A SURVEY ON SPECIAL-PURPOSE COMPUTER ARCHITECTURES FOR AI

BENJAMIN W. WAH and GUO-JIE LI

1. INTRODUCTION

Many of today's computers are single-processor von Neumann machines designed for sequential and deterministic numerical computations [23, 25, 358, 289], and are not equipped for artificial intelligence (AI) applications that are mainly parallel, nondeterministic, symbolic manipulations [16, 49, 127, 387]. Consequently, efficient computer architectures for AI applications would be sufficiently different from traditional computers [50, 104, 105, 134, 153, 187, 274, 281, 326, 407]. These architectures have the following requirements.

1. *Symbolic processing.* In the microlevel, AI applications require symbolic processing operations such as 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 [60, 191, 285]. The most important ones are tagged mechanisms [87, 191] and hardware stacks [117].

Research supported by National Science Foundation grant DCI 85-19649.

2. *Parallel and distributed processing.* Most AI problems are complex [84, 305] 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 [182, 187, 198, 249, 315, 387]. 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* [404]. To extend the solvable problem space of such problems, the key is to find better models and more efficient heuristics.
3. *Nondeterministic processing.* Most AI algorithms are nondeterministic [78], that is, it is impossible to plan in advance the procedure to execute and to terminate with the available information. Therefore, dynamic allocation and load balancing of computational resources are essential in AI architectures [29, 187]. 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 wihout the scheduling information must be made. Morever, efficient garbage collection is important for AI architectures, owing to be dynamically allocated storage [29, 79, 248].
4. *Knowledge-base management.* Because a very large amount of information has to be stored and retrieved in AI applications, large knowledge bases are inevitable [20, 34, 319, 412]. 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.
5. *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 [41, 205], object-oriented architectures [204, 381], Lisp machines [87, 105], and Prolog-like machines, such as the Fifth Generation Computer System [153].

Currently, extensive research is underway to design efficient AI architectures. Many existing concepts in computer architecture — such as dataflow processing [107, 388], stack machines [117], tagging [191], pipelining [198], direct execution of high-level languages [68, 144, 424], data base machines [237], multiprocessing, and distributed processing — can be incorporated into future AI architectures. New concepts in computer architectures are also expected.

2. AI LANGUAGES AND PROGRAMMING

One goal of computer scientists working in the field of AI is to produce programs that imitate the intelligent behavior of human beings [30, 44, 67, 313, 419]. 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 [21, 224, 417]. To enhance programmers' productivity, a type of problem-oriented languages called declarative languages have been developed and widely applied in AI programming [121]. *Functional programming* [22, 183, 417] and *logic programming* [72, 226–228, 316] 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 some-what interchangeably [57, 95, 199, 390]. Examples of functional languages are pure Lisp [26, 170, 260, 261, 321], Backus' FP [21], Hope [58], Val [264], and Id [15]. 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 very large scale integration (VLSI) and parallelism.

In its modest form, a logic program refers to the procedural interpretation of Horn clauses or predicate logic [226, 227]. The computer language Prolog [73, 75, 80, 81, 116, 406] is based on logic programming. Generally speaking, logic programming is reasoning-oriented or deductive programming. In fact, some ideas of logic programming, like automatic backtracking, have been used in early AI languages QA3 [30], PLANNER, MICROPLANNER, and CONNIVER [44, 363]. 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 [284]. 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" [316].

New languages and programming systems are being developed to simplify AI programming radically. It is expected that *object-oriented programming* [312] will be as important in the 1980s as structured programming was in the 1970s. The language Smalltalk [2, 160] is an example of object-oriented programming. Some ideas of object-oriented programming have been used in existing languages and systems, such as Simula, B5000, Lisp-AI notion of frame, ADA, and CLU. Other new object-oriented programming systems have also been developed [204, 272, 365, 381, 397].

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 in which it is natural for the ideas to be applied. Except for the widely used language Prolog, Lisp and its dialects, Maclisp [277], Interlisp [338, 376], Qlisp [318], Common Lisp [351], Franz Lisp [413], etc., many other AI language have been designed and implemented. Examples include IPL [296, 297] PLANNER [184], CONNIVER [363], KRL [46], NETL [126], SAIL [311], POP-2 [93], FUZZY [131], 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 [316] and QUTE [325], which amalgamate Prolog and Lisp in natural ways, have been developed. On the other hand, to explore parallelism, parallel versions of Prolog and Lisp, such as Parlog [70], Concurrent Prolog [333, 335], and Concurrent Lisp [361, 367], have been proposed. Recent efforts have been aimed at automatic programming that will allow the program to be generated from a simple specification of the problem [31, 32, 242, 256, 343].

It has also been apparent to the AI community since the mid 1960s that inferences 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 the *knowledge-based system*, the field of applied AI [13, 240].

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 [56, 99, 157, 179–181, 294, 407]. 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, whereas in an expert system, the knowledge base is separated from the inference procedure, and new knowledge can be added to the system without programming.

Contemporary expert-system development techniques are shifting towards the use of software development tools that resemble a programming language, but include internal user-accessible data bases and other high-level strategies for using knowledge to solve a class of problems [88, 157, 181, 220]. 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 [267, 268], KS300, and S.1; (2) HEARSAY-III [125] and AGE [299]; and (3) OPS that incorporates the R1 (XCON) expert-system families [145]. Other expert-system tools include LOOPS [354], ROSIE [130], RLL [167] MRS, and KMS. Some of these tools aim to provide a mixture of representations and inference techniques. Knowledge-acquisition tools such as TEIRESIAS [97], EXPERT [411], KAS [120], and learning tools such as META-DENDRAL [55] and EURISKO [241] have also been developed.

3. MICRO-AND MACROLEVEL AI ARCHITECTURES

The VLSI technology has flourished in the past ten years [1, 54, 266, 329, 382], resulting in the development of advanced microprocessors [384], semiconductor memories [415], and systolic arrays [147, 148, 168, 231, 330].

The microlevel architectures consist of architectural designs that are fundamental to applications in AI. In the design of massively parallel AI machines [129], 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 [35, 43, 51, 197, 229, 309, 378, 379, 395, 418] and selection [401, 425], computing transitive closure [171, 231], string and pattern matching [8, 13, 14, 162, 174, 192, 285, 374], selection from secondary memories [119, 161], dynamic programming evaluations [39, 64, 171], proximity searches [427], and unification [118, 146, 380, 399, 400].

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 [353]. Additional concepts that are well suited for list processing are the tagged-memory [137, 191] and stack architectures [117].

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, data-base 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 data-base [18, 37, 61, 141, 238, 303, 348]. Most designs are based on binary-tree architectures; however, designs using radix trees and a small number of processors have been found to be preferable when keys are long and clustered [141].

A data-base 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 [270, 271], back-end storage systems [150], and data-base machines [19, 52, 156, 177, 196, 217, 236, 251, 336, 339]. Data-base machines developed earlier were mainly directed towards general-purpose relational data-base management systems. Examples include the DBC, DIRECT, RAP, CASSM, associative array processors, text retrieval systems [196, 236], and CAFS [19]. Nearly all current research on data-base machines to support knowledge data-bases assumes that the knowledge data-base is relational, hence research is directed towards solving the disk paradox [52] and enhancing previous relational data-base machines by extensive parallelism [287, 319, 340, 373]. Commercially available data-base and backend machines have also been applied in knowledge management [213, 214, 295].

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 [222, 223, 408].

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 [195], branch-and-bound [10, 24, 106, 149, 234, 235, 245, 247], and AND/OR-graph search [140, 142, 258] have been developed. Various parallel architectures to support divide-and-conquer algorithms [307, 341] and branch-and-bound algorithms [108, 122, 139, 175, 201, 202, 359, 402 404] 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 [27-29, 33, 45, 79, 110, 114, 136, 188, 230, 248, 259, 298, 349, 352]. 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 [261] in the early 1960s, but their ancestry went directly back to the Lambda Calculus developed by Church in the 1930s. The objective of writing a functional program is to define a set of (possibly recursive) equations for each function [91]. 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 10 to 30% 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 [105, 132, 143, 290].

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 [261]. 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 [36, 165, 219, 328], was superseded in 1978 by a second-generation Lisp machine, the CADR. This machine was a model for the first commercially available Lisp machines [22, 257, 291], 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.'s Lambda, supporting Zetalisp and LMLisp [87, 257, 291]. The Symbolics 3600, supporting Zetalisp, Flavors, and Fortran 77 [19, 278, 405, 409], the Xerox 1108 and 1132, supporting Interlisp-D and Smalltalk [47, 280, 337], and the Fijitsu FACOM Alpha Machine, a back-end Lisp processor, supporting Maclisp [9, 178]. 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 [350, 353, 364]. Other experimental computers to support Lisp and list-oriented processing have been reported [111, 164, 166, 169, 292, 310, 322–324, 370]. 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 multiprocessoring systems have been proposed to execute Lisp programs concurrently [173, 186, 265, 273, 361, 362, 414]. Dataflow processing is suitable for Lisp because these programs are generally data driven [422, 423]. Other multiprocessing and dataflow architectures to support list-processing have been proposed and developed [11, 12, 77, 113, 159, 385].

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 [60, 103, 170, 320, 321, 376], and conventional microcomputers have been used to implement Lisp compilers [216, 368]. Lisp is also available on various general- and special-purpose work stations, typically based on multiple 68000 processors [216, 368]. Lisp has been developed on Digital Equipment Corp.'s VAXstation 100, a MC68000-based personal graphics work station, and clusters of 11/782s running several dialects of Lisp and Common Lisp [360]. 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 station.

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 object's state, and forming class-superclass hierarchies [185, 381, 421]. 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 systems. Special hardware and multiprocessors have been proposed to directly support the processing of object-oriented languages [204, 308, 365, 397].

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 [398]. 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 cells and optimization for redundant calls and array operations may be implemented [6, 38, 63, 350, 390]. Tree-structured machines usually employ lazy evaluations, but suffer from the bottleneck at the root of the tree [94, 210, 251, 253, 300]. 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 [74, 90, 151, 194, 211, 212, 218, 221, 344, 383, 385].

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 [278] 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 [397] was to demonstrate that a Reduced Instruction Set Computer could provide high performance in an exploratory 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 [48, 50, 118, 146, 326, 407].

Substantial research has been carried out on parallel computational models of utilizing AND-parallelism, OR-parallelism, and stream parallelism in logical inference systems [41, 62, 66, 69, 82, 83, 102, 215, 246, 250, 293, 302, 342, 396, 420, 426], production systems [301, 317, 377], and others [154]. The basic problem on the exponential complexity of logic programs remains open at this time.

Sequential Prolog machines using software interpretation [7, 200], emulation [76, 429], and additional hardware support such as hardware unification and backtracking [207, 372, 380] have been proposed. Single-processor systems for production systems using additional data memories [239] and a RISC architecture [146] have been studied.

New logic programming languages suitable for parallel processing have been investigated [193]. In particular, the use of predicate logic [123], extensions of Prolog to become Concurrent Prolog [53, 233, 331, 334, 366, 375, 393], Parlog [71], and Delta-Prolog [306], and parallel production systems [394] have been developed. Concurrent Prolog has also been extended to include object-oriented programming [331] and has been applied as a VLSI design language [366]. 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 [332].

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 [53], ZMOB, a multiprocessor of Z80's connected by a ring network [65, 208, 314, 389, 410], AQUARIUS, a heterogeneous multiprocessor with a crossbar switch [109], and MAGO, a cellular machine implementing a Prolog compiler that translates a Prolog program into a formal functional program [225]. Techniques for analyzing Prolog programs such that they can be processed on a dataflow architecture have been derived [40, 176, 203, 205]. DADO is a multiprocessor system with a binary-tree interconnection network that implements parallel production systems [172, 355 357]. An associative processor has been proposed to carry out propositional and first-order predicate calculus [115].

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 [369]. A 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 interpretor with a speed of 4.5 KLIPS has been developed for Lisp Machine's Lambda [257]. Some of the prototype multiprocessors, such as ZMOB [65, 208, 314, 389, 410] and MAGO [225] 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 [96].

Besides being represented in logic, knowledge can also be represented in terms of semantic nets. Proposed and experimental architectures have been developed. NETL [126, 128, 138], and its generalization to THISTLE [129], 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 [86, 186]. Associative processors for processing semantic nets have also been proposed [275, 276].

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 [186]. *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 methods to constraint-satisfaction searches in a parallel network [190]. The most interesting aspect of this machine lies in its domain-independent learning algorithm [17].

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 [98, 100, 133]. With the proposed formalism of the Contract Net, contracts are used to express the control of problem solving in a distributed processor architecture [345–347]. Related work in this area include Petri-net modeling [304], distributed vehicle-monitoring testbed [84, 243], distributed air-traffic control system [59], and modeling the brain as a distributed system [152, 158].

6. FIFTH GENERATION COMPUTER SYSTEM

The Fifth Generation Computer System (FGCS) project was a project that 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
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 data-bases [42, 153, 209, 282, 284, 386].

A first effort in the FGCS project is to implement a sequential inference machine, or SIM [392, 428]. Its first implementation is a medium-performance machine known as a personal sequential inference, or PSI, machine [371, 430]. The current implementation is on the parallel inference machine, or PIM [163, 205, 206, 283, 287, 288, 391]. Another architectural development is on the knowledge-base machine, Delta [286-319, 340]. 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 [155, 263, 269]. Currently, all the projects are progressing well; however, the struggle is still far from over [255].

The Japanese FGCS project has stirred intensive responses from other countries [3-5, 89, 90, 92, 101, 124, 279, 344, 416]. 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 human-machine interfaces. Hardware development has focused on ALICE, a Parlog machine using dataflow architectures and implementing both Hope, Prolog, and Lisp [89, 90, 92, 101, 279, 344]. The European Comission has started the \$1.5 billion five-year European Strategic Program for Research in Information Technologies (Esprit) in 1984 [4]. 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 FCGS project was the establishment of the Microelectronics and Computer Technology Corp. in 1983 [3]. 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 [3].

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 AI, *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 AI 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 AI 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 problem 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 speed-up.

Although many AI architectures have been built or proposed, the Lisp machines are the only architectures 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 currnet supercomputers will have restricted use in some AI applications. For example, the large amount of branch and symbolic processing operations in AI programs reduces 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. Prediction of performance and linear scaling
8. Management of the large memory space.

REFERENCES

1. *Proc. 2nd Caltech Conf. on Very Large Scale Integration*. Rockville, MD: Computer Science Press, 1981.
2. "Special Issue on Smalltalk," *Byte*, Aug. 1981.
3. "Special Issue on Tomorrow's Computers," *Spectrum*, 20(11):51–58, Nov. 1983.
4. "ESPRIT: Europe Challenges U.S. and Japanese Competitors," *Future Generation Computer Systems*, 1(1):61–69, 1984.
5. *Proc. ACM '84 Annual Conference: The Fifth Generation Challenge*, ACM, 1984.
6. P. S. Abrahms, *An APL Machine*. Ph.D. thesis, Stanford University, Menlo Park, CA, Feb. 1970.
7. J. P. Adam, et al., *The IBM Paris Scientific Center Programming in Logic Interpreter: Overview*. IBM France Scientific Center, Oct. 1984.
8. 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*. IEEE/ACM, May 1980, pp. 218–227.
9. H. Akimoto, S. Shimizu, A. Shinagawa, A. Hattori, and H. Hayashi, "Evaluation of the Dedicated Hardware in FACOM Alpha," *Proc. COMPCON Spring*, IEEE, 1985, pp. 366–369.
10. S. G. Akl, D. T. Barnard, and R. J. Doran, "Design, Analysis and Implementation of a Parallel Tree Search Algorithm," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-4(2):192–203, 1982.
11. M. Amamiya, R. Hasegawa, O. Nakamura, and H. Mikami, "A List-Processing-Oriented Data Flow Machine Architecture," *Proc. Natl Computer Conf.* AFIPS Press, 1982, pp. 144–151.
12. M. Amamiya and R. Hasegawa, "Dataflow Computing and Eager and Lazy Evaluations," *New Generation Computing*, 2(2):105–129, 1984.
13. J. Aoe, Y. Yamamoto, and R. Shimada, "A Method for Improving String Pattern Matching Machines," *IEEE Trans. on Software Engineering*, SE-10(1):116–120, 1984.
14. A. Apostolico and A. Negro, "Systolic Algorithms for String Manipulations," *IEEE Trans. on Computers*, C-33(4):361–364, 1984.
15. Arvind, K. Gostelow, and W. Plouffe, *An Asynchronous Programming Language and Computing Machine*. Technical Report. 114a, University of California, Irvine, CA, Dec. 1978.

16. 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.
17. D. H. Askley, G. E. Hinton, and T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, 9(1):147–169, 1985.
18. M. J. Atallah and S. R. Kosaraju, "A Generalized Dictionary Machine for VLSI," *Trans. on Computers*, C-34 (2):151–155, 1985.
19. E. Babb, "Joined Normal Form: A Storage Encoding for Relational Databases," *Trans. on Database Systems*, 7(4):588–614, 1982.
20. E. Babb, "Functional Requirements for Very Large Knowledge Bases," *Proc. ACM'84 Annual Conf.*, pp. 55–56, ACM, Oct. 1984.
21. J. Backus, "Can Programming be Liberated from the von Neumann Style? A Functional Style and Algebra of Programs," *Comm. of the ACM*, 21(8):613–641, 1978.
22. J. Backus, "Function-Level Computing," *Spectrum*, 19(8):22–27, 1982.
23. J.-L. Baer, *Computer Systems Architecture*. Rockville, MD: Computer Science Press, 1980.
24. J. L. Baer, H. C. Du, and R. E. Ladner, "Binary Search in a Multiprocessing Environment," *IEEE Trans. on Computers*, C-32(7):667–677, 1983.
25. J. L. Baer, "Computer Architecture," *Computer*, 17(10):77–87, 1984.
26. H. G. Baker, Jr., "Shallow Binding in Lisp 1.5," *Comm. of the ACM*, 21(7):565–569, 1978.
27. H. C. Baker, Jr., and C. Hewitt, "The Incremental Garbage Collection of Processes," *Proc. Symp. on Artificial Intelligence and Programming Languages* (*also SIGART Newsletter*, pp. 55–59), ACM, Aug. 1977.
28. H. G. Baker, Jr., "List Processing in Real Time on a Serial Computer," *Comm. of the ACM*, 21(4):280–294, 1978.
29. H. G. Baker, Jr, "Optimizing Allocation and Garbage Collection of Spaces." In P. H. Winston and R. H. Brown (eds.), *Artificial Intelligence: An MIT Perspective*, vol. 1. Cambridge, MA: MIT Press, 1979, pp. 391–396.
30. A. Barr and E. A. Feigenbaum, *The Handbook on Artificial Intelligence*, 2. Los Altos, CA: William Kaufman, 1982.
31. D. Barstow, "A Perspective on Automatic Programming," *Proc. 8th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: Aug. 1983, William Kaufman, pp. 1170–1179.
32. D. R. Barstow, "An Experiment in Knowledge-Based Automatic Programming." In B. L. Webber and N. J. Nilsson (eds.), *Readings in Artificial Intelligence*. Palo Alto, CA: Tioga, 1981. pp. 289–312.
33. J. M. Barth, "Shifting Garbage Collection Overhead to Compile Time," *Comm. of the ACM*, 20(7):513–518, 1977.
34. M. Bartschi, "An Overview of Information Retrieval Subjects," *Computer*, 18(5):67–84, 1985.
35. G. Baudet and D. Stevenson, "Optimal Sorting Algorithms for Parallel Computers," *Trans. on Computers*, C-27(1):84–87, 1978.
36. A. Bawden, R. Greenblatt, J. Holloway, T. Knight, D. Moon, and D. Weinreb, "The Lisp Machine." In P. H. Winston and R. H. Brown (eds.), *Artificial Intelligence: An MIT Perspective*, vol. 1, Cambridge, MA: MIT Press, 1979, pp. 343–373.
37. J. L. Bentley and H. T. Kung, "A Tree Machine for Searching Problems," *Proc. Int'l Conf. on Parallel Processing*, IEEE, 1979, pp. 257–266.
38. K. J. Berkling, "Reduction Languages for Reduction Machines," *Proc. Int'l Symp. on Computer Architecture*, IEEE/ACM, 1975, pp. 133–140.
39. D. P. Bertsekas, "Distributed Dynamic Programming," *IEEE Trans. on Automatic Control*, AC-27 (3):610–616, 1982.
40. L. Bic, "Execution of Logic Programs on a Dataflow Architecture," *Proc. 11th Annual Int'l Symp. on Computer Architecture*, IEEE/ACM, June 1984, pp. 290–296.
41. L. Bic, "A Data-Driven Model for Parallel Interpretation of Logic Programs," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 517–523.
42. L. Bic, "The Fifth Generation Grail: A Survey of Related Research," *Proc. ACM '84 Annual Conf.*, ACM, Oct. 1984, pp. 293–297.
43. D. Bitton, D. J. DeWitt, D. K. Hsiao, and J. Menon, "A Taxonomy of Paralle Sorting," *Computing Surveys*, 16(3): 1984.
44. D. Bobrow and B. Paphael, "New Programming Languages for AI research," *Computing Surveys*, 6(3):153–174, 1974.

45. D. Bobrow and D. Clark, "Compact Encoding of List Structure," *ACM Trans. on Prog. Language and Systems*, 1(2): 266–286, 1979.
46. D. G. Bobrow and T. Winograd, "An Overview of KRL—A Knowledge Representation Language," *Cognitive Science* 1(1):3–46, 1976.
47. D. G. Bobrow, *The LOOPS Manual*, Technical Report KB-VLSI-81-13, Xerox PARC, Palo Alto, CA, 1982.
48. D. G. Bobrow, "If Prolog Is the Answer. What is the Question?," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 138–145.
49. H. Boley, "A Preliminary Survey of Artificial Intelligence Machines," *SIGART Newsletter*, 72: 21–28, 1980.
50. H. Boley, "AI Languages and AI Machines: An Overview," *Proc. German Workshop on Artificial Intelligence*, Springer-Fachberichte, 1981.
51. M. A. Bonuccelli, E. Lodi, and L. Pagli, "External Sorting in VLSI," *IEEE Trans. on Computers*, C-33(10):931–934, 1984.
52. H. Boral and D. DeWitt, "Database Machine: An Idea Whose Time has Passed?," *Database Machines*. New York: Springer-Verlag, 1983, pp. 166–167.
53. P. Borgwardt, "Parallel Prolog using Stack Segments on Shared-Memory Multiprocessors," *Proc. Int'l Symp. on Logic Programming*, IEEE, Feb. 1984, pp. 2–11.
54. R. Bryant (ed.), *Proc. 3rd Caltech Conf. on Very Large Scale Integration*. Rockviller, MD: Computer Science Press, 1983.
55. B. G. Buchanan and E. A. Feigenbaum, "Dendral and Meta-Dendral: Their Applications Dimension," *Artificial Intelligence*, 11(1–2):5–24, 1978.
56. B. G. Buchanan, "New Research on Expert Systems." In Hayes, D. Michie, and Y.-H. Pao, (eds.), *Machine Intelligence 10*. Chichester, Eng.: Ellis Horwood Ltd., 1982, pp. 269–299.
57. R. Burstall, "Programming with Modules as Typed Functional Programming," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 103–112.
58. R. M. Burstall, D. B. MacQueen, and D. T. Sannella, "HOPE: An Experimental Applicative Language," *Conf. Record of Lisp Conf.*, Stanford University, Menlo Park, CA, 1980, pp. 136–143.
59. S. Cammarata, D. McArthur, and R. Steeb, "Strategies of Cooperation in Distributed Problem Solving," *Proc. 8th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman Aug. 1983, pp. 767–770.
60. J. Campbell and J. Fitch, "Symbolic Computing With and Without Lisp," *Conf. Record of Lisp Conf.*, Stanford University, Menlo Park, CA, 1980.
61. 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, University of California, Berkeley, CA, April 1982.
62. D. A. Carlson, "Parallel Processing of Tree-Like Computations," *Proc. 4th Int'l Conf. on Distributed Computing Systems*, IEEE, May 1984, pp. 192–198.
63. 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, 239–247.
64. J. Casti, M. Richardson, and R. Larson, "Dynamic Programming and Parallel Computers," *J. of Optimization Theory and Applications*, 12(4):423–438, 1973.
65. 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*, IEEE, Aug. 1982, pp. 347–349.
66. J. H. Chang, A. M. Despain, and D. DeGroot, "AND-Parallelism of Logic Programs Based on A Static Data Dependency Analysis," *Proc. COMPCON Spring*, IEEE, 1985, pp. 218–225.
67. E. Charniak, C. Riesbeck, and D. McDermott, *Artificial Intelligence Programming*. New York: Lawrence Erlbaum Press, 1980.
68. Y. Chu, "Direct-Execution Computer Architecture." In B. Gilchrist (ed.), *Information Processing 77*. North-Holland, 1977, pp. 18–23.
69. A. Ciepielewski and S. Haridi, "Execution of Bagof on the OR-Parallel Token Machine," *Proc. Int'l Conf. Fifth Generation Computer Systems*. ICOT and North-Holland, 1984, pp. 551–560.
70. K. Clark and S. Gregory, "Note on System Programming in PARLOG," *Proc. Int'l Conf. Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 299–306.
71. K. Clark and S. Gregory, *PARLOG: Parallel Programming in Logic*. Research Rep. DOC 84/4, Imperial College, London, England, 1984.
72. K. L. Clark and S.-A. Turnlund (eds), *Logic Programming*. New York: Academic Press, 1982.

73. K. L. Clark and F. G. McCabe, "Prolog: A Language for Implementing Expert Systems." In J. Hayes, D. Michie, and Y. H. Pao, *Machine Intelligence 10*. Chichester, England: Ellis Horwood Ltd., 1982, pp. 455–471.
74. T. Clarke, P. Gladstone, C. Maclean, and A. Norman, "SKIM —The S, K, I Reduction Machine," *Conf. Record of Lisp Conf.*, Stanford University, Menlo Park, CA, 1980.
75. W. F. Clocksin and C. S. Mellish, *Programming in Prolog*. New York: Springer-Verlag, 1981.
76. W. F. Clocksin, "Design and Simulation of a Sequential Prolog Machine," *New Generation Computing*, 3(2): 101–120, 1985.
77. G. Coghill and K. Hanna, "PLEIADES: A Multimicroprocessor Interactive Knowledge Base," *Microprocessors and Microsystems*, 3(2):77–82, 1979.
78. J. Cohen, "Non-Deterministic Algorithms," *Computing Surveys*, 11(2):79–94, 1979.
79. J. Cohen, "Garbage Collection of Linked Data Structures," *Computing Surveys*, 13(3):341–367, 1981.
80. A. Colmerauer, H. Kanoui, and M. Van Caneghem, "Last Steps Towards an Ultimate Prolog," *Proc. 7th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, Aug. 1981, pp. 947–948.
81. A. Colmerauer, "Prolog in 10 Figures," *Proc. 8th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1983, pp. 488–499.
82. J. S. Conery and D. F. Kibler, "Parallel Interpretation of Logic Programs," *Proc. Conf. on Functional Programming Languages and Computer Architecture*, ACM, 1981, pp. 163–170.
83. J. S. Conery and D. F. Kibler, "AND Parallelism and Nondeterminism in Logic Programs," *New Generation Computing*, 3(1):43–70, 1985.
84. S. A. Cook, "An Overview of Computational Complexity," *Comm. of the ACM*, 26(6):401–408, 1983.
85. D. D. Corkill and V. R. Lesser, "The Use of Meta-Level Control for Coordination in a Distributed Problem Solving Network," *Proc. 8th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, Aug. 1983, pp. 748–756.
86. Thinking Machines Corporation, *The Connection Machine Supercomputer: A Natural Fit to Application Needs*. Tech. Rep., Thinking Machines Corporation, Waltham, MA, 1985.
87. M. Creeger, "Lisp Machines Come Out of the Lab.," *Computer Design*, Penn Well, Nov. 1983, pp. 132–137.
88. A. S. Cromarty, "What Are Current Expert System Tools Missing?" *Proc. COMPCON Spring*, IEEE, 1985. pp. 411–418.
89. 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*, ACM, 1981, pp. 65–74.
90. 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.
91. J. Darlington, "Functional Programming". In F. B. Chambers, D. A. Duce, and G. P. Jones (eds.), *Distributed Computing*. London: Academic Press, 1984, Chap. 5.
92. J. Darlington, A. J. Field, and H. Pull, *The Unification of Functional and Logic Languages*, Tech. Rep., Imperial College, London, England, Feb. 1985.
93. D. Davies, et al., *POPLER 1.5 Reference Manual*. University of Edinburgh, Edinburgh, Scotland, 1973.
94. A. L. Davis, "A Data Flow Evaluation System Based on the Concept of Recursive Locality," *Proc. National Computer Conf.*, AFIPS Press, 1979, pp. 1079–1086.
95. A. L. Davis and R. M. Keller, "Data Flow Program Graphs," *Computer*, 15(2):26–41, 1982.
96. A. L. Davis and S. V. Robinson, "The FAIM-1 Symbolic Multiprocessing System," *Proc. COMPCON Spring*, IEEE, 1985, pp. 370–375.
97. R. Davis and B. Buchanan, "Meta-level Knowledge: Overview and Applications," *Proc. 5th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1977, pp. 920–928.
98. R. Davis, "Report on the Workshop on Distributed Artificial Intelligence," *SIGART Newsletter*, 73: 43–52. 1980.
99. R. Davis, "Expert Systems: Where Are We? And Where Do We Go From Here?' *AI Magazine*, AAAI, Spring 1982, pp. 3–22.
100. R. Davis, "Report on the Second Workshop on Distributed Artifical Intelligence," *SIGART Newsletter*. 80: 13 83, 1982.
101. M. Dawson, *A LISP Compiler for ALICE*, Tech. Rep. Imperial College, London, Eng., 1985.

102. D. DeGroot, "Restricted AND-Parallelism," *Proc. Int'l Conf. on Fifth Generation Computers*, ICOT and North-Holland, Nov. 1984, pp. 471–478.
103. M. Deering, J. Faletti, and R. Wilensky, "PEARL—A Package for Efficient Access to Representations in Lisp," *Proc. 7th Int'l Joint Conf. on Artifical Intelligence*. Los Altos, CA:William Kaufman, Aug. 1981, pp. 930–932.
104. M. F. Deering, "Hardware and Software Architectures for Efficient AI," *Proc. National Conf. on Artifical Intelligence*, AAAI, Aug. 1984, pp. 73–78.
105. M. F. Deering, "Architectures for AI," *Byte*, April 1985, pp. 193–206.
106. E. Dekel and S. Sahni, "Binary Trees and Parallel Scheduling Algorithms," *IEEE Trans. on Computers*, C-32(3):307–315, 1983.
107. J. B. Dennis, "Data Flow Supercomputers," *Computer*, 13(11):48–56, 1980.
108. B. C. Desai, "A Parallel Microprocessing System," *Proc. Int'l Conf. on Parallel Processing*, IEEE, Aug. 1979, p. 136.
109. A. M. Despain and Y. N. Patt, "Aquarius—A High Performance Computing System for Symbolic/Numeric Applications," *Proc. COMPCON Spring*, IEEE, Feb. 1985, pp. 376–382.
110. L. Deutsch and D. Bobrow, "An Efficient Incremental, Automatic Garbage Collector," *Comm. of the ACM*, 19(9):522–526, 1976.
111. P. Deutsch, "Experience with a Microprogrammed Interlisp Systems," *Proc. MICRO*, vol. 11, ACM/IEEE, Nov. 1978.
112. P. Deutsch, "ByteLisp and its Alto Implementation," *Conf. Record of Lisp Conf.*, Stanford University, Menlo Park, CA, 1980.
113. H. Diel, "Concurrent Data Access Architecture," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984. pp. 373–388.
114. 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*, 21(11):967–975, 1978.
115. W. Dilger and J. Muller, "An Associative Processor for Theorem Proving," *Proc. Symp. on Artifical Intelligence*, IFAC, 1983, pp. 489–497.
116. B. Domolki and P. Szeredi, "Prolog in Practice," In R. E. A. Mason (ed.), *IFIP Information Processing*. New York: Elsevier, 1983, pp. 627–636.
117. R. Doran, "Architecture of Stack Machine." In Y. Chu (ed.), *High-Level Language Computer Architecture*, New York: Academic Press, 1975.
118. R. J. Douglass, "A Qualitative Assessment of Parallelism in Expert Systems," *Software* 2(2): 70–81, 1985.
119. H. C. Du, "Concurrent Disk Accessing for Partial Match Retrieval," *Proc. Int'l Conf. on Parallel Processing*, IEEE, 1982, pp. 211–218.
120. 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*, Edinburgh, Scotland: Edinburgh University Press, 1979.
121. S. Eisenbach and C. Sadler, "Declarative Languages: an Overview," *Byte*, Aug. 1985, pp. 181–197.
122. O. I. El-Dessouki and W. H. Huen, "Distributed Enumeration on Between Computers," *IEEE Trans. on Computers*, C-29(9):818–825, 1980.
123. M. H. van Emden and G. J. de Lucena-Filho, "Predicate Logic as a Language for Parallel Programming." In S.-A. Tarnlund and K. Clark (eds.), *Logic Programming*, New York: Academic Press, 1982.
124. M. van Emden, "Towards a Western Fifth-Generation Computer System Project." *Proc. ACM'84 Annual Conf.* ACM, Oct. 1984, pp. 298–302.
125. L. Erman, P. London, and S. Fickas, "The Design and Example, Use of HEARSAY-III." *Proc. 7th Int'l Joint Conf. on Artifical Intelligence*. Los Altos, CA: William Kaufman, 1981, pp. 409–415.
126. S. Fahlman, *NETL: A System for Representing and Using Real-World Knowledge*. Series on Artifical Intelligence. Cambridge, MA: MIT Press, 1979.
127. S. Fahlman, "Computing Facilities for AI: A Survey of Present and Near-Future Options," *AI Magazine*, AAAI, Winter 1980–1981, pp. 16–23.
128. S. E. Fahlman, "Design sketch for a Million-Element NETL Machine," *Proc. 1st Annual Nat'l. Conf. on Artifical Intelligence*, AAAI, Aug. 1980, pp. 249–252.
129. S. E. Fahlman and G. E. Hinton, "Massively Parallel Architectures for AI: NETL, THISTLE, and BOLTZMANN Machines," *Proc. Nat'l. Conf. on Artifical Intellgence*, AAAI, 1983, pp. 109–113.

130. 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.
131. R. A. Le Faivre, *FUZZY Reference Manual*. Computer Science Dept., Rutgers University, New Brunswick, NJ, 1977.
132. R. Fateman, "Is a Lisp Machine Different from Fortran Machine?," *SIGSAM Bulletin*, 12(4): ACM, 1978.
133. M. Fehling and L. Erman, "Report on the Third Annual Workshop on Distributed Artificial Intelligence," *SIGART Newsletter*, (84):3–12, 1983.
134. E. A. Feigenbaum, F. Hayes-Roth, D. Waltz, R. Reddy, and V. Zue, "The Building Blocks," *Spectrum*, IEEE, Nov. 1983, pp. 77–87.
135. E. A. Feigenbaum, "Knowledge Engineering: The Applied Side." In J. E. Hayes and D. Michie (eds.), *Intelligent Systems: The Unprecedented Opportunity*. Chichester, England: Ellis Horwood Ltd., 1983, pp. 37–55.
136. R. Fenichel and J. Yochelson, "A Lisp Garbage-Collector for Virtual Memory Computer Systems," *Comm. of the ACM* 12(11):611–612, 1979.
137. E. A. Feustel, "On the Advantages of Tagged Architecture," *IEEE Trans. on Computers*, C-22(7): 644–656, 1973.
138. N. V. Findler, *Associated Network*. New York: Academic Press, 1979.
139. R. Finkel and U. Manber, "DIB—A Distributed Implementation of Backtracking," *Proc. 5th Int'l Conf. on Distributed Computing Systems*, IEEE, May 1985, pp. 446–452.
140. R. A. Finkel and J. P. Fishburn, "Parallelism in Alpha-Beta Search," *Artifical Intelligence*, 19(1): 89–106, 1982.
141. A. L. Fisher, "Dictionary Machines with a Small Number of Processors," *Proc. 11th Annual Int'l Symp. on Computer Architecture*, IEEE/ACM, June 1984, pp. 151–156.
142. D. H. Fishman and J. Minker, "π-Representation: A Clause Representation for Parallel Search," *Artificial Intelligence* 6(2):193–127, 1975.
143. J. Fitch, "Do We Really Want a Lisp Machine?" *SEAS/SMC Annual Meeting*, ACM, Jan. 1980.
144. M. J. Flynn, "Directions and Issues in Architecture and Language," *Computer* 13(10):5–22, 1980.
145. C. Forgy and J. McDermott, "OPS—A Domain-Independent Production Systems Language," *Proc. 5th Int'l Joint Conf. on Artifical Intelligence*. Los Altos, CA: William Kaufman, 1977, pp. 933–939.
146. C. Forgy, A. Gupta, A. Newell, and R. Wedig, "Initial Assessment of Architectures for Production Systems," *Proc. National Conf. on Artifical Intelligence*, AAAI, Aug. 1984, pp. 116–120.
147. 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 Accoustics, Speech, and Signal Proc.*, IEEE, 1985, pp. 8.9.1–8.9.4.
148. M. J. Foster and H. T. Kung, "The Design of Special-Purpose VLSI Chips," *Computer*, 13(1): 26–40, 1980.
149. M. A. Franklin and N. L. Soong, "One-Dimensional Optimization on Multiprocessor Systems," *IEEE Trans. on Computers*, C-30(1):61–66, 1981.
150. H. A. Freeman (ed.) "Special Issue on Backend Storage Networks," *Computer*, 13(2): Feb. 1980.
151. D. P. Friedman and D. S. Wise, "Aspects of Applicative Programming for Parallel Processing," *IEEE Trans. on Computers*, C-27(4):289–296, 1978.
152. W. Fritz and The Intelligent System, *SIGART Newsletter*, 90:34–38, 1984.
153. K. Fuchi, "The Direction the FGCS Project Will Take," *New Generation Computing*, 1(1):3–9, 1983.
154. B. V. Funt, "Whisper: A Problem-Solving System Utilizing Diagrams," *Proc. 5th Int'l Joint Conf. on Artifical Intelligence*. Los Atlos, CA: William Kaufman, 1977, pp. 459–464.
155. K. Furukawa and T. Yokoi, "Basic Software System," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 37–57.
156. 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*, IEEE/ACM, June 1984, pp. 134–141.
157. W. B. Gevarter, *An Overview of Expert Systems*. Tech. Rep. NBSIR 82-2505, National Bureau of Standards, Washington, DC, 1982.
158. A. S. Gevins, "Overview of the Human Brain as a Distributed Computing Network," *Proc. Int'l Conf. on Computer Design: VLSI in Computers*, IEEE, 1983, pp. 13–16.
159. W. K. Giloi and R. Gueth, "Concepts and Realization of a High-Performance Data Type

Architecture," *Int'l J. of Computer and Information Sciences*, 11(1):25-54, New York: Plenum Press, 1982.
160. A. J. Goldberg and D. Robson, *Smalltalk-80: The Language and Its implementation*. Reading, MA: Addision-Wesley, 1983.
161. R. Gonzalez-Rubio, J. Rohmer, and D. Terral, "The Schuss Filter: A Process for Non-Numerical Data Processing," *Proc. 11th Annual Int'l Symp. on Computer Architecture*. IEEE/ACM, June 1984, pp. 64–73.
162. K. Goser, C. Foelster, and U. Rueckert, "Intelligent Memories in VLSI," *Information Sciences*, 34(1):61-82, 1984.
163. A. Goto, H. Tanaka, and T. Moto-oka, "Highly Parallel Inference Engine PIE—Goal Rewriting Model and Machine Architecture," *New Generation Computing* 2(1):37–58, 1984.
164. E. Goto, T. Ida, K. Hiraki, M. Suzuki, and N. Inada, "FLATS, A. Machine for Numerical, Symbolic and Associative Computing," *Proc. 6th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1979, pp. 1058–1066.
165. 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*, ACM, March 1980, pp. 137-138.
166. N. Greenfeld and A. Jericho, "A Professional's Personal Computer System," *Proc. 8th Int'l Symp. on Comp. Architecture*, IEEE/ACM, 1981, pp. 217–227.
167. R. Greiner and D. Lenat, "A Representation Language," *Proc. First National Conference on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1980, pp. 165–169.
168. J. Grinberg, G. R. Nudd, and R. D. Etchells, "A Cellular VLSI Architecture," *Computer* 17(1): 68-81, 1984.
169. M. Griss and M. Swanson, "MBALM/1700: A Microprogrammed Lisp Machine for the Burroughs B1726," *Proc. MICRO-10*, ACM/IEEE, 1977.
170. M. L. Griss and E. Benson, "Current Status of a Portable Lisp Compiler," *Proc. SIGPLAN Symp. on Compiler Construction*, ACM, June 1982, pp. 276-283.
171. L. J. Guibas, H. T. Kung, and C. D. Thompson, "Direct VLSI Implementation of Combinatorial Algorithms," *Proc. Caltech Conf. on VLSI*, Caltech, Pasadena, CA, 1979, pp. 509–525.
172. A. Gupta, "Implementing OPS5 Production Systems on DADO," *Proc. Int'l Conf. on Parallel Processing*. IEEE, 1984, pp. 83-91.
173. A. Guzman, "A Heterarchical Multi-Microprocessor Lisp Machine," *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Database Management*. IEEE, Nov. 1981, pp. 309-317.
174. P. A. V. Hall and G. R. Dowling, "Approximate String Matching," *Computing Surveys*, 12(4): 381–402, 1980.
175. J. A. Harris and D. R. Smith, "Simulation Experiments of a Tree Organized Multicomputer," *Proc. 6th Annual Symp. on Computer Architecture*, IEEE/ACM, April 1979, pp. 83–89.
176. R. Hasegawa and M. Amamiya, "Parallel Execution of Logic Programs based on Dataflow Concept," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 507–516.
177. P. B. Hawthorn and D. J. DeWitt, "Performance Analysis of Alternative Database Machine Architectures," *IEEE Trans. on Software Engineering*, SE-8(1):61-75, 1982.
178. 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*, IEEE/ACM, June 1983, pp. 342–348.
179. F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, *Building Expert Systems*, Reading, MA: Addison-Wesley, 1983.
180. F. Hayes-Roth, "The Knowledge-Based Expert System: Z Tutorial," *Computer*, 17(9):11–28, 1984.
181. F. Hayes-Roth, "Knowledge-Based Expert Systems," *Computer*, 17(10):263-273, 1984.
182. L. S. Haynes, R. L. Lau, D. P. Siewiorek, and D. W. Mizell, "A Survey of Highly Parallel Computing," *Computer*, 15(1):9-24, 1982.
183. P. Henderson, *Function Programming, Application and Inplementation*, New York: Prentice-Hall, 1980.
184. C. Hewitt, *Description and Theoretical Analysis (Using Schemas) of PLANNER: A Language for Proving Theorems and Manipulating Models in Robots*, PhD. thesis, Artificial Intelligence Laboratory, MIT, Cambridge, MA, 1971.

185. C. Hewitt, "Viewing Control Structure as Patterns of Passing Messages," *Artificial Intelligence*, 8(3):323–364, 1977.

186. C. Hewitt, "The Apiary Network Architecture for Knowledgeable Systems," *Conf. Record of Lisp Conf.*, Stanford University, Menlo Park, CA, 1980, pp. 107–117.

187. C. Hewitt and H. Lieberman, "Design Issues in Parallel Architectures for Artificial Intelligence," *Proc. COMPCON Spring*, IEEE, Feb. 1984, pp. 418–423.

188. Y. Hibino, "A Practical Parallel Garbage Collection Algorithm and Its Implementations," *Proc. 7th Annual Symp. on Computer Architecture*, IEEE/ACM, May 1980, pp. 113–120.

189. W. D. Hills, "The Connection Machine: A Computer Architecture Based on Cellular Automata," *Physica*, North-Holland, 1984, pp. 213–228.

190. G. E. Hinton, T. J. Sejnowski, and D. H. Askley, *Boltzmann Machine: Constraint Satisfaction Network that Learns*. Tech. Rep. Carnegie-Mellon University, Pittsburgh, PA, 1984.

191. A. Hirsch, "Tagged Architecture Supports Symbolic Processing", *Computer Design*, PennWell, June 1984.

192. C. Hoffmann and M. O'Donnell, "Pattern Matching in Trees," *J. of the ACM*, 21(1):68–95, 1982.

193. C. J. Hogger, "Concurrent Logic Programming." In *Logic Programming*, S.-A. Tarnlund and K. Clark (eds.), New York: Academic Press, 1982, pp. 199–211.

194. 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*, IEEE/ACM, April 1982, pp. 248–256.

195. E. Horowitz and A. Zorat, "Divide-and-Conquer for Parallel Processing," *Trans. on Computers*, C-32(6):582–585, 1983.

196. D. K. Hsiaso (ed.), "Special Issue on Database Machines," *Computer*, 12(3):1979.

197. C. C. Hsiao and L. Snyder, "Omni-Sort: A Versatile Data Processing Operation for VLSI," *Proc. Int'l Conf. on Parallel Processing*, IEEE, 1983, pp. 222–225.

198. K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*, New York: McGraw-Hill, 1984.

199. T. Ida and J. Tanaka, "Functional Programming with Streams—Part II," *New Generation Computing*, 2(3):261–275, 1984.

200. 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 Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 643–651.

201. M. Imai and T. Fukumura, "A Parallelized Branch-and-Bound Algorithm Implementation and Efficiency," *Systems, Computers, Controls*, 10(3):62–70, 1979.

202. M. Imai, Y. Tateizumi, Y. Yoshida, and T. Fukumura, "A Multicomputer System Based on the Binary-Tree Structure: DON(2)," *TGEC*, EC83-23(1):19–30, IECE of Japan, 1983.

203. K. B. Irani and Y. F. Shih, "Implementation of Very Large Prolog-Based Knowledge Bases on Data Flow Architectures," *Proc. 1st Conf. on Artifical Intelligence Applications*, IEEE, Dec. 1984, pp. 454–459.

204. Y. Ishikawa and M. Tokoro, "The Design of an Object-Oriented Architecture," *Proc. 11th Int'l Symp. on Computer Architecture*, IEEE/ACM, 1984, pp. 178–187.

205. N. Ito, H. Shimizu, M. Kishi, E. Kuno, and K. Rokusawa, "Data-Flow Based Execution Mechanisms of Parallel and Concurrent Prolog," *New Generation Computing*, 3:15–41, 1985.

206. N. Ito and H. Shimizu, "Dataflow Based Execution Mechanisms of Parallel and Concurrent Prolog," *New Generation Computing* 3(1):15-41, 1985.

207. M. S. Johnson, "Some Requirements for Architectural Support of Software Debugging," *Proc. SIGPLAIN Symp. on Compiler Construction*, ACM, June 1982, pp. 140–148.

208. S. Kasif, M. Kohli, and J. Minker, "PRISM: A Parallel Inference System for Problem Solving," *Proc. 8th Int'l Joint Conf. on Artificial Intelligence*, Los Altos, CA: William Kaufman, 1983, pp. 544–546.

209. K. Kawanobe, "Current Status and Future Plans of the Fifth Generation Computer System Project," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 3–36.

210. R. M. Keller, G. Lindstrom, and S. Patil, "A Loosely Coupled Applicative Multiprocessing System," *Proc. National Computer Conf.*, AFIPS Press, 1979, pp. 613–622.

211. R. M. Keller and F. C. H. Lin, "Simulated Performance of a Reduction-Based Multiprocessor," *Computer*, 17(7):70–82, 1984.

212. R. M. Keller, F. C. H. Lin, and J. Tanaka, "Rediflow Multiprocessing," *Proc. COMPCON Spring*, IEEE, 1984, pp. 410–417.
213. C. Kellogg, "Knowledge Management: A Practical Amalgam of Knowledge and Data Base Technology," *Proc. National Conf. on Artificial Intelligence*, AAAI, 1982, pp. 306–309.
214. C. Kellogg, "Intelligent Assistants for Knowledge and Information Resources Management," *Proc. 8th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1983, pp. 170–172.
215. T. Khabaza, "Negation as Failure and Parallelism," *Proc. Int'l Symp. on Logic Programming*, IEEE, Feb. 1984, pp. 70–75.
216. T. King, "Expert Systems with 68000 and Lisp," *Microprocessors and Microsystems*, Vol. 8, no. 7, IPC Business Press, England, Sept. 1984, pp. 374–376.
217. M. Kitsuregawa, H. Tanaka, and T. Moto-oka, "Application of Hash to Data Base Machine and its Architecture," *New Generation Computing*, 1(1):63–74, 1983.
218. W. E. Kluge, "Cooperating Reduction Machines," *Trans. on Computers*, C-32(11):1002–1012, 1983.
219. T. Knight, "The CONS Microprocessor." AI Working Paper 80, MIT, Cambridge, MA, Nov. 1974.
220. V. P. Kobler, "Overview of Tool for Knowledge Base Construction," *Proc. Data Engineering Conf.*, IEEE, 1984, pp. 282–285.
221. W. A. Kornfeld, "ETHER—A Parallel Problem Solving System," *Proc. 6th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1979, pp. 490–492.
222. W. A. Kornfeld, "The Use of Parallelism to Implement a Heuristic Search," *Proc. 7th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, Aug. 1981, pp. 575–580.
223. W. A. Kornfeld, "Combinatorially Implosive Algorithms," *Comm. of the ACM*, 25(10):734–738, 1982.
224. B. D. Kornman. "Pattern Matching and Pattern-Directed Invocation in Systems Programming Languages," *J. of Systems and Software*, 3:95-102, 1983.
225. A. Koster, "Compiling Prolog Programs for Parallel Execution on a Cellular Machine," *Proc. ACM'84 Annual Conf.*, ACM, Oct. 1984, pp. 167–178.
226. R. Kowalski, "Predicate Logic as a Programming Language," *IFIP Information Processing*, North-Holland, 1974, pp. 569–574.
227. R. Kowalski, *Logic for Problem Solving*, North-Holland, 1979.
228. R. Kowalski, "Logic Programming." In R. E. A. Mason (ed.), *IEIP Information Processing*. New York: Elsevier, 1983, pp. 133–145.
229. C. P. Kruskal, "Searching, Merging, and Sorting in Paralle Computation," *Trans. on Computers*, C-32(10) 942–946, 1983.
230. H. Kung and S. Song. *An Efficient Parallel Garbage Collection Systems and Its Correctness Proof*. Tech. Rep. Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, Sept. 1977.
231. H. T. Kung, "Let's Design Algorithms for VLSI Systems," *Proc. Caltech Conf. on VLSI*, Caltech, Pasadena, CA, Jan. 1979, pp. 65–90.
232. T. Kurokawa, "Lisp Activities in Japan," *Proc. 6th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1979, pp. 502–504.
233. A. J. Kusalik, "Serialization of Process Reduction in Concurrnet Prolog," *New Generation Computing*, 2(3):289–298, 1984.
234. T. H. Lai and S. Sahni, "Anomalies in Parallel Branch-and-Bound Algorithms," *Comm. of the ACM*, 27(6):594–602, 1984.
235. T. H. Lai and A. Sprague, "Performance of Parallel Branch-and-Bound Algorithms," *Trans. on Computers*, C-34(10):962–964, 1985.
236. G. G. Langdon, Jr. (ed), "Special Issue on Database Machines," *Trans. on Computers*, C-28(6):1979.
237. G. G. Langdon, Jr., "Database Machines: An Introduction," *Trans. on Computers*, C-28(6):381–384, 1979.
238. C. E. Leiserson, "Systolic Priority Queues," *Proc. Caltech Conf. on VLSI*, Caltech, Jan. 1979.
239. D. B. Lenat and J. McDermott, "Less Than General Production System Architectures," *Proc. 5th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1977, pp. 923–932.
240. D. B. Lenat, "Computer Software for Intelligent Systems," *Scientific American*, 251(3):204–213, 1984.

241. D. B. Lenat and J. S. Brown, "Why AM and EURISKO Appear to Work?" *Artificial Intelligence*, 23(3) 269–294, 1984.
242. E. J. Lerner, "Automating Programming," *Spectrum*, 19(8):28–33, 1982.
243. V. R. Lesser and D. D. Corkill, "The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks," *AI Magazine*, Fall 1983, pp. 15–33.
244. S. P. Levitan and J. G. Bonar, "Three Microcomputer Lisps," *Byte*, Sept. 1981, pp. 388–412.
245. G.-J. Li and B. W. Wah, "Computational Efficiency of Parallel Approximate Branch-and-Bound Algorithms," *Proc. Int'l Conf. on Parallel Processing*, IEEE, 1984, pp. 473–480.
246. G.-J. Li and B. W. Wah, "MANIP-2: A Multicomputer Architecture for Evaluating Logic Programs," *Proc. Int'l Conf. on Parallel Processing*. IEEE, June 1985, pp. 123–130.
247. G.-J. Li and B. W. Wah, "Coping with Anomalies in Parallel Branch-and-Bound Algorithms," *IEEE Trans. on Computers*, C-35(6):568–573, 1986.
248. H. Lieberman and C. Hewitt, "A Real-Time Garbage Collector Based on the Lifetimes of Objects," *Comm. of the ACM*, 26(6):419–429, 1983.
249. N. R. Lincoln, "Technology and Design Tradeoffs in the Creation of a Modern Supercomputer," *Trans. on Computers*, C-31(5):349–362, 1982.
250. G. Lindstrom and P. Panangaden, "Stream-Based Execution of Logic Programs." *Proc. Int'l Symp. on Logic Programming*, IEEE, Feb. 1984, pp. 168–176.
251. G. Mago, "A Network of Microprocessors to Execute Reduction Languages, Part I," *Int'l J. of Computer and Information Sciences*, 8(5):349–385, 1979.
252. G. Mago, "A Network of Microprocessors to Execute Reduction Languages, Part II," *Int'l J. of Computer and Information Sciences*, 8(6):435–471, 1979.
253. G. Mago, "Making Parallel Computation Simple: The FFP Machine," *Proc. COMPCON Spring*, IEEE, 1985, pp. 424–428.
254. F. J. Malabarba, "Review of Available Database Machine Technology," *Proc. Trends and Applications*, IEEE, 1984, pp. 14–17.
255. T. Maneul, "Cautiously Optimistic Tone Set For 5th Generation," *Electronics*, Dec. 3, 1984, pp. 57–63.
256. Z. Manna and R. Waldinger, "A Deductive Approach to Program Synthesis," *Proc. 6th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1979, pp. 542–551.
257. T. Manuel, "Lisp and Prolog Machines are Proliferating," *Electronics*, Nov. 1983, pp. 132–137.
258. T. A. Marsland and M. Campbell, "Parallel Search of Strongly Ordered Game Trees," *Computing Surveys*, 14(4):533–551, 1982.
259. J. J. Martin, "An Efficient Garbage Compaction Algorithm," *Comm. of the ACM*, 25(8):571–581, 1982.
260. J. McCarthy, P. Abrahams, D. Edwards, T. Hart, and M. Levin, *Lisp 1.5 Programmer's Manual*. Cambridge, MA: MIT Press, 1962.
261. J. McCarthy, "History of Lisp," *SIGPLAN Notices*, 13(8):217–223, 1978.
262. 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 K. Preston, Jr., and L. Uhr (eds.), *Multicomputers and Image Processing Algorithms and Programs*. New York: Academic Press, 1982, pp. 431–451.
263. C. D. McCrosky, J. J. Glasgow, and M. A. Jenkins, "Nial: A Canadidate Language for Fifth Generation Computer Systems," *Proc. ACM'84 Annual Conf.*, ACM. Oct. 1984, pp. 157–166.
264. J. R. McGraw, "Data Flow Computing: Software Development," *Trans. on Computers*, C-29(12):1095–1103, 1980.
265. D. McKay and S. Shapiro, "MULTI – A Lisp Based Multiprocessing System," *Conf. Record of Lisp Conf.*, Stanford University, Menlo Park, CA, 1980.
266. C. Mead and L. Conway, *Introduction to VLSI Systems*. Reading, MA: Addison-Wesley, 1980.
267. 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.
268. Van Melle, A. C. Scott, J. S. Bennett, and M. Peairs, The EMYCIN Manual, Tech. Rep. HPP-81-16, Computer Science Dept., Stanford University. Menlo Park, CA, 1981.
269. D. Michie, "Inductive Rule Generation in the Context of the Fifth Generation," *Proc. Int'l Machine Learning Workshop*, University of Illinois, Urbana, IL, June 1983, pp. 65–70.
270. S. W. Miller (ed.), "Special Issue on Mass Storage Systems Evolution of Data Center Architectures," *Computer*, 15(7): July 1982.
271. S. W. Miller (ed), "Special Issue on Mass Storage Systems," *Computer*, 18(7): July 1985.

272. 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 Fifth Generation Computer Systems*, ICOT and North-Holland, 1982, pp. 606–612.
273. M. Model, "Multiprocessing via Intercommunicating Lisp Systems," *Conf. Recrod of Lisp Conf.*, Stanford University, Menlo Park, CA, 1980.
274. D. I. Moldovan, *Survey of Computer Architectures for Artificial Intelligence*. Tech. Rep. PPP-84-6, University of Southern California, Los Angeles, CA, July 1984.
275. D. I. Moldovan and Y. W. Tung, *SNAP: A VLSI Architecture for Artificial Intelligence Processing*. Tech. Rep. PPP 84-3, University of Southern California, Los Angeles, CA, 1984.
276. D. I. Moldovan, "An Associative Array Architecture Intended for Semantic Network Processing," *Proc. ACM'84 Annual Conf.*, Oct. 1984, pp. 212–221.
277. D. Moon, *Maclisp Reference Manual*. Cambridge, MA: MIT Press, 1974.
278. D. A. Moon, "Architecture of the Symbolics 3600," *Proc. 12th Annual Int'l Symp. on Computer Architecture*, IEEE/ACM, June 1985, pp. 76–83.
279. I. W. Moor, *An Applicative Compiler for a Parallel Machine*. Research Rep. DoC83/6, Imperial College, London, Eng., March 1983.
280. J. Moore, *The Interlisp Virtual Machine Specification*. Tech. Rep. CSL 76-5, Xerox PARC, Palo Alto, CA, Sept. 1976.
281. T. Moto-oka, et al. "Challenge for Knowledge Information Processing Systems," *Proc. Int'l. Conf. on Fifth Generation Systems*. ICOT, and North-Holland, 1981, pp. 3–89.
282. T. Moto-oka, "Overview to the Fifth Generation Computer System Project," *Proc. 10th Annual Int'l Symp. on Computer Architecture*. IEEE/ACM, June 1983, pp. 417–422.
283. 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 Fifth Generation Computer Systems*. ICOT and North-Holland, 1984, pp. 479–488.
284. T. Moto-oka and H. S. Stone, "Fifth-Generation Computer Systems: A Japanese Project," *Computer*, 17(3):6–13, 1984.
285. A. Mukhopadhyay, "Hardware Algorithms for Nonnumeric Computation," *Trans. on Computers*, C-28(6):384–394., 1979.
286. 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*. IEEE/ACM, June 1983, pp. 423–425.
287. K. Murakami, T. Kakuta, and R. Onai, "Architectures and Hardware Systems: Parallel Inference Machine and Knowledge Base Machine," *Proc. Int'l Conf. on Fifth Generation Computer Systems*. ICOT and North-Holland, 1984, pp. 18–36.
288. K. Murakami, T. Kakuta, R. Onai, and N. Ito, "Research on Parallel Machine Architecture for Fifth-Generation Computer Systems," *Computer*, 18(6):76–92, 1985.
289. G. Myer, *Advances in Computer Architecture*. New York: Wiley, 1978.
290. E. Myers, "Machine that Lisp," *Datamation*, 27(9):105–108, 1981.
291. W. Myers, "Lisp Machine Displayed at AI Conf., " *Computer*, 15(11):79–82, 1982.
292. M. Nagao, J. I. Tsujii, K. Nakajima, K. Mitamura, and H. Ito, "Lisp Machine NK3 and Measurement of Its Performance," *Proc. 6th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1979, pp. 625–627.
293. S. I. Nakagawa and T. Sakai, "A Parallel Tree Search Method," *Proc. 6th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1979, pp. 628–632.
294. D. S. Nau, "Expert Computer System," *Computer*, 16(2):63–85, 1983.
295. P. M. Neches, "Hardware Support for Advanced Data Management Systems," *Computer* 17(11):29–40, 1984.
296. A. Newell, J. C. Shaw, and H. A. Simon, "Programming the Logic Theory Machine," *Prof. 1957 Western Joint Computer Conference*. IRE, 1957, pp. 230–240.
297. A. Newell, J. C. Shaw, and H. A. Simon, "Empirical Explorations with the Logic Theory Machine." In E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*. 1963, pp. 109–133.
298. J. A. Newman and M. C. Woodward, "Alternative Approaches to Multiprocessor Garbage Collection," *Proc. Int'l Conf. on Parallel Processing*. IEEE, 1982, pp. 205–210.
299. H. P. Nii and N. Aiello, "AGE (Attempt to Generalize):A Knowledge-Based Program for Building Knowledge-Based Programs," *Proc. 6th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1979, pp. 645–655.

300. J. T. O'Donnel, *A Systolic Associative Lisp Computer Architecture with Incremental Parallel Storage Management*. Ph.D. dissertation, University of Iowa, Iowa City, IA, 1981.
301. K. Oflazer, "Partitioning in Parallel Processing of Production Systems," *Proc. Int'l Conf. on Parallel Processing*. IEEE, 1984, pp. 92–100.
302. H. Ogawa, T. Kitahashi, and K. Tanaka, "The Theorem Prover Using a Parallel Processing System," *Proc. 6th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1979, pp. 665–667.
303. T. A. Ottmann, A. L. Rosenberg, and L. J. Stockmeyer, "A Dictionary Machine (for VLSI)," *IEEE Trans. on Computers*. C-31(9):892–897, 1982.
304. J. Pavlin, "Predicting the Performance of Distributed Knowledge-Based Systems: A Modeling Approach," *Proc. National Conf. on Artificial Intelligence*. AAAI, 1983, pp. 314–319.
305. J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley, 1984.
306. L. M. Pereira and R. Nasr, "Delta-Prolog, A Distributed Logic Programming Language," *Proc. Int'l Conf. on Fifth Generation Computer Systems*. ICOT and North-Holland, 1984, pp. 283–291.
307. F. J. Peters, "Tree Machine and Divide-and-Conquer Algorithms," *Lecture Notes CS 111 (CONPAR81)*. New York Springer-Verlag, 1981, pp. 25–35.
308. A. Plotkin and D. Tabak, "A Tree Structured Architecture for Semantic Gap Reduction," *Computer Architecture News*, 11(4):30–44, 1983.
309. F. P. Preparata, "New Parallel-Sorting Schemes," *Trans. on Computers*. C-27(7):669–673, July 1978.
310. E. von Puttkamer, "A Microprogrammed Lisp Machine," *Microprocessing and Microprogramming*. 11(1):9–14, 1983.
311. J. F. Reiser, ed., SAIL, Tech. Rep. STAN-CS-76-574, Computer Science Dept. Stanford University, Menlo Park, CA, 1976.
312. T. Rentsch, "Object Oriented Programming," *SIGPLAN Notices*, 17(9):51–57, 1982.
313. E. Rich, "The Gradual Expansion of Artificial Intelligence," *Computer*. 17(5):4–12; 1984.
314. C. Rieger, R. Trigg, and B. Bane, "ZMOB: A New Computing Engine for AI," *Proc. 7th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1981, pp. 955–960.
315. J. P. Riganati and P. B. Schneck, "Supercomputing," *Computer*, 17(10):97–113, 1984.
316. J. A. Robinson, "Logic Programming—Past, Present and Future," *New Generation Computing*, 1(2):107–124, 1983.
317. M. D. Rychener, "Control Requirements for the Design of Production System Architectures," *Proc. Symp. on Artificial Intelligence and Programming Languages (also SIGART Newsletter)*, ACM, Aug. 1977, pp. 37–44.
318. E. D. Sacerdoti, R. E. Fikes, R. Reboh, D. Sagalowicz, R. J. Waldinger, and B. M. Wilber, "Qlisp—A Language for the Interactive Development of Complex Systems," *Proc. National Computer Conf.*, AFIPS Press, 1976, pp. 139–146.
319. H. Sakai, K. Iwata, S. Kamiya, M. Abe, A. Tanaka, S. Shibayama, and K. Murakami, "Design and Implementation of Relational Database Engine," *Proc. Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 419–426.
320. H. Samet, "Code Optimization Considerations in List Processing Systems," *Trans. on Software Engineering*, SE-8(2):107–113, 1982.
321. E. Sandewall, "Programming in an Interactive Environment: the Lisp Experience," *Computing Surveys*. 10(1):35–71, 1978.
322. J. Sansonnet, D. Botella, and J. Perez, "Function Distribution in a List-Directed Architecture," *Microprocessing and Microprogramming*, 9(3):143–153, 1982.
323. J. P. Sansonnet, M. Castan, and C. Percebois, "M3L: A List-Directed Architecture," *Proc. 7th Annual Symp. on Computer Architecture*. IEEE/ACM, May 1980, pp. 105–112.
324. 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*. ACM, March 1982, pp. 132–139.
325. M. Sato and T. Sakurai, "QUTE: A Prolog/Lisp Type Language for Logic Programming," *Proc. 8th Int'l Joint Conf. on Artifical Intelligence*. Los Altos, CA: William Kaufman, 1983, pp. 507–513.
326. D. Schaefer and J. Fischer, "Beyond the Supercomputer," *Spectrum*, 19(3):32–37, 1982.
327. H. Schmeck and H. Schroder, "Dictionary Machines for Different Models of VLSI," *Trans. on Computers*, C-34(5):472–475, 1985.

328. S. R. Schoichet, "The Lisp Machine," *Mini-Micro Systems*, June 1978, pp. 68–74.
329. C. L. Seitz (ed.), *Proc. Caltech Conf. on Very Large Scale Integration*, Caltech, Pasadena, CA, Jan. 1979.
330. C. L. Seitz, "Concurrent VLSI Architectures," *Trans. on Computers*, C-33(12):1247–1265, 1984.
331. E. Shapiro and A. Takeuchi, "Object Oriented Programming in Concurrent Prolog," *New Generation Computing*, 1(1):25–48, 1983.
332. E. Shapiro, "Systolic Programming: A Paradigm of Parallel Processing," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 458–470.
333. E. Y. Shapiro, "Object Oriented Programming in Concurrent Prolog," *New Generation Computing*, 1(1):25–48, 1983.
334. E. Y. Shapiro, *Subset of Concurrent Prolog and its Interpreter*. Tech. Rep. TR-003, ICOT, Tokyo, Japan, 1983.
335. E. Y. Shapiro, *A Subset of Concurrent Prolog and its Interpreter*. Tech. Rep. TR-003, ICOT, Tokyo, Japan, 1984.
336. D. E. Shaw, *Knowledge-Based Retrieval on a Relational Database Machine*, Ph.D. dissertation, Stanford University, Menlo Park, CA; also Tech. Rep., Columbia University, New York, NY, Aug. 1980.
337. B. Sheil, "Family of Personal Lisp Machines Speeds AI Program Development," *Electronics*, Nov. 1983, pp. 153–156.
338. B. Sheil, "Power Tools for Programmers," *Datamation*, Technical Publishing, Feb. 1983, pp. 131–144.
339. J. Shemer and P. Neches, "The Genesis of a Database Computer," *Computer*, 17(11):42–56, 1984.
340. 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*, 2(2):131–155, 1984.
341. M. R. Sleep and F. W. Burton, "Towards a Zero Assignment Parallel Processor," *Proc. 2nd Int'l Conf. on Distributed Computing Systems*, IEEE, April 1981, pp. 80–85.
342. C. Smith, "The Power of Parallelism for Automatic Programming Synthesis," *Proc. 22nd Annual Symp. on Found. of Computer Science*, ACM, 1981.
343. D. R. Smith, "A Design for an Automatic Programming System," *Proc. 7th Int'l Joint Conf. on Artificial Intelligence*, Los Altos, CA: William Kaufman, Aug. 1981, pp. 1027–1029.
344. K. Smith, "New Computer Breed Uses Transputers for Parallel Processing," *Electronics*, Feb. 24, 1983, pp. 67–68.
345. R. G. Smith, "The Contract Net: A Formalism for the Control of Distributed Problem Solving," *Proc. 5th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, Aug. 1977, p. 472.
346. R. G. Smith, "A Framework for Distributed Problem Solving," *Proc. 6th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, Aug. 1979, pp. 836–841.
347. R. G. Smith and R. Davis, "Frameworks for Cooperation in Distributed Problem Solving," *Trans. on Systems, Man and Cybernetics*, SMC-11(1):61–70, 1981.
348. A. K. Somani and V. K. Agarwal, "An Efficient VLSI Dictionary Machine," *Proc. 11th Annual Int'l Symp. on Computer Architecture*, IEEE/ACM, June 1984, pp. 142–150.
349. D. Spector, "Minimal Overhead Garbage Collection of Complex List Structure," *SIGPLAN Notices*, 17(3):80–82, 1982.
350. G. Steel and G. Sussman, Design of Lisp-Based Processor, or SCHEME: A Dielectric Lisp or Finite Memories Considered harmful, or LAMBDA: The Ultimate Opcode. AI Memo 514, MIT, Cambridge, MA, March 1979.
351. G. L. Steele, Jr, "An Overview of Common Lisp," *Conf. Record of the 1982 Symp. on Lisp and Function Programming*, ACM, 1982, pp. 98–107.
352. G. Steele, "Multiprocessing Compactifying Garbage Collection," *Comm. of the ACM*, 18,(9):495–508, 1975.
353. G. L. Steele Jr. and G. J. Sussman, "Design of a Lisp-Based Microprocessor," *Comm. of the ACM*, 23(11):628–645, 1980.
354. M. Stefik, D. Bobrow, S. Mittal, and L. Conway, "Knowledge Programming in LOOPS: Report on an Experimental Course," *AI Magazine*, Fall 1983, pp. 20–30.
355. S. J. Stolfo and D. E. Shaw, *DADO: A Tree-Structured Machine Architecture for Production Systems*. Tech. Rept, Columbia University, New York, NY, March 1982.

356. S. J. Stolfo and D. P. Miranker, "DADO: A Parallel Processor for Expert Systems," *Proc. Int'l Conf. on Parallel Processing*, IEEE, Aug. 1984, pp. 74–82.
357. S. J. Stolfo, "Five Parallel Algorithms for Production System Execution on the DADO Machine," *Proc. National Conf. on Artificial Intelligence*, AAAI, Aug. 1984, pp. 300–307.
358. H. S. Stone, *Introduction to Computer Architecture*, Second Ed. Science Research Associates, 1980.
359. Q. F. Stout, "Sorting, Merging, Selecting and Filtering on Tree and Pyramid Machines," *Proc. Int'l Conf. on Parallel Processing*, IEEE, Aug. 1983, pp. 214–221.
360. W. D. Strecker, "Clustering VAX Superminicomputers into Large Multiprocessor Systems," *Electronics*, Oct. 20, 1983, pp. 143–146.
361. S. Sugimoto, K. Tabata, K. Agusa, and Y. Ohno, "Concurrent Lisp on a Multi-Micro-Processor System," *Proc. 7th Int'l Conf. on Artificial Intelligence*, Los Altos, CA: William Kaufman, 1981, pp. 949–954.
362. S. Sugimoto, K. Agusa, K. Tabata, and Y. Ohno, "A Multi-Microprocessor Systems for Concurrent Lisp," *Proc. Int'l Conf. on Parallel Processing*, IEEE, 1983, pp. 135–143.
363. G. J. Sussman and D. V. McDermott, "From PLANNER to CONNIVER—A Genetic Approach," *Fall Joint Computer Conf.*, vol. 41, AFIPS Press, 1972, pp. 129–137.
364. G. J. Sussman, J. Holloway, G. L. Steel, Jr., and A. Bell, "Scheme-79—Lisp on a Chip," *Computer*, 14(7):10–21, 1981.
365. N. Suzuki, K. Kubota, and T. Aoki, "SWORD32: A Bytecode Emulating Microprocessor for Object-Oriented Languages," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, pp. ICOT and North-Holland, 1984, pp. 389–397.
366. N. Suzuki, "Concurrent Prolog as an Efficient VLSI Design Language," *Computer*, 18(2):33–40, 1985.
367. 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.
368. S. Taff, "The Design of an M6800 Lisp Interpreter," *Byte*, Aug. 1979, pp. 132–152.
369. I. Takeuchi, H. Okuno, and N. Ohsato, "TAO—A Harmonic Mean of Lisp, Prolog, and Smalltalk," *SIGPLAN Notices*, 18(7): 65–74, 1983.
370. K. Taki, Y. Kaneda, and S. Maekawa, "The Experimental Lisp Machine," *Proc. 6th Int'l Joint Conf. on Artificial Intelligence*, Los Altos, CA: William Kaufman, 1979, pp. 865–867.
371. 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 Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 398–409.
372. N. Tamura, K. Wada, H. Matsuda, Y. Kaneda, and S. Maekawa, "Sequential Prolog Machine PEK," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 542–550.
373. Y. Tanaka, "MPDC-Massive Parallel Architecture for Very Large Databases," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 113–137.
374. 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*, IEEE, Oct. 1983, pp. 253–256.
375. A. Taueuchi and K. Furukawa, "Bounded Buffer Communication in Concurrent Prolog," *New Generation Computing*, 3(2):145–155, 1985.
376. W. Teitelman and L. Masinter, "The Interlisp Programming Environment," *Computer*, 14(4):25–33, 1981.
377. M. F. M. Tenorio and D. I. Moldovan, "Mapping Production Systems into Multiprocessors," *Proc. Int'l Conf. on Parallel Processing*, IEEE, 1985. pp. 56–62.
378. C. D. Thompson and H. T. Kung, "Sorting on a Mesh-Connected Parallel Computer," *Comm. of the ACM*, 20(4):263–271, 1977.
379. C. D. Thompson, "The VLSI Complexity of Sorting," *Trans. on Computers*, C-32(12):1171–1184, 1983.
380. E. Tick and D. H. D. Warren, "Towards a Pipelined Prolog Processor," *New Generation Computing*, 2(4):323–345, 1984.
381. M. Tokoro and Y. Ishikawa, "An Object-Oriented Approach to Knowledge Systems," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 623–632.

382. B. P. Treleaven and C. Philip (eds.), *VLSI Architectures*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
383. P. Treleaven and G. Mole, "A Multi-Processor Reduction Machine for User-Defined Reduction Languages," *Proc. 7th Int'l Symp. Computer Architecture*, IEEE/ACM, 1980, pp. 121–130.
384. P. C. Treleaven, "VLSI Processor Architectures," *Computer*, 15(6):33–45, 1982.
385. P. C. Treleaven and R. P. Hopkins, "A Recursive Computer Architecture for VLSI," *Proc. 9th Annual Symp. on Computer Architecture*, IEEE/ACM, April 1982, pp. 229-238.
386. P. C. Treleaven and I. G. Lima, "Japan's Fifth-Generation Computer Systems," *Computer*, 15(8):79-88, 1982.
387. P. C. Treleaven, "The New Generation of Computer Architecture," *Proc. 10th Annual Int'l Symp. on Computer Architecture*, IEEE/ACM, June 1983, pp. 402-409.
388. P. C. Treleaven and I. G. Lima, "Future Computers: Logic, Data Flow, . . . , Control Flow?" *Computer*, 17(3):47-55, 1984.
389. R. Trigg, "Software on ZMOB: An Object-Oriented Approach," *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, IEEE, Nov., 1981, pp. 133-140.
390. D. A. Turner, "A New Implementation Technique for Applicative Languages," *Software—Practice and Experience*, 9(1):31–49, 1979.
391. S. Uchida, "Inference Machine: From Sequential to Parallel," *Proc. 10th Annual Int'l Symp. on Computer Architecture*, IEEE/ACM, June 1983, pp. 410–416.
392. S. Uchida and T. Yokoi, "Sequential Inference Machine- SIM Progress Report," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 58-81.
393. K. Ueda and T. Chikayama, "Efficient Stream/Array Processing in Logic Programming Languages," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 317–326.
394. L. M. Uhr, "Parallel-Serial Production Systems," *Proc. 6th Int'l Joint Conf. on Artificial Intelligence*. Los Altos, CA: William Kaufman, 1979, pp. 911-916.
395. J. D. Ullman, "Some Thoughts About Supercomputer Organization," *Proc. COMPCON Spring*, IEEE, Feb. 1984, pp. 424–432.
396. S. Umeyama and K. Tamura, "A Parallel Execution Model of Logic Programs," *Proc. 10th Annual Symp. on Computer Architecture*, IEEE/ACM, June 1983, pp. 349–355.
397. 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*, IEEE/ACM, 1984, pp. 188–197.
398. S. R. Vegdahl, "A Survey of Proposed Architectures for the Execution of Functional Languages," *Trans. on Computers*, C-33(12):1050–1071, 1984.
399. J. S. Vitter and R. A. Simons, "Parallel Algorithms for Unification and Other Complete Problems," *Proc. ACM'84 Annual Conf.*, ACM, Oct. 1984, pp. 75–84.
400. J. S. Vitter and R. A. Simons, "Parallel Algorithms for Unification and Other Complete Problems," *Trans. on Computers*, to appear 1986.
401. B. W. Wah and K. L. Chen, "A Partitioning Approach to the Design of Selection Networks," *IEEE Trans. on Computers*, C-33(3):261-268, 1984.
402. 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*, IEEE/ACM, June 1984. pp. 56-63.
403. B. W. Wah and Y. W. E. Ma, "MANIP - A Multicomputer Architecture for Solving Combinatorial Extremum-Search Problems," *Trans. on Computers*, C-33(5):377-390, 1984.
404. B. W. Wah, G.-J. Li, and C. F. Yu, "Multiprocessing of Combinatorial Search Problems," *Computer*, 18(6):93-108, 1985.
405. L. Walker, "Lisp Language Gets Special Machine," *Electronics*, Aug. 25, 1981, pp. 40-41.
406. D. H. Warren, L. M. Pereira, and F. Pereira, "Prolog – The Language and Its Implementation Compared with Lisp," *Proc. Symp. on Artificial Intelligence and Programming Languages*, ACM, Aug. 1977 (*also in SIGART Newsletter*, 64: 109-115).
407. D. A. Waterman and F. Hayes-Roth, *Pattern-Directed Inference Systems*. New York: Academic Press, 1978.
408. B. W. Weide, "Modeling Unusual Behavior of Parallel Algorithms," *Trans. on Computers*, C-31(11):1126-1130, 1982.
409. D. Weinreb and D. Moon, Flavors, Message Passing in the Lisp Machine. AI Memo 602, MIT Laboratories Cambridge, MA, Nov. 1980.

410. 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*, IEEE, Feb. 1985, pp. 71–73.
411. S. M. Weiss and C. A. Kulikowski, "EXPERT: A System for Developing Consulting Models," *Proc. 6th Int'l Conf., Artificial Intelligence*. Los Altos, CA: William Kaufman, 1979, pp. 942–947.
412. G. Wiederhold, "Knowledge and Database Management," *Software* 1(1):63–73, 1984.
413. R. Wilensky, *Lispcraft*. New York: W. W. Norton, 1984.
414. R. Williams, "A Multiprocessing System for the Direct Execution of Lisp," *Proc. 4th Workshop on Computer Architecture for Non-Numeric Processing*, ACM, Aug. 1978.
415. T. Williams, "Semiconductor Memories: Density and Diversity," *Computer Design*, Penn Well, Aug. 1984, pp. 105–116.
416. K. G. Wilson, "Science, Industry, and the New Japanese Challenge," *Proc. IEEE*, 72(1):6–18, 1984.
417. T. Winograd, "Beyond Programming Languages," *Comm. of the ACM*, 22(7):391–401, 1979.
418. L. E. Winslow and Y. C. Chow, "The Analysis and Design of Some New Sorting Machines," *Trans. on Computers*, C-32(7):677–683, 1983.
419. P. H. Winston and B. Horn, *Lisp*, Second Edition. Reading, MA: Addison-Wesley, 1984.
420. M. J. Wise, "EPILOG = Prolog + Data Flow: Arguments for Combining Prolog with a Data Driven Mechanism," *SIGPLAN Notices*, 17(12):80–86, 1982.
421. The Xerox Learning Research Group, "The Smalltalk-80 System," *Byte*, Aug. 1981, pp. 36–48.
422. 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*, IEEE/ACM, June 1983, pp. 363–369.
423. Y. Yamaguchi, K. Toda, J. Herath, and T. Yuba, "EM-3: A Lisp-Based Data-Driven Machine," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 524–532.
424. M. Yamamoto, "A Survey of High-Level Language Machines in Japan," *Computer*, 14(7):68–78, 1981.
425. A. C. C. Yao, "Bounds on Selection Networks," *SIAM J. on Computing*, 9(3):566–582, 1980.
426. H. Yasuhara and K. Nitadori, "ORBIT: A Parallel Computing Model of Prolog," *New Generation Computing*, 2(3):277–288, 1984.
427. P. N. Yianilos, "Dedicated Comparator Matches Symbol Strings Fast and Intelligently," *Electronics*, 1983, pp. 113–117.
428. T. Yokoi, S. Uchida, and ICOT Third Laboratory, "Sequential Inference Machine: SIM—Its Programming and Operating System," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 70–81.
429. M. Yokota, et al., "A Microprogrammed Interpreter for Personal Sequential Inference Machine," *Proc. Int'l Conf. on Fifth Generation Computer Systems*, ICOT and North-Holland, 1984, pp. 410–418.
430. 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*, 1(2):125–144, 1983.