug. 1984.
- [380] S. J. Stolfo, "Five Parallel Algorithms for Production System Execution on the DADO Machine," *Proc. Nat'l Conf. on AI*, pp. 300-307, AAAI, Aug. 1984.
- [381] W. Dilger and J. Muller, "An Associative Processor for Theorem Proving," *Proc. of the Symp. on AI*, pp. 489-497, IFAC, 1983.
- [382] I. Takeuchi, H. Okuno, and N. Ohsato, "TAO--A Harmonic Mean of Lisp, Prolog, and Smalltalk," *SIGPLAN Notices*, vol. 18, no. 7, pp. 65-74, ACM, July 1983.
- [383] A. L. Davis and S. V. Robison, "The FAIM-1 Symbolic Multiprocessing System," *Proc. COMPCON Spring*, pp. 370-375, IEEE, 1985.
- [384] N. V. Fiedler, *Associated Network*, Academic Press, 1979.
- [385] S. E. Fahlman, "Design Sketch for a Million-Element NETL Machine," *Proc. 1st Annual Nat'l Conf. on AI*, pp. 249-252, AAAI, Aug. 1980.
- [386] W. D. Hillis, "The Connection Machine: A Computer Architecture Based on Cellular Automata," *Physica*, pp. 213-228, North-Holland, 1984.
- [387] Thinking Machines Corporation, *The Connection Machine Supercomputer: A Natural Fit to Application Needs*, Tech. Rep., Thinking Machines Corporation, Waltham, MA, 1985.
- [388] D. I. Moldovan and Y. W. Tung, *SNAP: A VLSI Architecture for AI Processing*, Technical Report PPP 84-3, Univ. of Southern California, Los Angeles, CA, 1984.
- [389] D. I. Moldovan, "An Associative Array Architecture Intended for Semantic Network Processing," *Proc. ACM'84*, pp. 212-221, ACM, Oct. 1984.

- [390] G. E. Hinton, T. J. Sejnowski, and D. H. Askey, Boltzmann Machine: Constraint Satisfaction Network that Learns, Tech. Rep. Carnegie-Mellon Univ., Pittsburgh, PA, 1984.
- [391] D. H. Askey, G. E. Hinton, and T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, vol. 9, no. 1, pp. 147-169, 1985.
- [392] R. Davis, "Report on the Workshop on Distributed AI," SIGART Newsletter, no. 73, pp. 43-52, ACM, 1980.
- [393] R. Davis, "Report on the Second Workshop on Distributed AI," SIGART Newsletter, no. 80, pp. 13-83, ACM, 1982.
- [394] M. Fehling and L. Erman, "Report on the Third Annual Workshop on Distributed AI," SIGART Newsletter, no. 84, pp. 3-12, ACM, 1983.
- [395] R. G. Smith, "The Contract Net: A Formalism for the Control of Distributed Problem Solving," *Proc. 5th IJCAI*, p. 472, William Kaufman, Los Altos, CA, Aug. 1977.
- [396] R. G. Smith, "A Framework for Distributed Problem Solving," *Proc. 6th IJCAI*, pp. 836-841, William Kaufman, Los Altos, CA, Aug. 1979.
- [397] R. G. Smith and R. Davis, "Frameworks for Cooperation in Distributed Problem Solving," *Trans. on Systems, Man and Cybernetics*, vol. SMC-11, no. 1, pp. 61-70, IEEE, Jan. 1981.
- [398] J. Pavlin, "Predicting the Performance of Distributed Knowledge-Based Systems: A Modeling Approach," *Proc. Nat'l Conf. on AI*, pp. 314-319, AAAI, 1983.
- [399] D. D. Corkill and V. R. Lesser, "The Use of Meta-Level Control for Coordination in a Distributed Problem Solving Network," *Proc. 8th IJCAI*, pp. 748-756, William Kaufman, Los Altos, CA, Aug. 1983.
- [400] V. R. Lesser and D. D. Corkill, "The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks," *The AI Magazine*, pp. 15-33, AAAI, Fall 1983.
- [401] S. Cammarata, D. McArthur, and R. Steeb, "Strategies of Cooperation in Distributed Problem Solving," *Proc. of 8th IJCAI*, pp. 767-770, William Kaufman, Los Altos, CA, Aug. 1983.
- [402] A. S. Gevins, "Overview of the Human Brain as a Distributed Computing Network," *Proc. Int'l Conf. on Computer Design: VLSI in Computers*, pp. 13-16, IEEE, 1983.
- [403] W. Fritz and The Intelligent System, SIGART Newsletter, no. 90, pp. 34-38, ACM, Oct. 1984.
- [404] K. Kawanobe, "Current Status and Future Plans of the 5th Generation Computer System Project," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 3-36, ICOT and North-Holland, 1984.
- [405] P. C. Treleaven and I. G. Lima, "Japan's 5th-Generation Computer Systems," *Computer*, vol. 15, no. 8, pp. 79-88, IEEE, Aug. 1982.
- [406] T. Moto-oka, "Overview to the 5th Generation Computer System Project," *Proc. 10th Annual Int'l Symp. on Computer Architecture*, pp. 417-422, IEEE/ACM, June 1983.
- [407] L. Bic, "The 5th Generation Grail: A Survey of Related Research," *Proc. ACM'84*, pp. 293-297, ACM, Oct. 1984.
- [408] S. Uchida and T. Yokoi, "Sequential Inference Machine: SIM Progress Report," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 58-81, ICOT and North-Holland, 1984.
- [409] T. Yokoi, S. Uchida, and ICOT Third Laboratory, "Sequential Inference Machine: SIM--its Programming and Operating System," *Proc Int'l Conf. on 5th Generation Computer Systems*, pp. 70-81, ICOT and North-Holland, 1984.
- [410] K. Taki, M. Yokota, A. Yamamoto, H. Nishikawa, S. Uchida, H. Nakashima, and A. Mitsuishi, "Hardware Design and Implementation of the Personal Sequential Inference Machine (PSI)," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 398-409, ICOT and North-Holland, 1984.
- [411] M. Yokota, A. Yamamoto, K. Taki, H. Nishikawa, and S. Uchida, "The Design and Implementation of a Personal Sequential Inference Machine: PSI," *New Generation Computing*, vol. 1, no. 2, pp. 125-144, OHMSHA Ltd. and Springer-Verlag, 1983.
- [412] K. Murakami, T. Kakuta, R. Onai, and N. Ito, "Research on Parallel Machine Architecture for 5th-Generation Computer Systems," *Computer*, vol. 18, no. 6, pp. 76-92, IEEE, June 1985.
- [413] T. Moto-oka, H. Tanaka, H. Aida, K. Hirata, and T. Maruyama, "The Architecture of a Parallel Inference Engine (PIE)," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 479-488, ICOT and North-Holland, 1984.
- [414] A. Goto, H. Tanaka, and T. Moto-oka, "Highly Parallel Inference Engine PIE--Goal Rewriting Model and Machine Architecture," *New Generation Computing*, vol. 2, no. 1, pp. 37-58, OHMSHA Ltd. and Springer-Verlag, 1984.
- [415] S. Uchida, "Inference Machine: From Sequential to Parallel," *Proc. 10th Annual Int'l Symp. on Computer Architecture*, pp. 410-416, IEEE/ACM, June 1983.
- [416] N. Ito and H. Shimizu, "Dataflow Based Execution Mechanisms of Parallel and Concurrent Prolog," *New Generation Computing*, vol. 3, no. 1, pp. 15-41, OHMSHA Ltd. and Springer-Verlag, 1985.
- [417] K. Murakami, T. Kakuta, N. Miyazaki, S. Shibayama, and H. Yokota, "A Relational Data Base Machine: First Step to Knowledge Base Machine," *Proc. 10th Annual Int'l Symp. on Computer Architecture*, pp. 423-425, IEEE/ACM, June 1983.
- [418] K. Furukawa and T. Yokoi, "Basic Software System," *Proc. Int'l Conf. on 5th Generation Computer Systems*, pp. 37-57, ICOT and North-Holland, 1984.
- [419] C. D. McCrosky, J. J. Glasgow, and M. A. Jenkins, "Nial: A Candidate Language for 5th Generation Computer Systems," *Proc. ACM'84*, pp. 157-166, ACM, Oct. 1984.
- [420] D. Michie, "Inductive Rule Generation in the Context of the 5th Generation," *Proc. Int'l Machine Learning Workshop*, pp. 65-70, Univ. of Illinois, Urbana, IL, June 1983.
- [421] T. Maneul, "Cautiously Optimistic Tone Set For 5th Generation," *Electronics*, pp. 57-63, McGraw-Hill, Dec. 3, 1984.

ABSTRACTS

- [422] K. G. Wilson, "Science, Industry, and the New Japanese Challenge," Proc. IEEE, vol. 72, no. 1, pp. 6-18, IEEE, Jan. 1984.
- [423] "ESPRIT: Europe Challenges U.S. and Japanese Competitors," Future Generation Computer Systems, vol. 1, no. 1, pp. 61-69, North-Holland, 1984.
- [424] M. van Emden, "Towards a Western 5th-Generation Computer System Project," Proc. ACM'84, pp. 298-302, ACM, Oct. 1984.
- [425] Proc. ACM'84 Annual Conf.: The 5th Generation Challenge, ACM, 1984.
- [426] J. Darlington and M. Reeve, "ALICE: A Multi-processor Reduction Machine for the Parallel Evaluation of Applicative Languages," Proc. Conf. on Functional Programming Languages and Computer Architecture, pp. 65-74, ACM, 1981.
- [427] I. W. Moor, An Applicative Compiler for a Parallel Machine, Research Report DoC83/6, Imperial College, London, England, March 1983.
- [428] J. Darlington, A. J. Field, and H. Pull, The Unification of Functional and Logic Languages, Tech. Report, Imperial College, London, England, Feb. 1985.
- [429] M. Dawson, A LISP Compiler for ALICE, Tech. Report, Imperial College, London, England, 1985.
- [430] "Special Issue on Tomorrow's Computers," Spectrum, vol. 20, no. 11, pp. 51-58, 69, IEEE, Nov. 1983.

BINDINGS

Jim Hendler (formerly Brown University)
Computer Science Dept.
University of Maryland
College Park, Maryland 20742
301-454-2002

Adolfo Guzman (sabbatical year)
MCC
Parallel Processing Program
9430 Research Blvd.
Echelon Building #1, Suite 200
Austin, TX 78759
512-343-0860

The following abstracts may be ordered from:
Box 3CRL
Computing Research Laboratory
New Mexico State University
Las Cruces, NM 88003

Syntax, Preference and Right Attachment

Yorick Wilks, Xiuming Huang and Dan Fass
MCCS-85-5

The paper claims that the right attachment rules for phrases originally suggested by Frazier and Fodor are wrong, and that none of the subsequent patchings of the rules by syntactic methods have improved the situation. For each rule there are perfectly straightforward and indefinitely large classes of simple counter-examples. We then examine suggestions by Ford et al., Schubert and Hirst which are quasi-semantic in nature and which we consider ingenious but unsatisfactory. We offer a straightforward solution within the framework of preference semantics, and argue that the principal issue is not the type and nature of information required to get appropriate phrase attachments, but the issue of where to store the information and with what process to apply it. We present a prolog implementation of a best first algorithm covering the data and contrast it with closely related ones, all of which are based on the preferences of nouns and prepositions, as well as verbs.

Machine Translation in the Semantic Definite Clause Grammars Formalism

Xiuming Huang
MCCS-85-7

The paper describes the SDCG (Semantic Definite Clause Grammars), a formalism for Natural Language Processing (NLP), and the XTRA (English Chinese Sentence TRANslator) machine translation (MT) system based on it. The system translates general domain English sentences into grammatical Chinese sentences in a fully automatic manner. It is written in Prolog and implemented on the DEC-10, the GEC, and the SUN workstation, respectively.

SDCG is an augmentation of (Pereira et al 80)'s DCG (Definite Clause Grammars) which in turn is based on CFG (Context Free Grammars). Implemented in Prolog, the SDCG is highly suitable for NLP in general, and MT in particular.

A wide range of linguistic phenomena is covered by the XTRA system, including multiple word senses, coordinate constructions, and prepositional phrase attachment, among others.

Bad Metaphors: Chomsky and AI

Yorick Wilks
MCCS-85-8

The paper argues that the historical divisions between, on the one hand, cluster of approaches to language understanding by computer known as AI and, on the other, the Transformational Grammar system of Chomsky were caused not so much by matters of principle (as between a scientific linguistics and computational applications) or by methodology, as by Chomsky's attachment over the years to a succession of unfortunate metaphors which explain his position. As recent developments in linguistics have shown, once these are removed there are no issues of principle (though there may continue to be differences of