Chapter 2

COMPARATIVE STUDY OF REAL TIME EXPERT SYSTEMS

2.1 Introduction

Real time expert systems are being implemented in a number of domains like industry, military and space. All real time applications demand execution speed. However, speed alone is not real time. Many more issues are involved and different techniques have been developed to meet them. Some techniques are developed in generic tools for implementing real time expert systems, while some are implemented in specific expert systems. In this chapter, we have attempted a survey of the techniques and tools developed, and some of the real time expert systems implemented.

2.2 Techniques

New data and knowledge representation schemes, efficient algorithms, problem specific inference procedures and architectures, have been proposed to modify expert systems for real time applications.

2.2.1 Data and Knowledge Representation Schemes

Due to the time varying and continuous stream of input data, a suitable representation formalism that supports time operations, temporal reasoning must be available. The data representation scheme must be able to archive time tagged data. Similarly, the knowledge representation scheme must have facilities to represent event based
knowledge and knowledge about temporal relationships. In [12, 64] knowledge representation schemes capable of representing data and knowledge for temporal domains are presented, (details presented in Section 2.3).

2.2.2 Efficient Match Algorithms

The RETE state saving pattern match algorithm is a fast algorithm used in OPS5 [22]. RETE's principle of state saving is based on the concept of temporal redundancy and minimises the number of comparisons of working memory elements. Further, since similar rules compile into similar networks, identical paths on the network can be shared among different rules. But, RETE match algorithm has many disadvantages. The concept of state saving means that a deletion entails the same sequence of operations that occurred during the addition of a working memory element. Hence deletions are expensive. The Beta memories maintained in the RETE are the cartesian product of the two input memories. This beta memory grows exponentially in space resulting in large execution times. YES/RETE is an improvement of RETE which improves network sharing and hence improves network update efficiency [39].

The TREAT algorithm is developed to improve these shortcomings [51, 53]. TREAT is based on Mc Dermott's conjecture that retesting costs in production system match will be less than the cost of maintaining the state. TREAT is a non state saving algorithm and uses the concept of conflict set support. The non state saving nature of TREAT makes deletions cheaper but addition could be costly. It is empirically demonstrated that TREAT consistently out performs RETE. Miranker et.al. introduced match code compilation to enhance the processing speed of production systems. The TREAT compiler compiles production systems and generates C code that performs the TREAT match [52]. Recent results reported by Miranker show that a compiled version of RETE match out performs a compiled version of TREAT [65]. The discussion about RETE and TREAT is essentially about constants and both algorithms are exponential in time and space. LEAPS is a lazy match algorithm with linear worst case complexity [54]. LEAPS is based on the observation that RETE and TREAT perform eager computing of rule instantiations but fire only
observation that RETE and TREAT perform eager computing of rule instantiations but fire only one instantiation in a cycle. The lazy match algorithm folds the selection strategy into a search for instantiations such that only one instantiation is computed per inference cycle. Uni-RETE is a specialisation of the RETE for unique attribute representations. Unique attribute representation is introduced to eliminate combinatorics from match without losing production system functionality [87]. Unique attribute representation implies a single token per beta memory. This makes Uni-RETE match linear in both time and space. A speed up of upto 10 is reported in SOAR production programs using Uni-RETE. Match Box is an incremental matching algorithm. The idea is to precompute a rule's binding space and then have them monitor the working memory for incremental formation of tuple instantiations [66]. Match Box effectively exploits fine grained parallelism. On a massive parallel architecture, Match Box can perform incremental join testing in constant time. If the binding space is much smaller than the tuple space, then Match Box can also be deployed on serial machines. In REX, a new match algorithm based on the properties of real time data and knowledge (details in Chapters 4 and 5) is proposed.

2.2.3 Parallelism for Speedup

Parallelism in match is initially investigated by Gupta in building the Production System Machine (PSM) [27]. The objective of this study is to investigate fine grained parallelism in production systems using RETE match algorithm. Different sources of parallelism like production parallelism, intra-node and inter node parallelism, RHS parallelism are investigated. It is reported that the amount of speed up that can be obtained is limited, due to the small number of affected productions caused by a few number of working memory changes. Another reason cited for the low speed up factors is the large variance in the processing requirements of the affected productions. A bus based architecture is recommended for implementing parallel match algorithms [27].

The applicability of data flow principles to parallel production system match is
Chapter 2. COMPARATIVE STUDY OF REAL TIME EXPERT SYSTEMS

The piling up of tokens on the input arc of the root node. The splitting of root node into multiple nodes is proposed to avoid this congestion. Multiple root nodes are based on the principle that an\textsuperscript{-}attribute working memory element cannot match a m\textsuperscript{-}attribute condition element if m > n.

The improvements in parallel hardware resulted in efficient message passing computers. A match algorithm for message passing computers is developed at CMU. It is estimated that 98% of the RETE match time is spent in two input node activation. This algorithm reduces the computation time of two input node activations. The cross product effect in two input node activations is a major hurdle in achieving higher speed ups in parallel RETE match. PESA-1 is a RETE match algorithm that reduces the cross product effect [38], by partitioning the RETE net at the intra node level. DRETE (Distributed RETE) is a match algorithm implemented on a special purpose machine CUPID. DRETE reduces the cross product effect by a different method. A separate task is created for each element in the beta memories. A task has three ports, two for the incoming input memory nodes and one for output. CUPID is designed as a match processor attached to a host.

The DADO project at Columbia University investigates parallelism on a tree structured machine. This approach is found to provide little speed up because the inherent parallelism in production systems is low and consequently most of the processors are idle. Another reason is that due to the large number of processors, the processors used are simple and have narrow data paths, no cache and low speed which reduces the execution speed. A n\textsuperscript{-}ary tree architecture to exploit coarse grained parallelism in production systems is proposed in the HERMIES robot project [78]. The productions in a program are partitioned into subsets and each production processor receives one such subset. The efficient partitioning of the productions into distinct subsets is the central idea behind this architecture. Some of the partitioning strategies considered in this project are dynamic partitioning, static partitioning, heuristic partitioning and hybrid partitioning. Ing-Ray Chen and Poole present a performance analysis of rule grouping in real time expert systems developed using Activation Framework [7]. A model describing the run time behaviour of expert system reasoning in real time architectures is constructed. Based on a parametric evaluation of this model, they present an optimising rule grouping algorithm with
the Keringhan-Lin heuristic graph partitioning algorithm as its core for k-way partitioning of the rule set. The results of the experimental evaluation show that k-way partitioning is a trade off between processor overheads and rule overheads. A large number of partitions are efficient if there are large number of rules or large number of instructions per rule.

2.2.4 Asynchronous Execution of Inference Engines

Inference Engines are usually sequential processes requiring dedicated processors for their implementation. In real time domains inference procedures must be embedded in the system, and execute along with other processing tasks in the system.

The asynchronous production system architecture is an attempt in this direction. In the APS defined by Sabharawal et. al. for real time expert systems in HERMIES robot [79], the three inference cycle phase - match, select and act are executed asynchronously on three different processors of a shared memory multi processor system. A fourth processor takes external input and changes data in the working memory. Changes in working memory are communicated between the three modules using interrupts. Another variety of asynchronous production systems discount the necessity of synchronisation in the conflict resolution phase of the inference cycle across the different processors. In other words they adopt a fire when ready policy [56, 57]. Synchronous production systems quickly reach saturation speed ups while asynchronous systems can continue to exploit linear speed up under increased work loads.

2.2.5 Multiple Rule Firing Systems

The speedup achieved by parallel execution of production systems is constrained by the number of working memory changes in each inference cycle. The concept of firing a single rule in an inference cycle (single state change) in production systems coupled with few actions per rule results in small number of working memory changes. Thus only small speed up factors are obtained by the parallellisation efforts.
Multiple rule firing systems obtain higher speedup factors by allowing more than one rule to be fired concurrently. This increases the number of working memory changes and hence the speed up. The result of multiple rule firings can be different from the results of any sequential firing of those rules. In this case interference is said to occur among multiple rule firing. In order to guarantee a consistent firing environment for a set of rules, it is necessary that interference among rules be detected and interfering rules be inhibited from firing concurrently. Static compile time and run time analyses of rules and rule instantiations respectively using data dependency graphs are carried out to detect interfering rules. Static analysis requires large storage space, while run time analysis requires high computation times. Most multiple rule firing systems usually make a space/time trade off using a combination of compile time and run time analyses [39].

Ishida proposed a model of parallel execution that performs interference analysis among rule instantiations using a data dependency graph [31, 33]. The compile time analysis detects interference by utilising all information written in the source programs. Run time analysis of rule instantiations produces more accurate data dependency graphs. Consequently interference analysis at run time allows more concurrency than compile time analysis. A rule selection algorithm based on these interference analyses is also proposed. This multiple rule firing increases the degree of concurrency by a factor of 2 to 9. Ishida later extended this parallel production system model into a distributed production system [32].

Schmolze presents an approach to the problem of guaranteeing serializable behaviour in synchronous production systems that fire rules simultaneously [81]. Two approaches to the serialization problem based on examining data dependency graphs are presented. One examines rule instantiations. This approach allows higher concurrency. The algorithm is \( O(n^4) \) complex. A sub optimal \( O(n^2) \) complex algorithm is also presented. The second approach examines rules and hence detects lower degrees of concurrency. Both the approaches are similar to Ishida's solutions though the detection strategy in the graphs is different.

**Concurrent Rule Execution Language (CREL)** is designed to avoid run time overhead by performing extensive compile time analysis [37]. Compile time
analysis based on serializability criteria is used to partition the productions into uniform clusters. Rules in different clusters can be executed concurrently without any further run time checking. It is impossible to determine precisely which rule instantiations within a cluster can be executed concurrently. The problem of run time checking for multiple rule firings within a cluster is a two stage process in CREL. In the first step interference between rule instantiations is detected. This is an $O(n^2)$ problem. To cut the computation in this step, the interference checks are precompiled into C functions and are executed at run time to detect interference. In the second step, the set of instantiations to be fired concurrently are detected. This is again an $O(n^2)$ problem. To reduce computation time at the cost of reduced concurrency a $O(n)$ algorithm is employed. The CREL system is implemented on a Sequent Symmetry shared memory computer.

Kuo and Moldovan describe the implementation of a multiple rule firing system on a hypercube (iPSC/2) [38]. Unlike most other models which deal with only interference problem, this model takes into account the convergence problem also. The convergence problem is concerned with the correctness of the parallel solution. In this model, the original problem is successively divided into smaller problems, one for each context. The convergence problem is addressed at two levels - the context and program level. Each context within the program is analyzed to find out if a conflict resolution step is required within the context. If so the context is called a sequential context or else it is a converging context. Rule instantiations within a converging context can be fired concurrently, while those in the sequential context need to be synchronised. At the program level, the contexts are analyzed for compatibility and reachability. Three different rule firing models are proposed. They are rule dependence model (RDM), single context multiple rule (SCMR) model and multiple context multiple rule (MCMR) model. In RDM only the compatibility problem is addressed without any consideration for contexts. This model is similar to Ishida's multiple rule firing model. The SCMR model extends RDM by considering contexts. In this model rules from a single converging context are allowed to fire concurrently. The MCMR model extends the SCMR models by considering multiple contexts. Rule instantiations from a given context are allowed to fire with instantiations from other contexts, if the context is compatible with and is not present in
the reachable sets of the other contexts being considered. The MCMR models offer the highest degree of concurrency. The number of contexts affects the performance of the models. Hence partitioning of rules into contexts is an important problem to be addressed in these models.

Matsuzawa proposes a parallel execution method for production systems with multiple worlds [49]. A world is defined as the set of working memory instances generated by the last rule firing. Each fired rule instantiation generates one world. A combination of several such worlds is called a virtual world or a merged world. Two worlds generated by rules R1 and R2 can be merged together if different sequences of firing R1 and R2 yield the same results. If the worlds can be merged then the corresponding rules can be fired concurrently. An algorithm is proposed to detect if worlds can be merged by examining rule instantiations and their creation history. This model however does not take into account negated condition elements in the production system.

The multiple rule firing models studied so far, use syntactic knowledge to detect interfering rules and inhibit them from firing concurrently. PARULEL is a parallel rule language which employs user defined meta rules to perform synchronisation in its multiple rule firing model[84]. These meta rules are called 'redaction' rules. Redaction rules eliminate rule instances from the conflict set. Meta rules specify the conditions under which two rule instances conflict and which of the conflicting rule instances need to be deleted from the conflict set. Meta rules can not only deal with data consistency conflicts but can also be used to deal with semantic consistency conflicts. However this concept of 'redaction' leaves the door open for bugs and it is the programmer's responsibility to provide a complete set of 'redaction' rules. ALEXSYS is a financial expert system built using PARULEL.

The concurrent execution of production rules in a database implementation is studied by Raschid et. al.[74]. DBMS serialisability criterion is taken as the correctness criterion for a correct concurrent execution of productions. A protocol based on the two phase locking scheme to ensure serializability is proposed with two extensions to the transaction manager. One extension allows pages of relations to be accessed while the relation itself may be locked. The second extension explores
Chapter 2. COMPARATIVE STUDY OF REAL TIME EXPERT SYSTEMS

the concept of nested sub transactions to allow concurrent execution of productions within a single transaction. A simulation test bed developed to study the rule features and database characteristics that influence the performance of concurrent rule execution. Transactions are used as a means of encapsulation of rules in this simulation study. The positive RHS actions result in database inserts while negative action elements result in deletes to database. Experiments show that throughput is inversely proportional to the number of inserts and is proportional to the number of deletes. A more detailed summary of the results is given in [14].

The above rule firing models obtain speed up through the inherent concurrency in the reasoning model ignoring application parallelism in the quest for speed up. SPAM - a production system architecture for computer vision applications is an example of a system achieving speed up by exploiting application parallelism coupled with production system parallelism [29]. The performance results of SPAM point to a possibility of obtaining linear speed ups (a speed up of 12 is obtained with 14 processors) by coupling task parallelism with production parallelism. Swarm is a non deterministic parallel language [39], that provides formal methods for specifying production problems, for coding production systems and for verifying the correctness of coded programs. The Swarm execution cycle is as follows

- non deterministically choose a transaction from the transaction space.
- match sub transactions simultaneously against the tuple space
- execute simultaneously all sub transactions that evaluate to true and make changes to the tuple and transaction spaces.

Niemann [56, 57] proposes a parallel production system that provides appropriate language mechanisms to design serializable concurrent production programs rather than providing a guarantee of serializability. A simple locking scheme for working memory coupled with appropriate language mechanism ensures design of serializable programs. These concepts are implemented in UMass Parallel OPS5 which incorporates parallelism at the rule, act and match levels. This also focuses on aspects of control in parallel rule firing systems. Three different control strategies
Chapter 2. COMPARATIVE STUDY OF REAL TIME EXPERT SYSTEMS

serializable programs. These concepts are implemented in UMass Parallel OPS5 which incorporates parallelism at the rule, act and match levels. This also focuses on aspects of control in parallel rule firing systems. Three different control strategies are proposed viz. dynamic scheduling of rule actions to make effective use of processors, heuristic control and algorithmic control [58]. In REX, a polynomial time deadlock free interference analysis technique is proposed and implemented (details in Chapter 6).

2.2.6 Distributed Problem Solving Systems

The computational demands and the amount of knowledge typically exceed the capabilities of a single intelligent agent. Cooperative distributed problem solving architectures are being proposed to handle such computational requirements [23, 3, 26].

2.2.7 Problem Specific Inference Procedures

Problem specific inference procedures [34, 61, 62] and latest techniques like neural networks [83] are being developed to meet the specific needs of application domains. EXTASE [34] is such a situation assessment system. Inference in EXTASE is performed by traversing the plant structural graph and creating a hypothesis graph which corresponds to the current system configuration and state. The inference algorithm is explicitly designed to suit a particular situation and is not a generic facility.

2.3 Tools

There are different tools which implement the above techniques to facilitate development of real time expert systems. A survey of the different tools is presented below. In Table 2.1, a summary of the features (discussed in Chapter 1) of the different tools is presented.
LES did not support temporal reasoning and hence diagnosis is performed using current telemetry data without reference to historical data. A later version of LES provides temporal support facilities like handling frame slot values as a function of time. These values can be referred to in the rule consequents, external procedures and interactive updates to frame base. One major shortcoming in LES is the fixed amount of storage for temporal data. The knowledge engineer needs to define \textit{a priori} the number of data values to be stored for a particular slot. In continuous monitoring systems, it is impossible to decide \textit{a priori} the amount of history to be considered during inference.

Active databases is another area where constraints in real time applications are being studied. HiPAC is an active database project that seeks to perform time constrained management \cite{12}. This project addresses two critical areas in active databases: the handling of time constraints and the avoidance of wasteful polling by use of situation - action rules as an integral part of DBMS condition monitor. A rich knowledge model in HiPAC makes it possible to define timing constraints, situation - action rules and precipitating events. This model stands in contrast to models used in real time knowledge based systems.

\textbf{Procedural Reasoning System (PRS)} diagnoses malfunctions in process control applications in real time \cite{30}. A PRS agent consists of a database of the system's current beliefs about the world, a set of current goals and a library of plans and procedures that describe actions and tests required for achieving these goals. The PRS inference engine performs reasoning and planning by manipulating this database. At any time goals are established and events occur that change the beliefs in the database. These changes triggers Knowledge Areas (A plan is represented as a graphic network called Knowledge Area). One or more of the triggered KA's are selected and placed on the intention structure (agenda). The inference engine selects one of the KA's and executes one step of the plan represented by the KA. This results in the establishment of new goals and events in the database and completion of an inference cycle. PRS does not guarantee any reactive or response times. It is the responsibility of the programmer to program KA's to ensure any reactive or response time requirements. PRS has been applied in real time applications like mobile robot control, air traffic management, handling malfunctions
in the reaction control systems of NASA’s space shuttle and diagnosing control failures in a telecommunication network.

Bounded response time is an important consideration in applying rule based systems to real time applications. **MRL** is a real time production system language designed to facilitate accurate analysis of response times while maintaining the flexibility and expressiveness of traditional production languages like OPS5 [90]. MRL uses Rhyme match algorithm. Rhyme is a hybrid of RETE, TREAT and Match Box algorithms. Two expert systems viz. Orbital manoeuvring and reaction control systems valve and switch calculation expert system (OMS) and space station integrated situation assessment system (ISA) are coded in MRL and analyzed for bounded response times [91].

**PAMELA (Pattern Matching Expert System Language)** is developed for coding real time expert systems [1]. PAMELA implemented enhancements to the RETE match algorithm to improve processing speed. PAMELA offers interrupt handling facilities that are so essential to tackle real time problems. It handles interrupts at the end of right hand sides, after specific working memory element modifications and at specific points requested by the user. PAMELA is implemented in a language called CHILL and later ported to C. A parallel version of PAMELA is also implemented.

OPS5 does not provide facilities for temporal reasoning necessary for an real time expert system. So a restricted time map manager (TMM) is coupled with an OPS5 like inference engine [6]. TMM is a point based system and manages only the imprecise or unknown future. All past and known times are represented on a blackboard. TMM is used to perform temporal tests and reason about event sequences.

**Asynchronous Production Systems (APS)** is a result of a innovative execution strategy and shell architecture for rapid integration of asynchronous data in production systems, and the provision of real time response to the same data [79]. The APS architecture is similar to forward reasoning rule based systems like OPS5. The APS inference engine comprises of three asynchronous concurrent modules viz.
match, select and execute. A fourth concurrent module is used to gather external input data. Unlike traditional production systems, APS is able to accept sensory input and integrate it in the inference process. APS requires a MIMD machine with a shared memory, shared bus interrupt driven architecture. Expert systems for the HERMIES robot are implemented using APS. APS can continuously monitor input data but does not provide facilities for temporal reasoning.

**Concurrent OPS5 (CROPS5)** is a production system language designed for real time applications [63]. CROPS5 applications coexist with other real time tasks on the same computing platform. It is specifically designed to be preemptible and priority driven. In contrast to the single data stream of OPS5, CROPS5 has multiple concurrent prioritised data streams with disjoint sets of productions. CROPS5 reduces the granularity of interruptability from match processing boundary to token processing boundary. This helps in reducing execution time variance. An aircraft collision avoidance system is implemented using CR0PS5. CR0PS5 however does not provide any facilities for temporal knowledge representation and reasoning.

G2 is a commercially available object oriented real time expert system [13]. It offers an object oriented data definition facility for structured definition of system configuration. It offers an IF - THEN type rule definition scheme. A good window oriented developer's interface is available. A data interface facility (called GSI) is available to connect the expert system to the external world. G2 provides both forward and backward chaining. Due to it's ability of revising previous decision based on new information, G2 has real time capabilities.

**CAGE** and **POLIGON** are concurrent problem solving frameworks [60] are developed for a class of applications involving real time interpretation of continuous streams of inaccurate data using diverse sources of knowledge. CAGE uses concurrency at the knowledge source level and its target architecture is a shared memory multiprocessor. POLIGON studies concurrency at the blackboard node level and its target architecture is a distributed memory multiprocessor system. These problem solving architectures are appropriate for applications with large data parallelism.

**GEST** (General Expert System Tool) is a blackboard tool developed by GTRI
(Georgia Technical Research Institute) [76] which facilitates development of cooperating expert systems. GEST supports both forward and backward chaining. It further provides a blackboard architecture. It is used to develop Pilot aid's for real time decision making.

**BLOBS** is an object oriented blackboard system framework for reasoning in time [96]. BLOBS is designed for applications which require processing continuous streams of input data and reasoning about geometrical and temporal information. BLOBS attempts to amalgamate the best features in object oriented and blackboard systems. MXA is another blackboard system shell for real time applications [86]. It is developed for tactical picture compilation i.e. producing a representation of the environment surrounding a naval task group. Tactical pictures need to be updated every few seconds using sensor information from diverse sources like radar, ESM and intelligence. This task and its associated processing should be carried out in real time. MXA includes both event driven and goal driven reasoning. A blackboard is used as an internal means of communication between the rules and records the hypothesis reached so far and the way different hypotheses interconnect.

**MUSE** is a tool kit for embedded real time AI [75]. The typical intended applications are operator assistance, monitoring and fault diagnosis in complex electronic and mechanical systems. The basic structure of a MUSE application is a set of separate reasoning modules which communicate by means of shared access to particular databases. The other important features of MUSE are multiple representation languages (production rules, deductive rules, procedures and frames), object oriented approach and knowledge base compilation.

**HCVM** is a computational model of reflective control of resource bounded problem solving [19]. The key features of HCVM include interruptability, event driven adaptive control, modularity, information sharing and multitasking control. Schemer is used to build a number of real time applications like diagnosis of malfunctioning processes, automated performance management of advanced avionics systems and heuristic control virtual machine among many others.
RT-1 is a small scale coarse grained distributed architecture for real time applications based on the blackboard paradigm [16]. RT-1 focuses on four categories of software innovations to improve real time performance. They are control reasoning, focus of attention, parallelism and algorithm efficacy. RT-1 is implemented on a Symbolics Lisp machine using Flavors and Common Lisp.

Generic Black Board (GBB) provides an efficient pattern matching and retrieval functions. Work in GBB is directed towards improving performance of complex blackboard transactions. Efficiency of knowledge source execution and control issues are not addressed in GBB and are left to the application developer [16].

HOPES (Hierarchically Organised Parallel Expert System) also proposed for real time applications [10], is structured into knowledge sources organised in a hierarchical fashion. A multi level blackboard is introduced as a communication mechanism between different knowledge sources. The hierarchical structuring of knowledge sources allows HOPES to handle complex hierarchically structured applications like continuous signal interpretation. HOPES is implemented in an object oriented fashion. A radar signal processing and interpretation application is implemented in HOPES. Other important features of HOPES are uncertainty management and time management.

Erasmus is a configurable blackboard development system [2]. The main objective in Erasmus development is reconfigurability, and certain performance compromises are made. The chief feature of the knowledge source structures in Erasmus are interruptability, asynchronous knowledge source execution, phases, preconditions and obviation conditions. Erasmus is used to develop a cockpit information management application [35].

Activation Framework (AF) supports implementation of object oriented real time distributed cooperative problem solving on interconnected computers [26]. It is based on the paradigm of expert object communicating by messages in a community of experts. The issues addressed in AF development are concurrency, uncertainty about future events, resource limitations, reasoning about time, focus of attention, timeliness, modularity and distributability. It is implemented in C language. AF
timeliness, modularity and distributability. It is implemented in C language. AF belongs to a separate genre of problems solving systems that are classified as Distributed problem solving systems or Distributed AI (DAI) systems. AGORA [3], MACE [23] are other similar DAI frameworks.

In recent past, neural net based connectionist expert systems are being proposed as a means of achieving higher speedups. Sohn and Gaudiot present a three layer net architecture to model the inference cycle of production systems [83]. The neurons are connected in a ring structure and feed forward to follow the logical model of the inference. Hybrid Symbolic/Connectionist architecture (HSC-PS) is proposed to achieve parallel execution of rules [80]. HSC-PS is based on the network structure of the connectionist expert system tool SC-net. The distinctive feature of HSC-PS is its ability to implement variable binding which is not possible in other net based expert systems. Distributed Connectionist Production System (DCPS) consists of five spaces: two clause spaces, WM space, rule space and bind space [83]. Clause spaces are responsible for matching WME’s. WME’s and rules are represented as patterns of activity of neurons or connection weights between the cells. There is no conflict resolution in DCPS. In the current DCPS model a successful match takes nearly 90 seconds and the number of rules is limited.

2.4 Systems

Many real time expert systems are implemented and reported in literature. We present eleven such systems.

L*STAR [42] (Lockheed Satellite Telemetry Analysis in Real Time) is a monitoring system built to aid the HST (Hubble Space Telescope) operator in performing real time monitoring and analysis of data from HST. The system is divided into three processes viz. data management process, inference process (the expert system) and the I/O process. The data management process gathers the telemetry data, pre processes it and transmits the same to the inference and I/O processes. The inference process uses this data along with its to generate advice for the operator. The advice is passed to the I/O process which gives it to the operator through a user friendly
<table>
<thead>
<tr>
<th>Tools</th>
<th>Feature Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>LES[64]</td>
<td>Yes</td>
</tr>
<tr>
<td>PRS[30]</td>
<td>Yes</td>
</tr>
<tr>
<td>CROPS5[63]</td>
<td>PS</td>
</tr>
<tr>
<td>PAMELA[1]</td>
<td>No</td>
</tr>
<tr>
<td>MRL[90]</td>
<td>No</td>
</tr>
<tr>
<td>APS[79]</td>
<td>No</td>
</tr>
<tr>
<td>GEST[79]</td>
<td>Yes</td>
</tr>
<tr>
<td>BLOBS[96]</td>
<td>Yes</td>
</tr>
<tr>
<td>MXA [86]</td>
<td>No</td>
</tr>
<tr>
<td>MUSE[75]</td>
<td>No</td>
</tr>
<tr>
<td>CAGE[60]</td>
<td>Yes</td>
</tr>
<tr>
<td>POLIGON[60]</td>
<td>Yes</td>
</tr>
<tr>
<td>HOPES[10]</td>
<td>Yes</td>
</tr>
<tr>
<td>HCVM[19]</td>
<td>Yes</td>
</tr>
<tr>
<td>RT-1[16]</td>
<td>Yes</td>
</tr>
<tr>
<td>ERASMUS[2]</td>
<td>Yes</td>
</tr>
<tr>
<td>G2[13]</td>
<td>Yes</td>
</tr>
<tr>
<td>REX (proposed tool)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

PS: Partial Support

Table 2.1: Comparison of features of Real Time Expert System Tools

<table>
<thead>
<tr>
<th>Feature No.</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>continuous reasoning</td>
</tr>
<tr>
<td>2</td>
<td>reactive response behaviour</td>
</tr>
<tr>
<td>3</td>
<td>represent and reason about events and their temporal relationships</td>
</tr>
<tr>
<td>4</td>
<td>Focus of attention</td>
</tr>
<tr>
<td>5</td>
<td>Async. arrival of events</td>
</tr>
<tr>
<td>6</td>
<td>Model delayed feedbacks</td>
</tr>
<tr>
<td>7</td>
<td>Handle Interrupts</td>
</tr>
<tr>
<td>8</td>
<td>Embedded operation</td>
</tr>
<tr>
<td>9</td>
<td>Temporal rep. of data</td>
</tr>
<tr>
<td>10</td>
<td>Handling multiple events</td>
</tr>
<tr>
<td>11</td>
<td>Guaranteed RT response</td>
</tr>
</tbody>
</table>
on the same machine, the processes are implemented on three different machines. The inference process uses forward chaining and is coded in C. The facts are represented as attribute\object\value triples. Archival of data based on time stamping is provided.

Wheels [64] involves monitoring and diagnosing problems of the Hubble Space Telescope's four reaction wheel assemblies. To assess the wheels operating status and its health telemetry data is available. Wheels is developed using LES (Lockheed Expert System shell). Wheels uses forward inference to provide control, while backward inference is used to perform diagnosis.

QES [82] is a real time expert system for quality control of steel products in a mill. QES’s knowledge base is categorised intro structural knowledge and behavioural knowledge. Structured knowledge is an object oriented representation of processing steps, connections, steel products variables and defects. The behavioural knowledge includes predictions and diagnosis. It is represented as rules. These rules are further classified into

- inspection rules which confirm or deny the predictions made by prediction rules and
- feed-forward and feed-backward rules that send suggestions up and down the processing line.

A prototype of QES is implemented in Lisp and G2 standard interface provides the process interface. Since steel mill operations are of long duration and do not change often. Hence, the real time requirements are not as stringent as in applications like telemetry data analysis.

REACTOR [55] assists nuclear reactor operators in the diagnosis of accidents. It monitors a nuclear reactor facility, detects deviations, determines the seriousness and recommends an appropriate response. The knowledge about diagnosis is called event oriented knowledge and is represented as production rules. The reactor configuration is called function oriented knowledge and is used to aid the diagnosis process.
ESCORT (Expert System for Complex Operations in Real Time) is a model based expert system that deals with the problem of cognitive overload experienced by operators in process plants [62]. ESCORT diagnoses the underlying problem that caused the alarm using process models (both structural and functional) and causal reasoning. It is implemented in KEE on a Xerox 1186. It is reported that ESCORT deals with around 500 analog and 2500 digital signals and can provide advice within one second of an alarm occurrence. EXTASE is a situation assessment system [34]. It finds cause of alarm signals occurring on the heating furnace of a distillation column. Extase’s knowledge base is designed to explain unit functioning based on known relationships. A causal graph (C-graph) is used to represent permanent process information. To diagnose it builds and validates a search space of hypothesis (a H-graph). The H-graph contains temporary information related to the problem and represents the current status of the reasoning process.

IPCS (Intelligent Process Control System) uses hierarchial fault propagation models, structural and functional models to perform control, monitoring and fault diagnosis in a process control system [61]. IPCS is used to build a diagnostic system for a cogenerator plant in Japan. The IPCS reasoning algorithm consists of two parts viz. the fault component identification algorithm and the inter level migration process. The fault component identification algorithm is responsible for identifying a faulty component in the context of a process fault model. The inter level migration process migrates the fault identification algorithm to lower levels of the fault hierarchy and refines the fault diagnosis. IPCS accurately diagnoses single fault occurrences, but fails in accurate diagnosis of non interacting and interacting multiple faults. It is prone to diagnose such faults as single independent faults.

Integrated Operator Advisor (OA) is a prototype knowledge based system built for the operation and safety maintenance of nuclear power plants [5]. Safety threats are organised in a safety function hierarchy. Each node in the hierarchy represents a safety goal that is lost if the corresponding safety function cannot be maintained. OA detects safety threats by monitoring the threat identifying conditions stored in the nodes of the hierarchy. If a match between the conditions and the current sensor values is obtained the corresponding safety threat is established. OA stores safety and operational procedures in an integrated procedure hierarchy and
chooses a relevant safety procedure from this hierarchy for execution. It does not provide any general purpose temporal representation facilities and these need to be explicitly compiled into the procedures. OA is tested on a set of realistic simulated scenarios and a comprehensive testing is being undertaken.

Mimic [17] is another model based approach to fault diagnosis. It maintains a set of candidate models since a given behaviour might be caused by one of the several faults. Each model represents a possible condition of the system including its state and faults. Two tasks in Mimic maintain the model. One is the tracking task, which keeps the model in step with the observations of the physical systems. If a discrepancy between the observed behaviour and the model occurs, the second task called the diagnosis task is invoked. The diagnosis task detects the fault and injects the same in the model, so that predictions will continue to be in step with the observations. Diagnosis in Mimic is assisted by semi-quantitative simulation. Semi-quantitative simulation uses both qualitative and numerical simulation (a combination of QSim and a numerical model). Mimic refutes or confirms a hypothesis by tracking the process system. If it cannot refute/confirm the hypothesis, the tracking set keeps growing. In practical terms it requires an adequate number of well placed sensors. This could be a difficult proposition in large process control systems. If a catastrophe or an unusual combination of faults occur the simulation model will be at loss and cannot respond in real time. Like the earlier model based systems, this approach is ideal for slowly evolving systems but not systems with a fast dynamic response.

REACT (Rapid Expert Assessment to Counter Threats) is developed to aid pilots in determining appropriate threat response strategies during combat situations [76]. It is also being used to monitor the state of aircraft's on board systems in order to off load the pilot. REACT is built around the paradigm of cooperating expert systems in a blackboard environment. Knowledge is organised as one or more frames. Each knowledge source has an associated fact and rule bases. The rules operate on the frames. REACT is developed on GEST (General Expert System Tool) developed by GTRI (Georgia Technical Research Institute).
Chapter 2. COMPARATIVE STUDY OF REAL TIME EXPERT SYSTEMS

TRICERO is a signal understanding system in the area of air defense. It consists of three expert systems (ELINT, COMINT and ELINT - COMINT correlation) [93]. The three expert systems in tandem perform analysis and correlation of signals. A blackboard architecture is selected as the means of structuring the solution by the three expert systems. In addition to dealing with the issues of building distributed cooperating expert systems, TRICERO addresses issues like control in blackboards, knowledge application strategies, strategies for dealing with time and methods for dealing with stale data and conclusions.

2.5 Summary

In this chapter, a survey of the different techniques, tools and systems is presented. Though all the techniques and tools surveyed are not necessarily about real time expert systems, they touch upon related aspects of knowledge representation, speed up and problem solving organisations. It can be observed that no system meets all the criteria defined for a real time expert system. Each system focuses on a specific subset of the requirements which are salient to the domain under investigation. In REX (our proposed tool), we provide essential features of real time expert systems(refer Table 2.1). REX is designed to be a continuous reasoning reactive real time expert system. It's characteristic features are its data and knowledge representation scheme capable of handling temporal properties, its polynomial time match algorithm and asynchronous multiple rule firing model. In the following chapters, we present the detailed design of REX.