J. Indian Inst. Sci., Mar.–Apr. 1990, 70, 159–186.
© Indian Institute of Science.

IISc THESES ABSTRACTS

Thesis Abstract (Ph.D.)

A knowledge-based approach to pattern clustering by B. Shekar.
Research supervisors: G. Krishna and M. Narasimha Murty.
Department: Computer Science and Automation.

1. Introduction

Clustering a set of objects into groups is a well-known scientific methodology, which finds applications in a variety of domains such as biology, medicine, and sociology. The conventional similarity measures[1] which have been used to perform clustering are free from the influence of context and any *apriori* defined concepts. Though there have been developments in this area, to capture the effects of context[2] and concepts[3] on clustering, the groupings and their associated descriptions (*e.g.* [colour = red] and [shape = round]) have been more at the syntactic than at the semantic level.

2. Contributions of the thesis

In this thesis, we propose a different approach to the clustering of objects. This approach encompasses the usage of the conceptual entities mentioned earlier, and goes further to perform the clustering on the basis of functions of an object. The proposed methodology of clustering utilises explicitly declared knowledge. This knowledge is abstracted from the real world where the objects can be used to execute certain tasks (which we call as semantics of an object(s)).

We first define the term concept, and give it a functional interpretation. Then we use it in establishing a 'semantics-directed clustering mechanism'. We next identify three different categories of cohesiveness which can have an influence on the partitioning of objects (based on the proposed approach) into clusters. To make the resulting clusters more meaningful from the execution of the relevant function point of view, we provide a set of axioms which define the meaning of such clusters. A natural bifurcation of the set of concepts which is used (from the functional angle) in the real world is identified as comprising the necessary concepts and the quality-improvement concepts. Further, we define a graph structure called cohesion forest consisting of N- and Q-trees which represent the necessary concepts and the quality-improvement concepts, respectively. An algorithm to perform this semantics-directed clustering is given, and an example to illustrate the same is presented.

Starting with basic cluster structures which are a consequence of the proposed mechanism, we examine different structures which are composed of these basic structures. We also examine different interpretations which one can assign to the definition of concept. While doing so we observe the relevance with respect to the functional interpretation of a small subset of the set of partial mappings which go into the definition of concept. Consequently, we justify the adequacy of the description connectives (which are used along with the concepts defined in the cohesion forest) to describe the clusters.

We observe a property which we call the undecidable property, namely that there exists a non-empty subset of the constituent objects in a cluster, which cannot always be assigned to a particular concept (defined in the cohesion forest), implied by the cluster under consideration. We identify a sufficient condition for the presence of this property in a given set of objects. The need for a parallel or a quasi-parallel implementation of the proposed functional clustering paradigm is not difficult to justify. The invariance of the resulting cluster descriptions with respect to the order in which the given set of objects is examined plays a key role in any such implementation. This invariance is established for the semantics-directed clustering mechanism in a formal manner using the axioms which define the mechanism.

A separate class of concepts, called the concept transformer class, which displays certain peculiar properties, in addition to satisfying the functional interpretation of concept, is identified and studied in detail. Its role in the formation of virtual clusters (virtual because of its non-existence in the current cluster configuration) in the context of the proposed knowledge-based approach to clustering is examined. The notion of the cohesion forest is extended to capture the properties of the concept transformer class.

As another significant application of the proposed functional approach, we formulate the problem of pattern synthesis from the knowledge-based angle. We also show how the synthesis process can be visualized as a goal-directed pattern-matching problem where the goal may be interpreted as the optimization of a criterion function consisting of more than one parameter; and the pattern-matching activity may be viewed as a repeated process of application of pieces of knowledge from various domains such as physics, chemistry, and the like, to N- and Q-trees. This leads to the formation of concepts, which are either new, or improved versions of the existing ones.

3. Conclusion

The investigations carried out in this thesis have significant potential in the clustering of any set of objects that imply functions (which we call higher level concepts or semantics) by virtue of their physical properties (which we call lower level concepts or syntactic entities).

References

1. Anderberg, M. R.	*Cluster analysis for applications*, 1973, Academic.
2. Gowda, K. C. and Krishna, G.	Disaggregative clustering using the concept of mutual nearest neighbourhood, *IEEE Trans.*, 1978, **SMC-8**, 888–895.
3. Michalski, R. S. and Stepp, R. E.	Automated construction of classifications: Conceptual clustering *versus* numerical taxonomy, *IEEE Trans.*, 1983, **PAMI-5**, 396–410.

Thesis Abstract (Ph.D.)

Computer recognition of handprinted characters: an automated approach to the design of recognizers by V. K. Govindan.
Research supervisor: A. P. Shivaprasad.
Department: Electrical Communication Engineering.

1. Introduction

Computer recognition of handprinted characters is an active field of research since more than three decades. Though a great deal of work has already been done, the ultimate goal of developing a recognizer having all the reading capabilities of humans is as yet unachieved.

A typical recognition system development for an alphabet set involves two main parts, *viz*., learning and recognizer design. The learning part involves collection of samples of the character set, selection of appropriate features, and development of unambiguous prototype class-descriptions. The recognizer design mainly involves sub-system designs for character preprocessing, feature extraction, description generation and analysis for recognition.

2. Design of recognizers

The main theme of the work attempted in this thesis is the suggestion of a descriptive (feature-descriptive) approach and the automation of the design of such recognizers for different handprinted character sets, especially for English (Latin) alphanumerals and a representative South Indian cursive (unconnected) character set.

The characters are described by independent features which directly reflect the structure of the character patterns. A unified and flexible feature representation is employed to deal with character sets of different structural complexities. A feature is represented by details like its location in the character frame, number of limbs diverging from the feature points, and directions, curvatures, lengths, etc., of diverging limbs. The prototype class-descriptions are built-up in terms of such stable and reliable features. The features are generalized by introducing tolerances for their component values to accommodate the deviations between the samples, and also to take care of the writing style not covered by the learning set.

The recognition is effected by comparing the unknown character sample description with that of the prototype after a pre-classification based on feature locations and number of limbs diverging from the feature points. A class-feature-table (CFT) is employed for this purpose.

The description and recognition procedures described above are automated by 1) automating the feature selection problem, and 2) automating the learning of prototype description problem. The feature selection problem is simplified to the selection of the appropriate components of the feature defined earlier. It is done by employing a criterion *viz*., ambiguity of descriptions. The right and the minimum number of components are selected so as to achieve unambiguous class-descriptions. The prototype class-descriptions are learned automatically from a learning sample set in terms of the selected features. The learned descriptions are stored in the class-feature-table to achieve a fast pre-classification. The descriptions are made adaptive to future samples by combining descriptions learned from new samples with the old descriptions.

3. Main results and conclusion

The automated approach provides a very fast way of designing recognizers for different alphabet sets. The prototype structural description learning technique eliminates the designers difficult job of examining the character samples to identify and select valid features of various classes.

Both the manual and the automated approaches are tested on English (Latin) alphanumerals and Malayalam, a South Indian character set consisting of about 45 basic cursive characters and symbols. Reasonably high recognition rates are obtained without the use of any contextual information.

The automated approach is practicable and very suitable for character sets containing less than 100 characters or symbols, and provides promising results for stroke-intensive characters. The extension of the approach to large cursive character sets may be feasible, and worth examining.

References

1. Naylor, W. C.	Some studies in the interactive design of character recognition systems, *IEEE Trans.*, 1971, **C-20**, 1075–1086.
2. Ishii, K., Kanemaki, M. and Komori, K.	Automatic design of a character recognition dictionary based on feature concentration method, *Proc. 4th Int. Jt Conf. Pattern Recognition, Kyoto, Japan*, 1978, pp. 804–806.
3. Kami, H.	Evaluation of automatic dictionary generation for character recognition, *NEC Res. Dev.*, 1984, No. 72, 42–47.

Thesis Abstract (Ph.D.)

Data flow implementation of functional languages supporting fault tolerance and resource management by R. Govindarajan.
Research supervisor: L. M. Patnaik.
Department: Computer Science and Automation.

1. Introduction

The phenomenal advancement witnessed in recent years in VLSI technology encourages the design of massively parallel processing systems. Functional programming (FP) languages have been widely accepted as the programming languages for these future generation computers. This is due to the fact that functional languages have the ability to describe algorithms in a clear, concise, and natural way. The simple semantics and hierarchical structure of these languages have attracted many researchers. Further, they are free from side effects and express parallelism in a quite natural manner. These attractive features have led many researchers to carry out research on functional languages from an implementation point of view. This thesis also focuses on the implementation of FP.

2. A data structure for FP

In implementing functional languages, the inefficiencies associated with the accessing of elements of a complex data structure such as Lists is of major concern. To circumvent these inefficiencies, we have proposed a new data structure by imposing a regularity constraint and by embedding the concepts of 'bulk-arrays[1]' on the well-known I-structures[2]. The proposed data structure, called the restricted

I-structure, can be represented using a tree structure which facilitates sharing of substructures. For the restricted I-structure, we have developed an efficient implementation scheme exploiting the regularity constraint. This implementation scheme facilitates random access to the elements of a restricted I-structure. Also, the implementation scheme allows us to exploit spatial parallelism on this data structure. The superiority of the proposed data structure is more from an implementation point of view.

3. Data flow execution of FP

Data-driven evaluation[3] has been chosen as the model of computation for executing FP. This is because data-flow model of computation exploits parallelism at fine-grain level. We propose to execute FP programs by converting them into data-flow graphs. The data-flow actors, which correspond to the primitive actors of FP, themselves inherit a large extent of parallelism. A distributed execution of these graphs on a data-flow machine promises high performance in terms of both speedup and parallelism exploited.

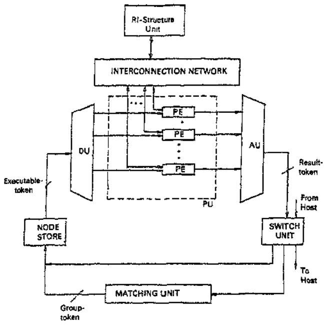
In order to support FP, several hardware and software modifications have been proposed to the Manchester data-flow machine[4]. The hardware modifications include designing structure memory unit, structure allocation and reclamation units and smart memory network interface. With the help of the memory network interface, the generalized synchronization-primitive 'fetch and $\emptyset$' can be implemented in the proposed architecture. This primitive is useful in realizing new implementation schemes for concurrent execution of reentrant routines and lenient execution has been developed. By supporting lenient execution, stream parallelism can be exploited in the production and consumption of a restricted I-structure. Further, a novel garbage collection scheme has been designed to reuse the unreferenced memory cells of restricted I-structures. These various schemes constitute the software modifications to the Manchester machine. We call the modified Manchester machine as Architecture A1 (fig. 1).

4. Evolution of three data-flow architectures and their performance

The effect of the proposed modifications on the Manchester data-flow model has been studied by constructing a simulator on a DEC-1090 system using SIMULA 67. The performance experiments conducted on four benchmark problems show encouraging results. A linear improvement in speedup has been observed when the number of processing elements in A1 is increased from 1 to 8. The reasons for performance saturation have been analyzed by measuring various performance parameters such as queue length, service time and utilization of various functional units.

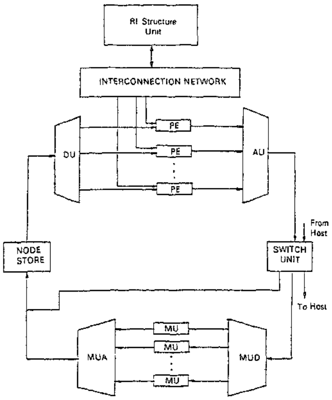
From a careful observation of the performance analysis, we identify the bottlenecks present in the proposed model. The major bottleneck center in A1 is the matching unit. To improve the performance of A1, we replicate the matching unit. This modification calls for two additional functional units, namely, matching unit arbitrator and matching unit distributor. The performance of the modified architecture, called Architecture A2 (fig. 2), has been evaluated using simulation. We observe that the proposed modifications lead to improvement in performance (compared to A1). However, a saturation in speedup still occurs for a small number of processing elements (equal to 12).

The greed to further enhance the performance leads to the replication of the node store unit as well as the matching unit. The resulting architecture (Architecture A3, fig. 3), resembles the multi-ring Manchester machine[5]. The performance of A3 displays tremendous improvement in terms of both execution speed and parallelism exploited (fig. 4). For Architecture A3, the saturation in speedup occurs only when the number of processing elements is 48.



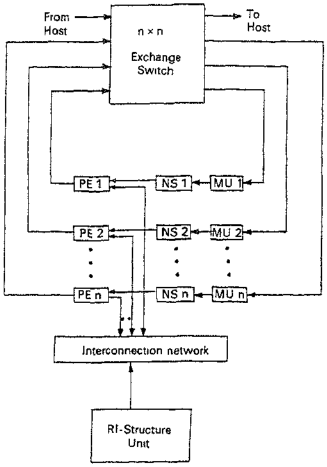
PE: Processing element; PU: Processing unit; DU: Distributor unit; AU: Arbitrator unit.

FIG. 1. Block schematic of architecture A1.



MU: Matching unit; PE: Processing unit; MUA: Matching unit arbitrator; DU: Distributor unit; MUD: Matching unit distributor; AU: Arbitrator unit.

FIG. 2. Block schematic of architecture A2.



PE: Processing element; NS: Node store; MU: Matching unit.

FIG. 3. Block schematic of architecture A3.

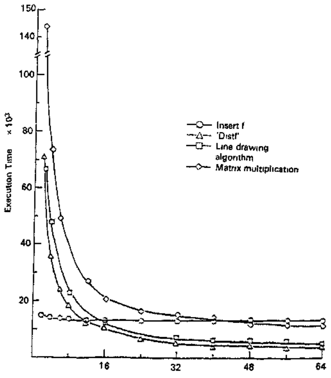


FIG. 4. Speedup test for A3.

5. Experimental hardware system

With the performance evaluation of A3 demonstrating encouraging results, we propose to implement the above architecture. The motivations behind the construction of A3 are to demonstrate the feasibility of the design of the proposed architecture and to gain first-hand experience in working with such systems. Single-board computers based on Intel's 8088 processors have been used for the processing element, switch unit and the structure memory unit. The node store unit and the matching units have been designed to work under microprogram control. The prototype system with four rings is operational.

6. Fault tolerance and resource management

Having studied the implementation of FP on a data-flow architecture, we investigate certain important features which are not present in FP. Notations for handling exceptions[6] and recovery blocks[7] are important from a programming point of view. We have introduced a set of new constructs for supporting exception handlers and recovery blocks in FP. The execution control for these constructs has been derived using the data-flow approach. By exploiting the functionality property of the language, we have devised an extremely simple scheme for the recovery of data. Further, the effects of the new constructs on the algebraic properties of FP have been formally studied.

Synchronization and scheduling of resources are vital tasks in any multiprocessing system. To develop resource management programs in FP, it is essential to support synchronization and non-determinism. For this purpose, we have introduced new communication constructs and guarded functionals which are similar to the input/output commands and guarded commands of CSP[8]. A novel scheme based on data-flow model has been proposed to implement the new constructs. The correctness of the implementation has been established by proving the 'safety' and 'liveness' properties.

7. Conclusion

This thesis concentrates on two important aspects, namely, the implementation and enrichment of functional languages. The proposed implementation of FP using data-flow model exploits stream parallelism, spatial parallelism, and fine-grain asynchronous parallelism. The efficacy of the implementation is well established by the simulation results while the feasibility (of the implementation) has been demonstrated by building the experimental hardware system. New constructs have been proposed to enrich the features of the languages. By adopting data-driven evaluation, simple and elegant schemes have been proposed to implement the new constructs. The enrichments (to FP) proposed in this thesis are more from a pragmatic point. Still, the theoretical soundness of functional languages is maintained making the enriched languages more attractive.

References

1. Keller, R. M. — *FEL (function equation language) programmer's guide*, TR #7, Deptt of Computer Science, University of Utah, Salt Lake City, 1983.

2. Arvind and Thomas, R. E. — *I-Structures: An efficient data structure for functional languages*, Technical Report, *MIT LSC/TM-178*, Lab. for Computer Science, MIT, Cambridge, 1981.

3. Treleaven, P. C., Brownbridge, D. R. and Hopkins, R. P. — Data-driven and demand-driven computer architecture, *Computing Sur.*, 1982, **14**, 93–143.

4 GURD, J R., WATSON, I AND KIRKHAM, C C	The Manchester prototype data flow computer, *Commun. ACM*, 1985, **28**, 34–52.
5. BARAHONA, P M C C. AND GURD, J. R	Processor allocation in a multi-ring dataflow machine, *J. Parallel Distributed Computing*, 1986, **3**, 315–327.
6 GOODENOUGH, J. B.	Exception handling. Issues and a proposed notation, *Commun ACM*, 1975, **18**, 683–696.
7. ANDERSON, T AND KERR, R.	Recovery blocks in action· A system supporting high reliability, *Proc. Second Int. Conf. on Software Engineering*, 1976, pp. 447–457.
8. HOARE, C. A. R.	Communicating sequential processes, *Commun. ACM*, 1978, **21**, 666–677.

Thesis Abstract (Ph.D.)

Applications of finite graph models of context-free languages by B. S. Adiga.
Research supervisors: Priti Shankar and S. Nagabhusana (orgn).
Department: Computer Science and Automation.

1. Introduction

A finite graph model called an extended transition system (ETS) is proposed for the representation of a context-free language. The nodes of the graph encode (PDA state, top stack symbol) pairs, (where PDA is the pushdown automaton that recognizes the context-free language), and the arcs encode (input symbol, change to stack) pairs. The graph displays the dynamic behaviour of the PDA. The equivalence of the ETS and the PDA has been proved, and an algorithm to construct the ETS from an LR grammar is also presented.

2. Proofs of classical theorems

Some classical theorems in Formal Language Theory are the Pumping Lemma, the Chomsky, Schutzenberger theorem and Parikh's theorem. It is shown that using the ETS representation for a context-free language all these results can be proved in a simple and intuitive manner.

3. Decidability issues

Of the many non-trivial questions in formal languages, particularly those related to the properties of context-free languages, there are very few that are decidable. One of these is the question of whether an arbitrary deterministic context-free language (DCFL) is a regular set. Stearns[1] showed that this question was decidable and gave an algorithm of complexity $t^{q^{q^q}}$ for the test; Valiant[2] reduced this bound to t^{q^q} where t is the number of stack symbols and q the number of states of the deterministic pushdown automaton recognizing the deterministic context-free language. It is shown that using the ETS representation for the DCFL, a test of complexity $0(qt)^4 t^{4(qt)^4}$ can be derived to check the regularity of the DCFL.

The error repair of context-free languages has attracted considerable attention by researchers, particularly the error repair of DCFLs as this class includes practical programming languages. The best least-cost repair algorithm shown has complexity $0(n^3)$ where n is the length of the input string.

It is shown that if one restricts the class of errors to substitution errors only then a linear time algorithm exists for the near minimal repair of a subclass of DCFLs.

Some extensions of the work have been proposed.

References

1. Stearns, R. E.	A regularity test for pushdown machines, *Inf Control*, 1967, **11**, 323–340.
2. Valiant, L. G.	Regularity and related problems for deterministic pushdown automata, *JACM*, 1975, **22**, 1–10.

Thesis Abstract (Ph.D.)

Concurrency control algorithms for database systems exploiting the semantics of read-only transactions by Ramesh Chandra Hansdah.
Research supervisor: L. M. Patnaik.
Department: Computer Science and Automation.

1. Introduction

Concurrency control algorithms greatly influence the performance of both centralized and distributed database management systems. As a result, considerable attention has been paid in the literature to the design of efficient concurrency control algorithms for both centralized and distributed database systems[1-3]. In distributed databases, it is also necessary that the concurrency control algorithms are resilient to site failures and/or network partitioning[4]. That is, they operate correctly even in the presence of site failures and/or network partitioning. There are essentially four techniques to design a concurrency control algorithm. These four techniques are: (a) locking, (b) timestamping, (c) serialization graph checking, and (d) the use of readsets and writesets of transactions. Most concurrency control algorithms that do not use some form of locking suffer from the problem of cascading abortion of transactions. As a result, most commercial and prototype database management systems use locking as the concurrency control mechanism[5-7]. Even in a locking-based concurrency control algorithm, if the problem of cascading abortion of transactions is to be avoided, then the exclusive(X) mode locks in an update transaction T must be held till the commitment of T. Read-only transactions lock data items using shared(S) mode locks, and therefore, they can be allowed to be non-two phase without causing the problem of cascading abortion of transactions.

This thesis proposes new locking-based concurrency control algorithms for both centralized and distributed databases. These algorithms exploit the semantics of read-only transactions to allow the read-only transactions to be non-two phase, and thereby, the semantics of read-only transactions help to improve concurrency without causing the problem of cascading aborts. The distributed concurrency control algorithms proposed are also resilient to site failures and/or network partitioning.

2. Main contributions

Three semantics of read-only transactions are identified and defined. They are: (a) serializability, (b) weak consistency, and (c) semiweak consistency. The read-only transactions which are specified as either weakly consistent or semiweakly consistent are called non-serializable. The concurrency control algorithms proposed in this thesis require that all the non-serializable read-only transactions are specified as either weakly consistent or semiweakly consistent. Most concurrency control algorithms proposed in the literature aim to achieve view serializability[8]. This thesis defines two new types of serializability. They are: (a) strong update serializability, and (b) weak update serializability. Strong update serializable schedules are always view serializable but not *vice versa*. In addition, weak update serializable schedules are always strong update serializable but not *vice versa*. We also prove results that can be used to show that a locking-based concurrency control algorithm ensures strong or weak update serializability in both centralized and distributed databases.

The weak consistency of read-only transactions is used to improve concurrency in heterogeneous locking protocols. We show that if a locking protocol P that uses X mode locks only and that ensures view serializability is made heterogeneous, then the resulting heterogeneous locking protocol P' ensures strong update serializability. However, if the read-only transactions are two phase in P', then P' ensures view serializability. Hence, a read-only transaction can remain serializable in P' just by being two phase. If a user is satisfied with the weak consistency of a read-only transaction T, then T can be allowed to be non-two phase, and thereby the weak consistency of a read-only transaction helps to improve concurrency in heterogeneous locking protocols. We also present a sufficient condition that ensures the deadlock-freeness of P'.

This thesis also proposes a non-two-phase locking protocol, called the SGN2PL protocol, for general databases. The SGN2PL protocol recognizes three types of transactions. They are: (a) update transactions, (b) serializable read-only transactions, and (c) non-serializable read-only transactions. The SGN2PL protocol requires that all the non-serializable read-only transactions are specified as weakly consistent or semiweakly consistent. Only the non-serializable read-only transactions are allowed to be non-two phase in the SGN2PL protocol, and therefore, the semantic information that a read-only is weakly or semiweakly consistent helps to improve concurrency. If the non-serializable read-only transactions are specified as weakly consistent, then the SGN2PL protocol ensures strong update serializability; however, if the non-serializable read-only transactions are specified as semiweakly consistent, then the SGN2PL protocol ensures weak update serializability. The proof of correctness of the SGN2PL protocol is also given. Simulation results indicate that the performance of the SGN2PL protocol in terms of average response time and throughput of transactions is considerably better than that of the two-phase locking protocol.

We have extended the SGN2PL protocol to distributed databases and we call this extended protocol 'the distributed SGN2PL protocol'. The distributed SGN2PL protocol is essentially like the basic 2PL protocol[1] and it recognizes the same three types of transactions as the SGN2PL protocol. In addition, in the distributed SGN2PL protocol, the non-serializable read-only transactions are assumed to be local. The distributed SGN2PL protocol ensures weak update serializability irrespective of whether non-serializable read-only transactions are specified as weakly or semi-weakly consistent. However, it ensures strong update serializability if the non-serializable read-only transactions are specified as weakly consistent and if the following two conditions are satisfied: (a) The distributed database is fully replicated, (b) All the update transactions request locks in mode X only. The proof of correctness of the distributed SGN2PL protocol is also given. The sample simulation results indicate that the distributed SGN2PL protocol performs relatively better than the basic 2PL protocol.

The distributed SGN2PL protocol is not resilient to site failures and/or network partitioning. This thesis proposes a model of distributed systems that is helpful to describe a distributed database system in the presence of failures. We have used this model of distributed systems to design a resiliency control scheme for replicated distributed databases. Using the resiliency control scheme, we have presented a distributed resiliency control algorithm that can be used to make a basic 2PL-like distributed locking protocol resilient to site failures and/or network partitioning. The distributed resiliency control algorithm is used to make the distributed SGN2PL protocol resilient to site failures and/or network partitioning. The essential features of the distributed resiliency control algorithm are as follows.

(a) A data item x is accessible to the update transactions even if there is only one operational site storing a copy of x, provided the following two conditions are satisfied.
   (i) Site failures are detectable.
   (ii) A site storing a copy of x has not failed after the occurrence of network partitioning.
(b) A user may be prevented from accessing the database only for a short duration after the occurrence of the failure.

This thesis also proposes a site recovery and partition merge algorithm using the same scheme as that used by the distributed resiliency control algorithm. The essential features of this algorithm are as follows.

(a) The site recovery and partition merge operations are treated in a unified way, *i.e.*, no distinction is made between site recovery and partition merge operations.
(b) An obsolete copy of a data item x need not be made up to date as soon as a site recovers or a partition merge takes place.
(c) The up-to-date copy of a data item x can be traced even if there is no operational site storing a copy of x.

References

1. BERNSTEIN, P A. AND GOODMAN, N.	Concurrency control in distributed database systems, *ACM Computing Surv.*, 1981, **13**, 185–221.
2. BERNSTEIN, P. A., HADZILACOS, V. AND GOODMAN, N.	*Concurrency control and recovery in database systems*, 1987, Addison-Wesley.
3. CELLARY, W., GELENBE, E. AND MORZY, T.	*Concurrency control in distributed database systems*, 1988, Elsevier.
4. DAVIDSON, S. B., GARCIA-MOLINA, H. AND SKEEN, D	*Consistency in partitioned networks, ACM Computing Surv.*, 1985, **17**, 341–370.
5. ASTRAHN, M. M. *et al*	System R: A relational approach to database management, *ACM Trans. Database Systems*, 1976, **1**, 97–137.
6. CERI, S. AND PELAGATTI, G.	*Distributed database: Principles and systems*, 1984, McGraw-Hill.
7. STONEBRAKER, M., WONG, E., KREPS, P. AND HELD, G.	The design and implementation of INGRES, *ACM Trans. Database Systems*, 1976, **1**, 189–222.
8. YANNAKAKIS, M.	Serializability by locking, *J. ACM*, 1984, **31**, 227–244.

Thesis Abstract (Ph.D.)

Performance analysis of a multiprocessor machine based on data-flow principles by Ranjani Narayan.
Research supervisor: V. Rajaraman.
Department: Computer Science and Automation.

1. Introduction

Extensive research is in progress on multiprocessor architectures[1-5], with the primary aim of reducing the execution time of programs by cooperative functioning of processors. Among the various proposed multiprocessor architectures, the static data-flow architecture[6] has attracted considerable attention as it radically differs from von Neumann architecture and has an excellent potential of utilizing inherent parallelism in programs at the 'fine-grain' level. This architecture, however, has high communication cost as it attempts to exploit parallelism at 'fine-grain' level.

In this thesis we propose a model of a static data-flow-oriented parallel computer and study issues pertaining to the scheduling and communication delays with the aim of improving the efficiency of the system.

2. The multiprocessor machine model

The proposed multiprocessor system consists of the following components: 1. a bank of homogeneous processing elements (PEs), 2. a special processing element which holds global memory, which is conceived as I-Structure memory[7], 3. three broadcast type communication media, 4. a conventional von-Neumann peripheral processor (referred to as host or I/O processor) and 5. a scheduler.

We model the proposed machine (fig. 1) to evaluate its performance under three scheduling policies *viz.*, dynamic, static and quasi-dynamic, for different machine configurations. Since no work is

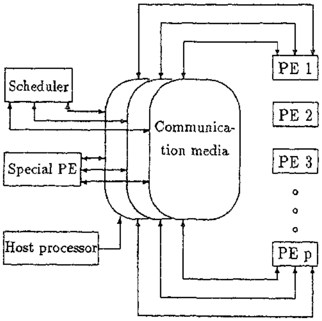


FIG. 1. Schematic diagram of the proposed machine.

reported in the literature which systematically analyses the problem using a general model of computation[8] to clearly bring out the trade-offs between the number of PEs in the system, communication delays in transmitting operations and operands to PEs and the number of operations executed in each PE, in our study we attempt at defining a 'coarse-grain' model of the machine to obtain a balance between computation and communication time in the system. We initially develop an idealised model of the machine for evaluating its performance for a general class of problems. To facilitate this, we make certain assumptions about a given program P (its parallelism, data dependencies, etc.) and study the behaviour of the machine. We assume that the most 'fine-grain' data-flow graph G_k, a k node graph of P is given, with specified computation time per node. We evaluate the machine for a set of parameters that define the machine attributes. We then vary the number of nodes to obtain different data-flow graphs of P which capture the resident parallelism in P to different extents. For each graph of P, we repeat the performance analysis. This study leads us to an m node graph, G_m of P, which will minimize the execution time of P in the system (with the specified set of parameters). Each node of the graph G_m incorporates several nodes of G_k. In other words, G_m is a coarse-grain data-flow graph of P. We repeat the analysis for a variety of parameters (that configure the machine in different ways) so that we can arrive at that set of parameters which best defines the attributes of the machine.

Next we extend the study for evaluating the machine for executing application-specific problems. We relax our assumptions about the nature of graphs representing the problems. The scheme used to model the machine closely follows that used for the idealized model.

This analysis helps us determine the extent of 'coarseness' that should be resident in the data-flow graph representation of any program $P1$ (for a set of machine attributes) to execute $P1$ in minimal time. (The most fine-grain data-flow graph G_{k1} of $P1$ is assumed to be available.) Thus, to obtain the best representation of $P1$, there is a need to reduce G_{k1} to the best graph G_{m1}. In this thesis, we also develop an algorithm to coalesce the nodes of G_{k1} to obtain G_{m1} by (a) combining linear sequences of instructions, and (b) coalescing instructions that would reduce the volume of data to be communicated amongst themselves.

The results of the analysis are used to bring out the design criterion of the scheduler. A scheduler which performs its functions in a distributed fashion is proposed to minimize its overhead during program execution.

In this study, we measure the extent of PE utilization during program execution. We also derive performance measures that provide a comparative evaluation of the machine with ideal uniprocessor and realistic uniprocessor machines.

3. Results and conclusions

Results obtained by the idealized model provide a guideline to identify the best suited graph representation of a problem. The algorithm developed for coalescing a fine-grain data-flow graph to the desired graph serves to represent the problem by the best suited graph. The model tuned to execute application-specific programs can then be used for obtaining a good estimate of the execution time on an actual implementation of the machine. This evaluation study also enables us to decide on the best design of the machine for a class of problems.

The study concludes with the following observations. The machine achieves reasonable speedup for large programs (of the order of 10,000 instructions) with eight processing elements using dynamic scheduling strategy. This strategy, however, imposes the requirement of a high-speed scheduling unit and communication media. In contrast, the 8-PE machine using static scheduling achieves comparable speed up with slower components. The 8-PE machine using quasidynamic scheduling strategy is better than that using static policy by permitting even slower scheduling units, which are easily realisable in hardware.

References

1. ARVIND AND IANNUCCI, R. A.	A critique of multiprocessing von-Neumann style, *Proc. 10th Annual Inter. Symp. on Computer Archit.*, 1983, pp. 243–250.
2. TRELEAVEN, P. C., BROWNBRIDGE, D. R. AND HOPKINS, R. P.	Data-driven and demand-driven computer architecture, *ACM Computing Surv.*, 1982, **14**, 93–143.
3. RUMBAUGH, J.	A data-flow multiprocessor, *IEEE Trans.*, 1977, **C-26**, 138–146.
4. ARVIND, KATHAIL, V. AND PINGALI, K.	*A data-flow architecture with tagged tokens*, Technical Memo 174, Lab. for Computer Science, MIT, Sept. 1980.
5. GURD, J. R., WATSON, I. AND KIRKHAM, C. C.	The Manchester prototype data-flow computer, *Commun. ACM*, 1985, **28**, 34–52.
6. DENNIS, J. B.	Data-flow supercomputers, *Computer*, 1980, **13**, 48–56.
7. ARVIND, NIKHIL, R. S. AND PINGALI, K. K.	I-structures: Data structures for parallel computing, *Proc. Workshop on Graph Reduction*, Los Alamos, NM, Sept–Oct, 1986.
8. GAUDIOT, J. L. AND ERCEGOVAC, M. D.	Performance evaluation of a simulated data-flow computer with low-resolution actors, *J. Parallel Distributed Computing*, 1985, **2**, 321–351.

Thesis Abstract (Ph.D.)

Algorithms for geometric design rule checking of VLSI layouts in distributed architectures by Soumitra Kumar Nandy.
Research supervisor: V. Rajaraman.
Department: Computer Science and Automation.

1. Introduction

Nowadays electronic systems are invariably built out of integrated circuit components. Improvements in fabrication technology has resulted in the development of very large scale integrated (VLSI) circuits, comprising several hundred thousand transistors. The design of such complex VLSI circuits is very expensive and requires major design effort. In the physical design of VLSI circuits, the behavioral or structural representations are transformed into the geometric shapes that are used in the fabrication of the system. The process of automatic synthesis, analysis and verification of a design is commonly referred to as physical design automation. Considerable research has been focussed on the development of physical design tools to aid in the design of VLSI circuits. Faster and efficient algorithms to support various CAD tools for design automation of VLSI circuits are still evolving.

Associated with the continually increasing complexity and size of chips (measured either in terms of silicon real estate or number of transistors) is the amount of loss incurred when a fabricated chip fails to work. This makes layout analysis an important requirement in the physical design automation process. Layout analysis is the process of performing computations on polygons (layout geometries) to derive information about a chip. This analysis is carried out to predict possible error sites in the layout, prior to its fabrication, within the acceptable tolerance of process variations.

A layout analysis tool that verifies geometrical interactions, such as intersection between two or more layout geometries (LGs), finding LGs within a specified distance from a point of reference, or detecting of point enclosures of LGs is referred to as a design rule checker (DRC). In practical applications, the number of such LGs run into a few millions. Whereas a native algorithm that processes an input size N in $O(N^2)$ time is acceptable for low values of N, the amount of computation is prohibitive for problems with input size $N \approx 10^6$. In addition to the complexity of the algorithm that performs DRC, the size of the layout problem imposes other constraints of storage, often resulting in thrashing on a computer with large virtual memory, but small main memory. Consequently, layout representation plays a very important role in the performance of a DRC tool.

We observe that quadtrees have been effectively used as a hierarchical bitmap data structure to represent regions of an image in image processing applications[1]. However, they have not been used to represent VLSI layout data. We establish the suitability of quadtree as a data structuring technique for VLSI layout tools and propose a new data structure called dual quadtree for hierarchical VLSI designs[2].

2. Layout representation and DRC

Kedem[3] proposed the quad-CIF tree structure for hierarchical design of VLSI circuits. His main idea was that of overlaying a tree of coordinates on top of a hierarchical representation of a layout. Layout geometries are encoded as maximal rectangles and stored in CIF form in the quad-CIF tree. Consequently, operations like identification of all rectangles that comprise a layout geometry, identification of overlapping layout geometries, zooming in and out of a window, etc. involve a lot of search operations and can result in significant computation time for reasonably large layouts. In our proposal the dual quadtree structure represents a cell (in the layout) at the first level of the dual structure as a linear quadtree in order to support efficient region queries. The linear quadtree stores the entire region spanned by all the layout geometries in the cell. At the second level of the dual structure, the bounding boxes of the cells are stored as objects in a quad-CIF tree. Since a cell may be repeated several times in more than one place in the design, storing of layouts of each cell would require a lot of storage. In order to avoid duplication, we store pointers to a single instance of a cell along with appropriate transformation in the dual structure. Therefore the dual structure affords us the convenience of using the painted quadtree for interactive design and the excellent features of the quad-CIF tree for hierarchical design and compact storage representation.

Based on the quadtree representation of VLSI mask layouts, we develop a linear time algorithm to perform DRC[4]. DRC is achieved by following the boundaries of layout geometries. We provide extensions to this algorithm to perform both on-line and hierarchical DRC.

3. Accelerating DRC

We observe that a DRC tool is associated with large CPU execution time. In order to handle large designs and to reduce the time for DRC, there is a need for accelerating DRC. Since the nature of VLSI lends itself to high degree of parallelism, layout tools like DRC can be accelerated by exploiting geometrical locality. Algorithms for CAD tools can thus be developed for both parallel and distributed computing systems to exploit coarse-grain parallelism. In order to accelerate DRC that exploits parallelism at fine grain we need hardware solutions that can hide the cost of interprocess communications. In either case, the optimal solution depends on the nature of the requirement, design environment, methodology, number of users and uses, etc., and the value associated with the individual factors.

However, there exists to the best of our knowledge (at the time of writing this thesis) no DRC tools that attempt accelerating DRC by execution of DRC tasks in a distributed computing environment[5] We investigate this method of accelerating DRC in the thesis. An instance of DRC, together with a layout partition (called DRC task), is executed in a distributed computing environment, by utilizing the idle CPU cycles of workstations connected in an existing LAN. Overlaps between two layout partitions at the boundaries by an amount called design rule interaction distance (DRID) ensures that no false errors are reported at the partition boundaries. The novelty of such an algorithm is that speedup in DRC is obtained at no extra cost.

We provide a heuristic layout partitioning algorithm that minimizes the overheads of performing overlapped DRC at the partition boundaries. We also propose a distributed task allocation strategy that strives to attain load balancing amongst the processors in the network. Performance guarantees for the partitioning and task allocation algorithms are developed

We extend the sequential algorithm for DRC to be realized on mesh-connected processors and the hypercube architecture[6]. This is achieved by developing an algorithm to follow the boundaries of layout geometries in parallel[1]. The technical issues of combining partial results of DRC generated by the PEs in the mesh and hypercube captures the novelty of the algorithm.

Algorithms for hardware acceleration of DRC[7,8] are developed, based on a bitmap representation of a layout. This is an extension to the earlier proposed algorithms for DRC on mesh-connected processors. It can be observed that in the limiting case, when the number of processors in the mesh equals the number of leaf nodes in a quadtree (representing a layout), the layout geometries or their partitions that are allotted to a processor reduce to that of a pixel in the bitmap representation of the layout.

The algorithm for hardware acceleration of DRC is based on a window scan and uses a flexible window size. The time taken to perform DRC is independent of the length of the rule being verified. This algorithm is suitable for realization in hardware since it is possible to develop hardware DRC modules (that perform DRC on layout partitions of size 2×2 pixels) and can be interconnected in an array to perform design rule checking on larger layout partition.

4. Results and conclusions

In this thesis we have established that the quadtrees are well suited to represent layouts of cells for the physical design of large-scale integrated circuits. We proposed a dual quadtree structure to represent a layout.

We developed linear quadtree algorithms for neighbor finding and boundary following of layout geometries. Based on these quadtree algorithms, we developed an algorithm for performing design rule checks of layouts. We have shown that the execution time performance of algorithm for design rule checking is linear with respect to the number of layout geometries.

We observe that a design rule checking (DRC) tool is associated with large CPU execution time. In order to handle large designs and to reduce the time for DRC, there is a need for accelerating DRC. We proposed three solutions to accelerating DRC, *viz.* distributed, parallel and hardware.

A distributed DRC algorithm was developed to exploit parallelism in DRC by exploiting geometric locality in layout geometries. In a distributed computing environment, we have shown that DRC can be performed simultaneously on different partitions of a layout.

We also show how the sequential algorithm for DRC can be extended for its realizations on mesh-connected processors and the hypercube architecture. This is achieved by developing an algorithm to follow the boundaries of layout geometries in parallel. The technical issues of combining partial results of DRC, generated by the PEs in the mesh and hypercube, which captures the novelty of the algorithm are also described in detail. Through a complexity analysis, we establish that the parallel realization of the DRC algorithm can achieve linear speedup with respect to the number of processors in both the mesh and the hypercube.

The algorithm for hardware acceleration of DRC is based on a window scan and uses a flexible window size. The time taken to perform DRC is independent of the length of the rule being verified.

References

1. Panwar, R. B. and Nandy, S. K	Parallel architecture for boundary following of regions of an image stored as a linear quadtree, *Proc. 26th Annual Allerton Conf. on Communication, Control and Computing*, September 1988, pp. 1025–1034.
2. Nandy, S K and Ramakrishnan, I. V.	*Dual quadtree representation for VLSI designs, Proc. 23rd ACM/IEEE Design Automation Conf.*, Las Vegas, July 1986, pp. 663–666.
3. Kedem, G.	The quad-CIF tree: A data structure for hierarchical on-line algorithms, *Proc. 19th ACM/IEEE Design Automation Conf.*, 1982, pp. 352–357
4. Nandy, S. K. and Patnaik, L. M.	Linear time geometrical design rule checker based on quadtree representation of VLSI mask layouts, *Computer-Aided Des.*, 1986, **18**, 380–388.
5. Nandy, S. K.	Geometric design rule check of VLSI layouts in distributed computing environment, *Inter. J. Computer-Aided VLSI Des.* (to appear).
6. Nandy, S K., Rajat Moona and Rajagopalan, S.	Linear quadtree algorithms on the hypercube, *Proc 1988 Inter. Conf. on Parallel Processing*, 1988.
7. Bhat, N. B. and Nandy, S. K.	New algorithms for hardware acceleration of DRC, *Proc. 2nd Inter. Workshop on VLSI Design*, Bangalore, India, 1988, pp. 382–413
8. Bhat, N. B. and Nandy, S. K.	Special purpose architecture for accelerating bitmap DRC, *Proc. 26th ACM/IEEE Design Automation Conf*, Las Vegas, 1989 (to appear).

Thesis Abstract (Ph.D.)

The numerical integration of ordinary differential equations on multiprocessing systems by S. K. Ghoshal.
Research supervisor: V. Rajaraman.
Department: Computer Science and Automation.

1. Introduction

The rapid ongoing advances in VLSI technology have reduced the price of computing hardware to such a low level that applying multiple processors to solve application problems has become a useful and viable option. Such multiple-microprocessor computing systems need parallel algorithms and task-partitioning schemes appropriate to their architecture to solve application problems with high speedups and efficiencies. Integration of systems of ordinary differential equations is an important application problem area. Many good sequential algorithms have been implemented on contemporary computing systems. Digital computers dedicated to solving ordinary differential equations have also been built in the past.

With the advantages of being able to integrate many low-cost, medium-performance microprocessors, in one multiprocessing system, we look for parallel computer architectures suitable for solving ordinary differential equations.

The architecture should be capable of solving the differential equations, with such high speeds, which hitherto were possible only with analog computers. At the same time, the programmability, ease of use and the degree of perfection available only with the state-of-the-art sequential algorithms have to be retained in the multiprocessing environment. The parallel algorithm should be able to utilize the hardware present in the multiprocessor with the maximum efficiency towards solving user problems.

A study of the state of the art of solving ODEs is made for both sequential and parallel integration of ODEs. This lets us put down, in a precise way, the desirable properties of the parallel computers and the algorithms that would run on them to solve ODEs. We select the process view of analog computation as the model of integration of ordinary differential equations.

2. Algorithm

A parallel algorithm[1] is developed thereafter to solve ordinary differential equations. We decide to develop a set of parallel multistep predictor-corrector formulae, with more than one correction being performed in parallel with the prediction. We establish the consistency, zero-stability and convergence of the finite step approximation formulae.

The set of parallel finite-step approximation formulae is optimized[2] for maximal performance in a general multiprocessing context. The stability of the finite step algorithm is improved for larger step-sizes by modelling the propagation of errors using a matrix notation, and applying Rosenbrock's nonlinear optimization techniques. Numerical experiments are performed to study the effects of optimization.

We develop parallel techniques[3] to handle variable steps. The techniques are so designed as to cause the least amount of interprocessor communication during a step-size change. We optimize the techniques to provide the maximum interval of absolute stability, to cope with the eventuality of worst-case driving by an interval-oriented integrator.

We develop error-estimators that can be computed in parallel with the prediction and corrections. The error estimators are so designed as to provide stable and reliable estimates with back values stored both for equally and unequally spaced mesh-points.

We formulate the requirements of the driver program for a parallel initial-value integrator. The scheme that the driver program would use in order to decide upon the next step-size and order to be used is finalized.

We thus combine the approximation formulae, the error estimators, the technique to handle variable steps and the scheme to select the next step-size and order, into an interval-oriented parallel algorithm (called a 'method of integrating systems of ordinary differential equations' by contemporary numerical analysts, dealing with the sequential integration of ordinary differential equations.)

3. Comparison

We upgrade other parallel finite-step approximation formulae to complete methods, using similar design and optimization techniques. This makes the basis of comparison well balanced, fair and uniform.

We decide on a benchmark architecture that is unbiased, simple and realistic. We select a subset of the well-known Hull's set of problems as the benchmarks. We put down a set of comparison criteria, that is of interest to the computer architect, the numerical analyst, and the end-users of the parallel initial value integrator. We use all these methods to solve many systems of ODEs in simulated environments. Sequential methods are also used to measure the effective speedups.

The fact that our algorithm is acceptable for solving ODEs and is competitive with other parallel algorithms is brought out by extensive simulation.

The simulation studies also indicate a potential drawback in the task-partitioning schemes used hitherto to solve ODEs. We design a task-partitioning scheme to alleviate this. Another set of simulation studies to evaluate our task-partitioning scheme is then carried out. This study establishes the superiority of our task-partitioning scheme, and points us to the ideal architecture of a multiprocessing digital differential analyser[4]. We notice that it has functional and morphological similarity to an analog computer.

4. A digital differential analyser

We proceed to build such a digital differential analyser[5]. The necessary system software and runtime systems to implement a high-level language multiprocessing compiler is then designed and implemented. A small four-processor machine is thus realized that can solve differential equations with speedups ranging from 1.2 to 3.6 depending on the system of equations. We analyse the lack of speedup in certain classes of ordinary differential equations, and suggest the appropriate remedies.

5. Conclusion

We thus propose to grow the multiprocessor in such a way, that it may also serve well as a general-purpose multiprocessor. We observe that our algorithm and the task-partitioning scheme need improvements in practical implementation. We propose changes in the design of the hardware, system kernel, and high-level language compilers, which will make the parallel integration more efficient.

We conclude the thesis by pointing out the rules of the design of efficient parallel algorithms for initial-value integration. We provide guidelines towards building large multiprocessing systems that can solve ordinary differential equations, and the design and implementation of efficient parallel algorithms and task-partitioning schemes for such multiprocessors.

References

1. Ghoshal, S. K., Gupta, M. and Rajaraman, V. — A parallel multistep predictor corrector algorithm for solving ordinary differential equations, *J. Parallel Distributed Computing*, 1989, **6**, 636–648.

2. Ghoshal, S. K. and Rajaraman, V. — Increasing stability interval of parallel multistep predictor corrector methods, *Proc. National Seminar on Parallel Processing Systems and their Applications*, Institution of Engineers, Calcutta, India, December 9–11, 1988.

3. GHOSHAL, S. K. AND RAJARAMAN, V. — Optimally stable parallel techniques for handling variable steps in initial value integration, *Computer Sci., Inf.*, 1987, 17, 22–38.

4. GHOSHAL, S. K. AND RAJARAMAN, V. — A parallel digital differential analyzer, *Proc. Indo-US Workshop on Spectral Analysis in One and Two Dimensions*, New Delhi, November 27–29, 1989.

5. GHOSHAL, S. K., GUHA, S., ARIFF, S. M. AND RAJARAMAN, V. — A simple low-cost multiprocessor based on message passing FIFO links, accepted for publication in *Microprocessors Microsystems*.

Thesis Abstract (M.Sc.(Engng))

Learning from examples using hierarchical counterfactual expressions by Malini Krishnan.
Research supervisor: M. Narasimhamurty.
Department: Computer Science and Automation.

1. Introduction

In this study, we develop algorithms for learning concepts from examples.

Learning is the capability that allows a system to improve its performance. It involves the ability to correct errors, learn domain knowledge, generate algorithms, formulate new abstractions and solutions by observing the environment or by drawing analogies to past experiences. Machines with such abilities would have an edge over their human counterparts as they do not suffer from poor memory, distracted attention, low efficiency and the difficulty of transferring acquired knowledge from one to another.

Machine learning manifests itself as a spectrum of information processing[1]. Learning by induction involves presenting a system with labelled examples. The system must generalize the examples and find higher level rules, which can be used to guide the system more efficiently and effectively.

A flexible data structure is required to represent knowledge and allow for the representation of any internal structure in the data. Additionally since knowledge is dynamic, it is necessary that the data structure be easily modifiable to incorporate the new knowledge.

2. Contribution of the thesis

We have developed algorithms to learn concepts using the flexible data structure HC-expressions[2] to represent them. HC-expressions are tree like. The nodes represent a subset of the event space. The arcs that are labelled positive and negative on alternate levels, lead to positive and negative exception? nodes.

1) We implemented algorithm 'Oneshot' which learns a concept description given all the positive and negative events together. It was used on Ayurvedic data on jaundice collected from the Sri Jayachamarajendra Institute of Indian Medicine, Bangalore. The descriptions generated were simple and the time taken little.

2) An incremental algorithm was designed and implemented which developed the concept description consuming one event at a time. The algorithm avoided the pitfalls of incremental learning schemes by introducing controlled generalization and specialisation. This was done by using a 'similarity' measure as in clustering in pattern recognition. If there were too many exceptions to a node

artitioned in the event space. The algorithm yielded simple descriptions and needed less time ; Oneshot algorithm.

ble parallelism is discussed: a) Within a single HC-expression by evaluating sibling nodes ently. Feasible on a highly parallel machine with little communication overheads, and b) By ; two or more HC-expressions. The input data could be partitioned into subsets and ressions generated for each of them in parallel. Later these could be merged.

stly we compare and contrast the following learning algorithms: Candidate elimination on spaces[3]. ID3, ID4 and ID5[4] which build decision trees, ExceL[5] that generates rules with ns, Counterfactuals that generates multilevel counterfactuals[6] and HC-expressions. The son is on: input and rule representation, background knowledge and evaluation functions ., expression of disjunctive concepts, incremental learning ability, handling of noise, and parallelism.

ces

ALSKI, R. S	Understanding the nature of learning. Issues and research directions, in *Machine learning. An artificial intelligence approach*, Vol. II, R. S Michalski, J. G. Carbonell and T. M. Mitchell, (eds), Morgan Kaufmann, Los Altos, Calif., 1986.
I, C. S	RACE: A system for rule acquisition from examples, M. E. Thesis, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, 1987.
ELL, T. M.	A candidate elimination approach to rule learning, *Int. Jt Conf. on Artif. Intell.*, Vol. 5, 305–310, 1977.
F, P E.	*ID-5. An incremental ID3*, Coins Technical Report (Massachusetts at Amherst), 87–95, Dec. 30, 1987.
R, J. M.	Inductive learning of decision rules with exceptions: Methodology and experimentation, MS Thesis, University of Urbana Champaigne, 1985
S. A.	Multilevel counterfactuals for generalizations of relational concepts and productions, *Artif. Intell.*, 1980, 14, 139–164.

Abstract (M.Sc.(Engng))

ti-ring dataflow architecture for parallel execution of logic programs by Sastry.
h supervisor: L. M. Patnaik.
nent: Computer Science and Automation.

luction

blems in artificial intelligence (AI) are highly search-intensive requiring enormous amount of tion time on a sequential computer. This has motivated the researchers to design parallel r architectures for the efficient execution of AI programs. The basic idea of this approach is to se the search space into several subspaces and to search for the solution in each of these s in parallel.

The theme of our work is the design of a multi-ring dataflow architecture for efficient execution of logic programs. The motivation for choosing a logic language for designing parallel machines is that logic languages have good knowledge-based inferencing capabilities; therefore, they are highly suitable for AI applications. The parallelisms exploited are the OR-parallelism, the Argument parallelism, and the Stream AND-parallelism. An elegant scheme for handling deferred read requests generated by the I-structure memory[1] is proposed in this thesis. A novel scheme for maintaining multiple binding environments is also discussed.

The performance evaluation of the proposed multi-ring architecture for studying its efficacy in OR-parallel execution is carried out by executing a few sample logic programs on a simulator developed in a discrete event simulation language SIMULA 67 and implemented on a DEC 1090 system. The performance parameters studied are the speedup, efficiency, and resource utilization. Schemes for the dataflow implementation of two languages known as the flat concurrent Prolog (FCP) and FUNLOG are discussed.

2. The multi-ring dataflow architecture

The dataflow model is an attractive alternative for the von Neumann model for building parallel machines. Our architecture is based on the Manchester ring[2,3]—a dynamic dataflow machine. We have introduced one more unit to the machine which we call the definition search unit. The architecture is organized as rings of hardware units which communicate with one another through a multistage interconnection network as shown in fig. 1. The three types of rings are 'processor ring' containing a processor, a node store and a matching unit, 'memory ring' containing a memory module and 'definition ring' containing the definition search unit. For OR-parallel execution, the function of simultaneously selecting the candidate clauses for unification is accomplished by the definition search unit.

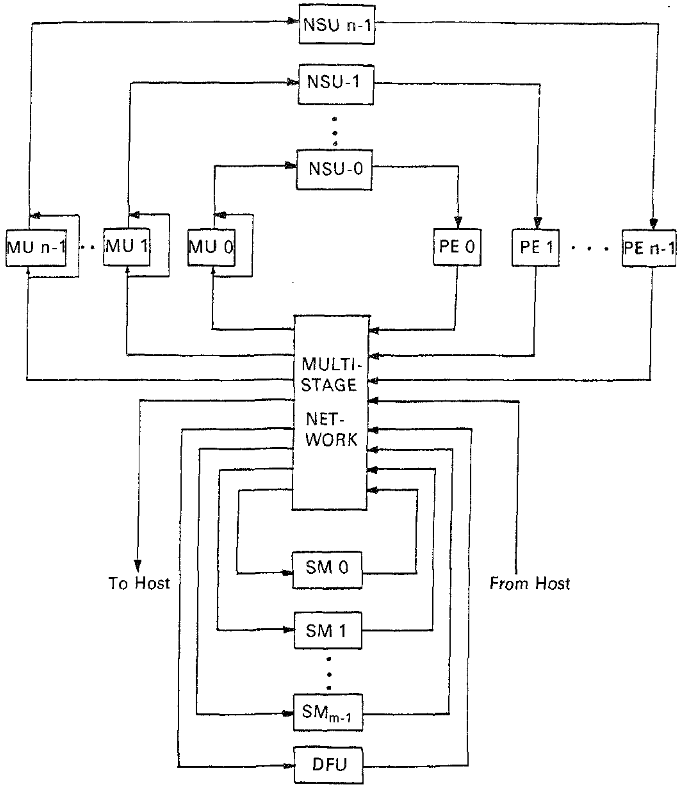
In the multi-ring architecture, we provide a new way of handling deferred read requests arising in the I-structure memory[1] using the matching unit. The reason to do so is two fold. Firstly, it simplifies the design of the memory units. Secondly, the hashing mechanism of the matching unit[6] can be used to support the deferred read mechanism without any extra hardware.

3. Environment management in OR-parallel execution of logic programs

In OR-parallel execution of logic programs, all the solution paths of the search tree are examined independently. When a goal invokes a set of clauses, the goal variables may get bound to different values simultaneously because of the unification with different clause heads. These bindings form the alternative solutions to the goal and have to be maintained independently during the program execution[3,5,7]. Unification with each clause generates new bindings which have to be propagated down the search tree for solving the next level of subgoals. Thus the main issue in OR-parallel execution is the efficient maintenance of these multiple environments.

We propose a novel scheme called the tagged variable scheme for the maintenance of multiple environments for executing logic programs on dataflow machines. Our scheme is based on the sharing of binding environments; therefore, it does not incur the overheads of the copying technique. An environment created in a clause invocation is represented by a 2-tuple ⟨local BE, goal BE⟩. The local BE stores the bindings of the variables present in the head of a clause. The goal BE represents the external bindings generated during unification. When a clause execution is over, the environments are modified suitably to facilitate variable searching. The two distinct advantages of our scheme are:

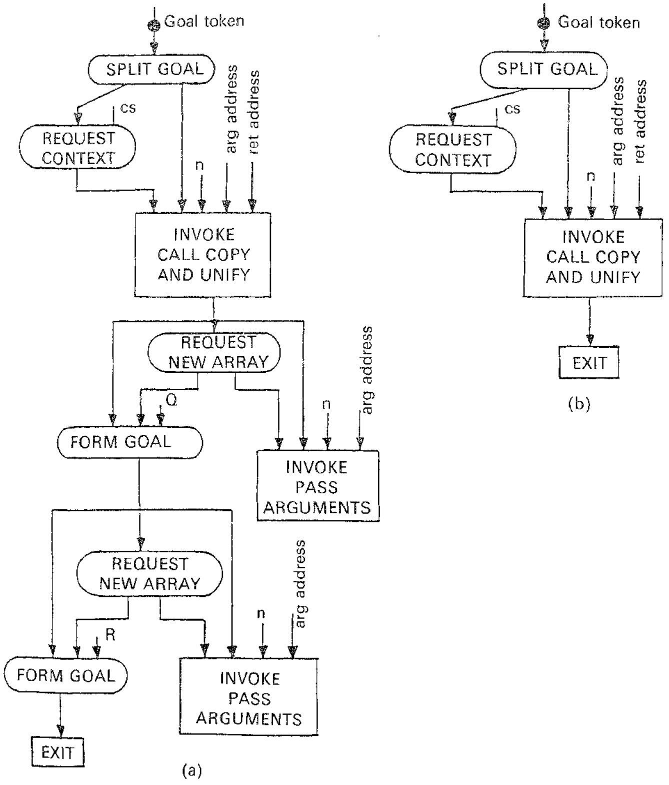
1. Search for a variable need be done only in one context.



NSU: Node store unit; PE: Processing element; SM: Structure memory; DFU: Definition search unit.

FIG. 1. Multi-ring dataflow architecture.

2. No copying of environment is required. Since we represent the binding environment as a 2-tuple ⟨local env, goal BE⟩, the local variable bindings can be extracted from the local variable frame of a context in constant time whereas the goal variable search requires linear time because of the list structure of goal bindings. The basic premise to choose such structure is that the number of goal variables bound in a clause is very small; therefore, the goal BE to be searched is small. We believe



n: number of arguments; arg address: pointer to argument array; ret address: return address; cs: number of variables in the head of a clause.

FIG. 2. The dataflow graph of: (a) a definite clause p(X, Y). – q(X, Z), r(Z, Y), (b) a unit clause p(X, X).

that some sequential search is unavoidable when exploiting OR-parallelism. For example, even in the hash window technique, a non-local variable has to be searched in several hash windows. Moreover, the hash window scheme[2] incurs overheads of initialization of the hash windows which have been avoided in our scheme. Since the memory latency can be hidden in a dataflow machine, and the search for all the variables can be performed concurrently, the overheads of sequential

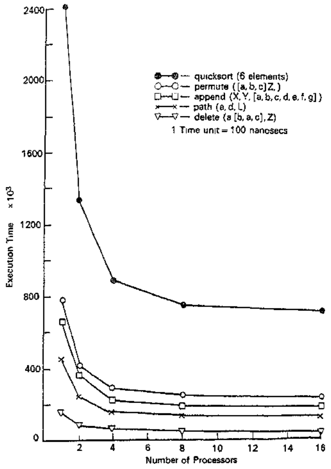


FIG. 3. Execution time *vs* number of processors.

search can be tolerated because there will be several threads of sequential searches which can run concurrently.

The required data structures and the built-in dataflow procedures for OR-parallel execution are discussed. The program clauses are compiled into dataflow graphs. The graph of a unit clause and a definite clause are shown in fig. 2. These graphs are nothing but calls to the built-in procedures; therefore, they are modular and independent of argument complexity. This feature makes the compilation of the clauses very easy.

4. Performance evaluation of the architecture

To study the efficacy of the proposed architecture for OR-parallel execution, a software simulator has been developed in SIMULA 67 and implemented on a DEC 1090 system. The performance of the architecture is studied in terms of the execution time, resource utilization, and speedup efficiency. The speedup curve for the chosen examples is shown in fig. 3. The results obtained do not indicate a near-linear speedup beyond a certain number of processors, say four. The maximum speedup achievable for sixteen processors is 4.5. One possible reason for this is that the problems in the domain of AI are highly unstructured leading to excessive overheads at runtime. However, better speedups are expected for larger problems which have more number of clauses.

5. Dataflow implementation schemes for flat concurrent Prolog (FCP) and FUNLOG

We demonstrate the versatility of the proposed architecture by suggesting a scheme for the implementation of flat concurrent Prolog(FCP), and AND-parallel logic language[8]. FCP is a committed choice nondeterministic language. We define the necessary data structures and dataflow procedures for efficient implementation of FCP. A hypercube implementation of FCP has been given by Shapiro which involves the overheads of variable migration[8] and maintenance of the suspended process list for each read-only variable. In our dataflow implementation, such overheads are avoided.

Another area of research in parallel languages for the new generation machines is the unification of logic and functional languages. The motivation for such a unification is the complementary nature of the two language paradigms. Logic programming languages have good knowledge-based inferencing capabilities whereas the functional programming languages have efficient function evaluation mechanisms. Thus a unified language based on these two paradigms will provide a very powerful programming environment where symbolic and numerical computations can be integrated. A scheme for the implementation of a unified logic and functional language FUNLOG which is based on an extended unification algorithm known as the semantic unification is suggested.

6. Conclusions

The work presented in this thesis is concerned with the design of a multi-ring dataflow architecture for parallel execution of logic programs. The multi-ring dataflow architecture supports OR-parallelism, argument parallelism, and the stream AND-parallelism of logic programs. We propose a novel scheme for handling multiple binding environments arising in the OR-parallel execution of logic programs. Schemes for constraining OR-parallel execution are also discussed. A simulator has been developed in SIMULA 67 on a DEC 1090 system with a view to studying the performance of the architecture for the OR-parallel execution of logic programs. Five sample logic programs have been tested on the simulator.

Implementation of flat concurrent Prolog (FCP) and FUNLOG in the proposed machine are also discussed.

References

1. Arvind, Nikhil, R. S. and Pingali, K. K.	*I-structures—Data structures for parallel computing*, Computation Structures Group Memo 269, Laboratory for Computer Science, MIT, February 1987.
2. Barahona, P. M. C. C. and Gurd, J. R.	Processor allocation in a multi-ring dataflow machine, *J. Parallel Distributed Computing*, 1986, **3**, 305–327.
3. Borgwardt, P.	Parallel Prolog using stack segments on shared memory multiprocessors, *Proc. Inter. Symp. on Logic Programming*, pp. 2–11, 1984, IEEE Computer Society Press.
4. Ciepielewski, A. and Haridi, S.	A formal model for OR-parallel execution of logic programs, *Proc. IFIP 83*, North-Holland.
5. Crammond, J.	A comparative study of unification algorithms for OR-parallel execution of logic languages, *IEEE Trans.*, 1985, **C-34**, 911–917.
6. Gurd, J. R., Watson, I. and Kirkham, C. C.	The Manchester prototype dataflow computer, *Commun. ACM*, 1985, **28**, 34–52.
7. Lindstrom, G.	OR-Parallelism on applicative architectures, *Second Inter. Logic Programming Conf.*, 1984, pp. 159–170.
8. Safra, S., Taylor, S. and Shapiro, E.	*A parallel implementation of flat concurrent Prolog*, Technical Report, Weizmann Institute of Science, 1986.

Thesis Abstract (M.Sc.(Engng))

Design of parallel algorithms for a multiple bus multiprocessor system by A. Sarala.
Research supervisor: V. Rajaraman.
Department: Computer Science and Automation.

1. Introduction

Of late parallel computing has emerged as an important area of research. Many of the commonly used numerical algorithms are being reevaluated to determine their usefulness and viability for parallel computing. Matrix operations play a very important role in scientific computations. In this thesis, we design and analyse five parallel matrix computation algorithms specially suited for a broadcast bus-based multiprocessor system. We consider parallel algorithms for matrix–matrix multiplication, solution of a given system of linear equations, matrix inversion, and solution of a given system of nonlinear equations. The multiprocessor system considered is a homogeneous private memory multicomputer system[1]. It consists of n^2 identical processing elements (PEs) connected by a broadcast bus network in an $n \times n$ configuration with two bus-control units (BCU). This computer is called a multiple bus multiprocessor system (MMS) by us. The suitability of a parallel processing system for a specific application depends primarily upon how well the system architecture corresponds to the structure of the algorithms to be implemented. Our intention in this study is to explore the usefulness of the MMS being developed at the Indian Institute of Science, for the four applications mentioned above. Thus, four MMS algorithms have been developed and analysed for the important applications mentioned above. In addition to this a cost-optimal 3-D VLSI algorithm has been designed for matrix–matrix multiplication.

2. Contributions of the thesis

(i) Parallel algorithms have been developed to execute efficiently on the MMS[2], for the following applications:

(a) *Matrix inversion*: The method used is 'the exchange method'. In this method the given matrix, A, is viewed as the coefficient matrix of a system of linear equations $AX = C$; and at each step a dependent variable is exchanged with an independent variable.
(b) *Solutions of a system of linear algebraic equations*: Gauss Jordan method with partial pivoting is considered so that we do not restrict our attention to positive definite matrices. The given task is subdivided among the PEs as uniformly as possible. This combined with the torous data distribution is shown to result in high PE utilization.
(c) *Solution of a given system of nonlinear algebraic equations*: The solution method used is Newton-Raphson method. This method has a quadratic rate of convergence; that is, the error at each iteration is proportional to the square of the error at the previous iteration. This algorithm with a slight modification is shown to be useful in solving nonlinear circuit equations.

PE utilization has been derived and the algorithms have been shown to be cost-optimal and correct.

(ii) An implementation and execution scheme has been proposed to execute the parallel algorithms on the PEs of the MMS using a dataflow approach.

(iii) A 3-D VLSI systolic mesh-connected structure has been designed to perform matrix–matrix multiplication. There are many 2-D VLSI and linear arrays, both modular and nonmodular, for matrix multiplication. We now present a matrix multiplication algorithm for a 3-D VLSI mesh-connected structure. 3-D VLSI has many advantages over the other VLSI[3,4]. Some of them are: (a) High-packing density, or superlarge integration becomes possible without having a serious problem about power dissipation because the multilayer devices in 3-D integrated circuits can have one set of input/output circuit in common. (b) High-speed performance of 3-D integrated circuits is associated with shorter interconnection delay time and decrease of parasitic capacitance due to silicon on insulator (SOI) structure. (c) Parallel processing is one of the special features of 3-D integrated circuits. Widely different large number of information signals could be transferred from the upper to the bottom layer *via* holes with 1–2 μm diameter as compared with 2-D VLSI.

Although 3-D VLSI with multiple (unlimited) active layers has not been realized so far, some 3-D algorithms have been designed. Since 3-D IC with three active layers has already been fabricated experimentally[5], we can hope to realize our 3-D matrix multiplier. The correctness of this algorithm is demonstrated. The parallel run time has been computed and the algorithm is shown to be cost-optimal.

(iv) It has been established that efficient parallel solution of a given system of nonlinear circuit equations can be achieved on the MMS.

References

1. Siva Ram Murthy, C. and Rajaraman, V.	A microprocessor architecture for solving partial differential equations, *Microprocessing Microprogramming*, 1987, **20**, 113–118.
2 Sarala, A. and Rajaraman, V.	Fast parallel solution of systems of linear equations and systems of nonlinear circuit equations on a multiprocessor system, to appear in the *Proc. Fourth Inter. Conf. on Super Computing.*
3 Rosenberg, L.	Three-dimensional VLSI: A case study, *JACM*, 1983, **30**, 397–416.
4. Akasaka, Y and Nishimura, J	Concept and basic technologies for 3-D IC structure, *IEEE Proc. Inter. Conf. on Electron Devices*, pp. 488–491, 1986.
5. Inoue, Y *et al*	A three-dimensional static RAM, *IEEE*, 1986, **EDL-7**, 327–329.