1987

An Expert System Controlling an Adaptable Distributed Data Base System

Bharat Bhargava
Purdue University, bb@cs.purdue.edu

John Riedl

Detlef M. Weber

Report Number:
87-693

AN EXPERT SYSTEM CONTROLLING AN ADAPTABLE DISTRIBUTED DATA BASE SYSTEM

Bharat Bhargava
John Riedl
Detlef M. Weber

CSD-TR-693
June 1987

DRAFT

AN EXPERT SYSTEM CONTROLLING AN ADAPTABLE DISTRIBUTED DATA BASE SYSTEM

by

Dr. Bharat Bhargava

John Riedl

Detlef M. Weber

School of Computer Sciences

Purdue University

West Lafayette, IN 47907

May 1987

ABSTRACT

The task of concurrency control is to synchronize the user transactions on this data base and with this to avoid conflicts that lead to faulty or inconsistent data. Concurrency control has been subject to active research for quite some time and several concurrency control algorithms have been proposed and implemented.

The relative performance of these methods varies depending on the system environment (work load, user performance, reliability requirements etc.). No method showed a superior performance in all or a majority of applications.

A adaptable concurrency control mechanism is needed that provides the optimal behavior in terms of response time and throughput in all or at least many possible applications. This mechanism is able to change the concurrency control algorithm depending on the system environment.

The target of the research that is described in this paper is the selection of the concurrency control algorithm that is most suitable for achieving a high performance in the given system environment.

An expert system is proposed for this function that operates on-line, real-time and without human assistance. A prototype software has been implemented in PROLOG.

1. INTRODUCTION

Concurrency control is a necessary function in a distributed data base system. The task of concurrency control is to synchronize the user transactions on this data base and with this to avoid conflicts that lead to faulty or inconsistent data. The different types of conflicts and their consequences are described in [1].

Concurrency control has been subject to active research for quite some time and several concurrency control algorithms have been proposed and implemented [1], [17], [18], [19], [20], [21], [22], [23], [24]. The relative performance of these methods varies depending on the system environment (work load, user performance, reliability requirements etc.). No method showed a superior performance in all or a majority of applications [1],[2].

These performance differences are significant problems for the design of distributed data base systems. It is not sufficient for flexible and high performance distributed data base systems to implement just one of these concurrency control algorithms. A concurrency control mechanism is needed that provides the optimal behavior in terms of response time and throughput in all or at least many possible applications. This mechanism has to be able to adapt to changing system environments by changing the concurrency control algorithms when appropriate.

The adaptability of the concurrency control mechanism requires the distributed data base system to be able to identify the user behavior and system environment, to measure the system performance, to evaluate this information and to adapt to the systems environment by choosing the concurrency control algorithms for operation that is most suitable.

This paper is organized in five major sections. In

Section 2 the problem of adaptability is described and analyzed in detail. The subject of interest for this paper is defined. Section 3 analyzes the control problem and proposes an expert system as a solution. The expert system is outlined in its components. The details of the expert system are discussed in Section 4. The software and its capabilities are described and the format for the knowledge base is defined. Section 5 presents a scenario and shows how the expert system will operate on it. Some tests are described. Finally, in Section 6 our conclusions and suggestions for future work are outlined.

2. CONCURRENCY CONTROL AND ADAPTABILITY

2.A. CONCURRENCY CONTROL

The concurrency control problem has been well formalized by [4]. It can be shown that all concurrency control algorithms are variations of three basic techniques [4],[5], which are DSR (acyclic cycle graph), 2PL (two phase locking) and SSR (time stamps).

Figure 1 [4] displays the three classes of concurrency control algorithms in the space of legal and serializable histories. Each rectangle represents the set of histories that is accepted by a particular class of concurrency.

The different areas that are covered by these rectangles give a taste of the different synchronizing capabilities.

Several parameters were identified in [1], [2] and [3] that can be used to describe the performance of a particular concurrency control algorithm in a given system environment. These parameters and their informational power will be discussed in Section 5.

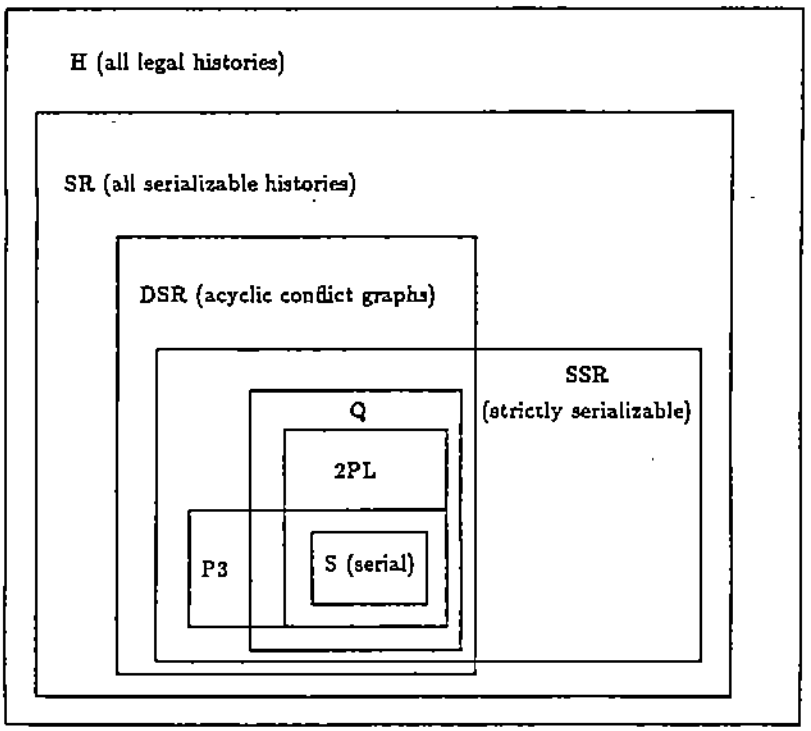

Figure 1 Classes of Concurrency

2.B. ADAPTATION

The problem of adaptation consists of three sub-problems. The first aspect is the measurement and identification of the current system performance state. The information for the adaptation decision making is gathered and evaluated. The second aspect is the test for a concurrency control algorithm that would outperform the currently running algorithm. Finally, the adaptation strategy to replace the current concurrency control algorithm by this other algorithm is of interest.

In the adaptable distributed data base system each site is responsible for gathering locally the values for the parameters that are used to describe the performance state. The data is collected and converted into globally representative parameter values that describe the system performance state.

The decision making mechanism then evaluates the global parameter values that are generated and eventually triggers a process that decides about replacing the currently running concurrency control algorithm with another one. The tasks of this mechanism can be described as follows:

- to recognize the system environment changes that might reduce the performance of the distributed data base system,
- to identify the concurrency control algorithm that is most suitable for this system state.
- to evaluate whether the performance problems justify a replacement and eventually
- to trigger the adaptation mechanism.

Figure 2 summarizes these four tasks.

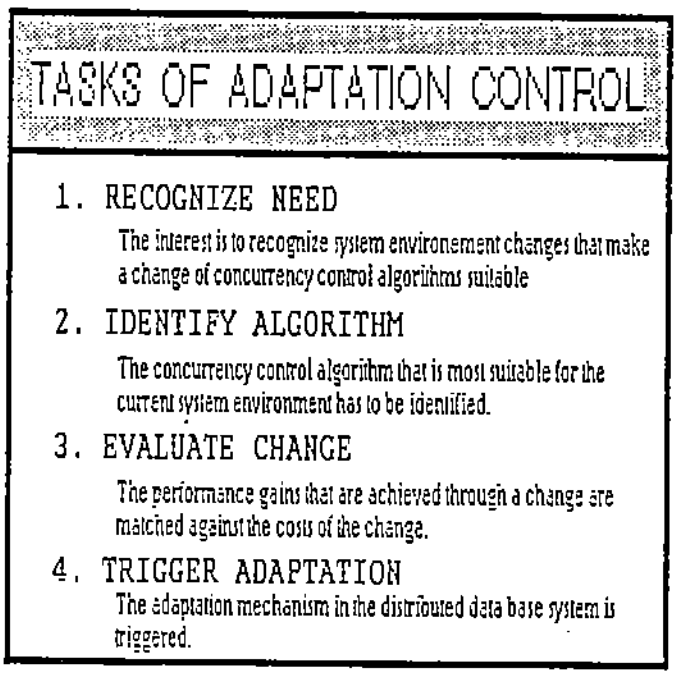

Figure 2 Tasks of the Decision Making Mechanism

Unfortunately, the relationships between the concurrency control algorithms and the parameters that represent the system performance state are difficult to model. Very

little information is available about the magnitude and direction of the performance variation of the algorithms, no method to quantify the performance of the algorithms is defined and the impact of each parameter on the performance of an algorithm is not precisely known.

An expert system appears to be the best strategy to capture the incomplete and unorganized expert knowledge in this domain and to control the adaptation strategy of the system {6}, {7}, {8}, {9}. This expert system would operate on the parameters that are provided by the system and would identify the most suitable algorithm for the given system performance state using deduction rules that describe the known relationships between the concurrency control algorithms and the parameters. Due to the configuration of the distributed data base system this process is executed without human assistance.

The questions that arise are: the frequency of data collection and decision making, the parameters that have to be collected, the hysterese of reaction of adaptation etc. {2}.

2.C. Adaptation Strategy

One way of changing concurrency control methods while the system is operating is to simply stop accepting new transactions, wait until all in-progress transactions are completed, and start accepting transactions again using the new concurrency control method {3}. This solution has a severe drawback: the transactions that arrive at the system during this intermission have to be aborted or have to wait. The distributed data base system is not available. As a consequence the response time will go up and the throughput will decrease significantly for this time. The cost of this temporary

performance reduction will in many cases exceed the performance gains that are achieved through the change of concurrency algorithms.

Another possible strategy of adaptation is to convert the concurrency control information that is used by the currently running algorithm (e. g. locks) into concurrency control information that is required by the replacement algorithm (e. g. time stamps). This requires the implementation of conversion routines. The conversion process still requires an intermission of the transaction processing while the conversion process is running but this intermission appears to be shorter than in the first strategy. The question is how costly this conversion process is.

A third strategy is preferred by [2], [3]. It is proposed to run both, the currently running and the replacement algorithm, for an interlude time. During this time transactions have to be accepted by both, the new and the old algorithm, in order to commit. Transactions that are not accepted by one of the two algorithms have to be aborted. The old algorithm is shut down once the history is satisfactory to the new method. This is the case when all transactions are terminated that were started before the new concurrency controller began operation.

The performance of the distributed data base system will decrease, due to the cost of running two concurrency control algorithms in parallel. It is expected that this decrease will be less significant than for the two other methods. Another problem that needs investigation is to identify the percentage of transactions that are aborted due to running two concurrency control algorithms in parallel.

2.C. SUBJECT OF RESEARCH

The target of the research that is described in this paper is the second aspect, the testing of candidate concurrency control algorithms for their suitability in the current system environment. The other two aspects, the collection of the parameter values and the actual change of the concurrency control algorithm, are discussed in {2} and {3}.

An expert system will be proposed for performing the four tasks that were defined in Subsection 2.B. A prototype software has been implemented in PROLOG.

3. SYSTEM ARCHITECTURE

3.A. THE ADAPTATION CONTROL PROBLEM

The four tasks of the adaptation control module are to recognize unsatisfactory behavior of the distributed data base system, identify the most suitable replacement algorithm, justify the change and to trigger the change (see figure 2).

The parameter 'response time' is a sufficient indicator to recognize unsatisfactory behavior (task 1). Critical values for the response time are specified by the systems designer. The decision making mechanism is triggered whenever the response time exceeds these critical values.

The decision about a change is made based on a heuristic measure that represents the ration between replacement cost and performance gain (task 3). If the decision is to replace the currently running algorithm then simply an identification of this algorithm is passed to the system. There the required routines are loaded and the adaptation process takes place (task 4).

The crucial part of the decision making mechanism is the test of the available set of concurrency control algorithms for their suitability in the current environment (task 2). The suitability value for an algorithm describes its relative applicability for the current system performance state. This test is based on the parameter values that are supplied by the system and the knowledge that was implemented in the knowledge base. The expert system performs the inferences on these parameters and this knowledge and returns the suitability measures for the algorithms.

The specifications for the implementation of the expert system can be described as follows:

- several candidate concurrency control algorithms have to be tested at the same time and on the same parameter values for their suitability,
- the parameter values that are supplied by the distributed data base system describing its current state vary in completeness, correctness, reliability and recency,
- the decision making has to be done in real-time and the results have to be obtained in a minimum of time,
- the decision making has to be executed on-line and without human assistance,
- the behavior of the distributed data base system is dynamic, which means that the system may update the parameter values while the decision making process at the expert system is executing. The expert system has to react to these updates in order to avoid decisions that are already obsolete when issued to the system.

Figure 3 summarizes these specifications.

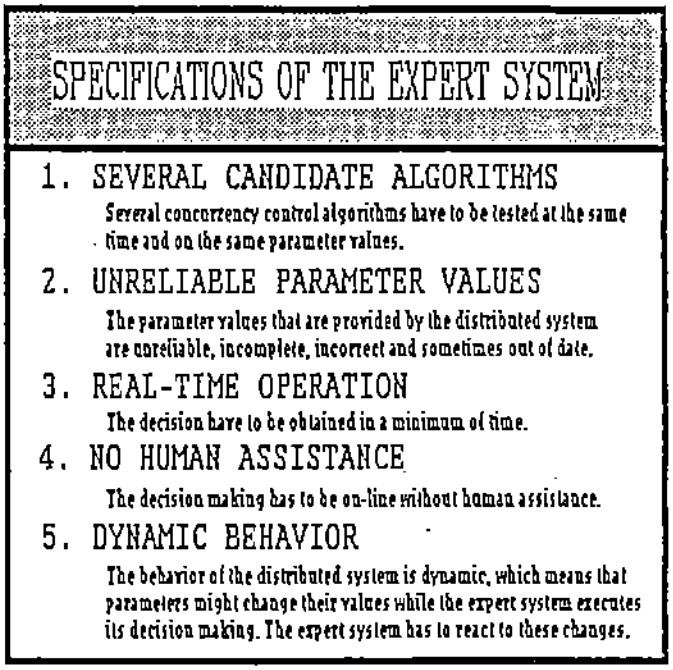

Figure 3 Specifications for the Expert System

3.B. EXPERT SYSTEM ENVIRONMENT

Figure 4 is a schematic diagram showing the components of the expert system and its relationship to the other components of our distributed data base system.

In figure 4 the distributed data base system is represented by three sites. The major components are the concurrency controller with the algorithm pool, the atomicity controller, the access manager, the data base and the parser connecting to the users. The algorithm pool is the location where the routines reside that implement the different concurrency control algorithms.

The system interface connects the distributed data base system to the adaptation control module. This interface has two major functions. First, it collects the local performance states from the sites and generates the parameter values for the global system state.

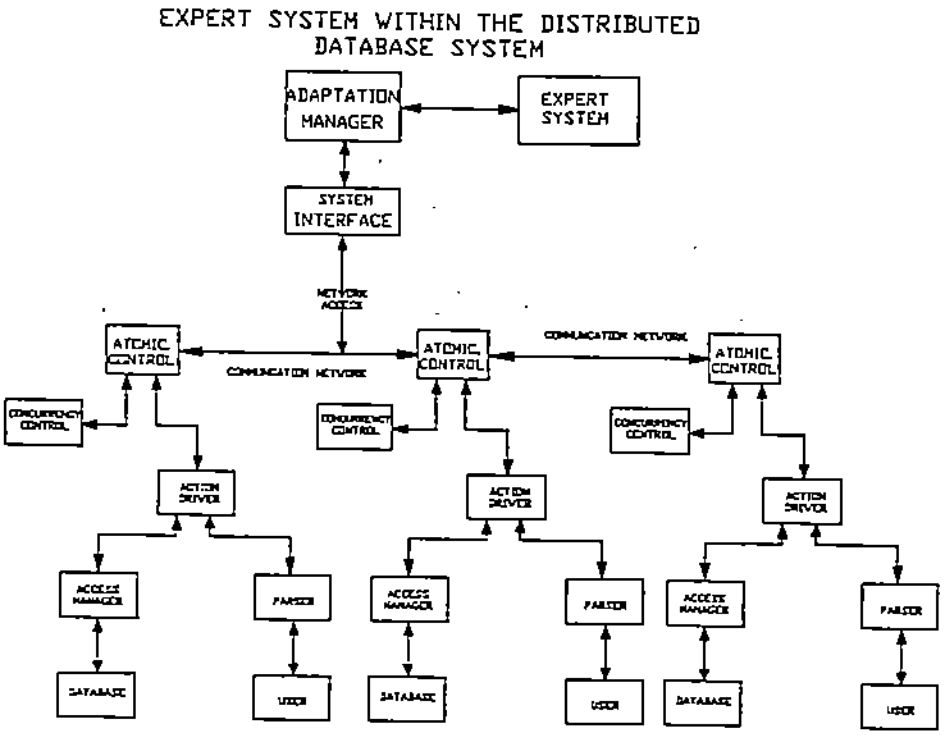

Figure 4 Adaptable Distributed Data Base System

These parameter values are available to the adaptation manager. The other function is to pass replacement decisions to the concurrency controllers of the local sites thus triggering the change of concurrency control algorithms.

The adaptation manager monitors the global performance state as provided by the system interface and decides when to start the expert system for another test for a suitable replacement concurrency control algorithm. During these tests at the expert system the adaptation manager provides the parameter values. After a successful test the adaptation manager evaluates the replacement costs against the gains and decides finally whether to execute the replacement.

The expert system is the location of the suitability test. This test matches the available candidate concurrency control algorithms to the current global

system performance state and returns for each algorithm a suitability measure. This measure describes a prediction how well this algorithm would perform in the current system environment relative to the other algorithms.

3.C. EXPERT SYSTEM CONFIGURATION

The general approach for building the inference engine the expert system can be chosen based on the specifications that were defined in Subsection 3.A. The backward deduction approach tests a single goal for its suitability and returns a measure for this. The approach seems therefore not to be suitable for the given problem that requires to test several candidates in relation to each other at the same time.

The forward deduction approach appears to be the better choice {10}, {11}. A forward deduction system generates a solution to a given problem by expanding a solution tree level by level. On each level the locally optimal path is chosen based on heuristics for continuing the expansion. The operation terminates whenever a goal (solution) is achieved or a termination criterion is reached {11}. Several candidates are processed in parallel.

Figure 5 displays the expert system components in detail. Four functions are distinguished: the knowledge base, the shell, the system link and the user interface.

The knowledge base is the location of deduction rules and declarative knowledge (facts) about the system. The shell (or inference engine) is the mechanism that generates the solution based on parameter inputs and the knowledge. The user interface permits a human operator to interact with the system in order to solve problems that are created through missing knowledge. The system link is used by the adaptation manager to start

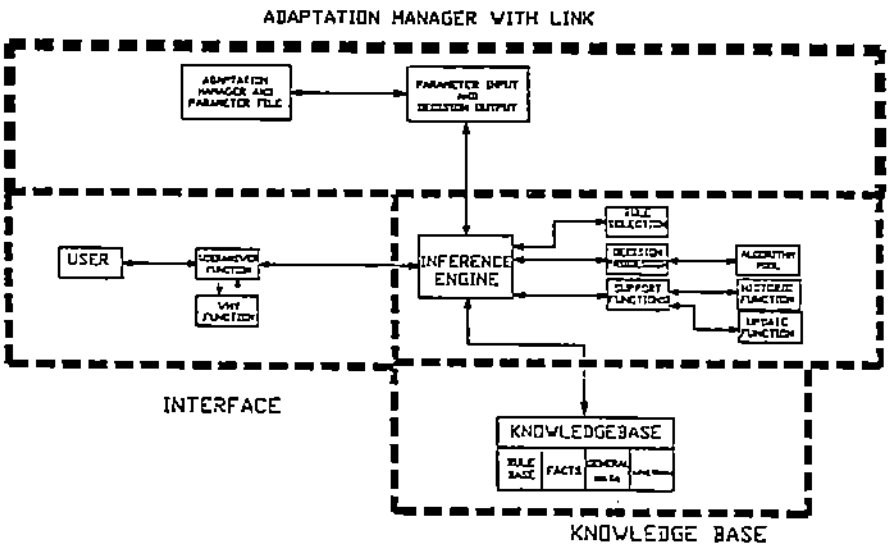

Figure 5 The Expert System

and control the operation of the expert system, to provide the parameter values and finally to extract the results.

In the next Section the implementation of these functions is presented and the topics, dynamic behavior of the expert system and decision making under uncertainty are discussed.

4. THE EXPERT SYSTEM

4.A. EXPERT SYSTEM SHELL

The expert system shell consists of four components: the inference engine, the rule selection mechanism, the decision processor and the support functions. Figure 6 displays the expert system shell.

The task of the inference engine is to match the

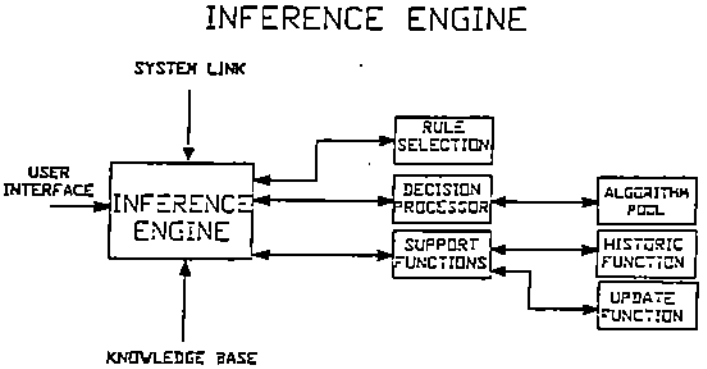

Figure 6 Expert System Shell

current condition set against the conditions of deduction rules in the knowledge base. The last loaded parameter value is used by this process as the variable value. The rules that were successfully matched and unified are collected in the set of applicable rules. Figure 7 displays the match and expansion mechanism.

Some of the rules might have additional conditions besides the conditions that were matched. For these rules an additional consistency check tests whether these conditions violate the decision state. If they are consistent the deduction rule is appended to the set of applicable rules.

The selection mechanism loads the set of applicable rules and tests them on three heuristics in order to identify a locally optimal rule. The optimal rule in this case is defined to be the deduction rule that has the highest impact on the identification of a most suitable replacement concurrency control algorithm.

The heuristics use the recency of the parameter that is processed, the number of the conditions in the deduction rule and the strength of the rule as factors.

The number of the conditions is an indicator for

```
forward(Cond1 and Cond2,[Alg/P|Rest],New1,Depth,Trace,
        [Rulenum,Cond1 and Cond2|Solution]):-
   bagof(Rule:if candidate Alg and Cond1 and Cond2 and Morecondition
                then Action with Strength,
         (Rule:if candidate Alg and Cond1 and Cond2 and Morecondition
                 then Action with Strength,
          test(Morecondition,Cond1 and Cond2,Trace)),
         Ruleset),
   pick(Ruleset,[Rulenum:if candidate Alg and Cond1 and
         Cond2 and Morecondition then Action with Strength],Restrules),
   Depth1 is Depth+1,
   !,
   Trace1 = [Rulenum:if candidate Alg and Cond1 and
         Cond2 and Morecondition then Action with Strength|Trace],
   (update([Alg/P|Rest],Strength,Cond1 and Cond2,New),
    forward(Action,New,New1,Depth1,Trace1,Solution);
    not backtrack_mark(Ii),
    match([Alg/P|Rest],New1,Depth1,Trace,Solution,Restrules)).
```

Figure 7 Match and Expansion Mechanism

the specifity of the rule under consideration. Specifity means that this rule has a higher degree of refinement than another more general rule and therefore should be considered first. The recency of the parameter values describes the usability of these parameters for the decision making, old values are less usable than recent values. Finally the strength of a rule indicates the confidence of the expert and systems designer into the deduction rule under consideration. Figure 8 displays a deduction rule with two match and one concistency condition.

```
rule2 : if
   candidate X
     and
   update_readonly equals Y - Z
     and
   number_of_conflicts equals P - K
     and
   Y is_less_equal 0.5 - 1
        then
   number_of_conflicts request R - V
     and
   update_readonly equals Y - V
     and
   number_of_conflicts equals R - V
        with
   strength(0.8,1.2).
```

Figure 8 Deduction Rule

The deduction rule that was selected by the selection mechanism is passed to the decision processor. Here the rule is applied on the candidate concurrency control

algorithms. The goal of the decision processing is to modify the suitability measure for the different candidate algorithms.

The reliability measure of the parameter, the strength values of the deduction rule, the parameter-algorithm relationship factor and the current suitability measure of the algorithm are the arguments for computing a new suitability measure.

All candidate algorithm whose suitability value fall below a predefined minimum are discarded. The remaining candidate algorithms are sorted depending on the suitability value.

The history supporting function records the result of a decision making operation of the expert system in a history file. This file can be used by experts to evaluate, debug and refine the knowledge base of the expert system in order to improve its performance. The knowledge supporting function records new knowledge that was provided by experts during a decision making run.

4.B. USER INTERFACE

The user interface provides a link to human operator. The Expert system can use the link to prompt the systems operator for missing knowledge and the systems operator can use it for on-line monitoring of the performance of the adaptation module.

During a decision making process the expert system might reach a point from where it cannot proceed because necessary knowledge is missing. It normally would backtrack and restart the decision making process from a lower level. If a systems operator is on-line the expert system can prompt the operator for the missing knowledge. The system operator informs the expert system about his availability by asserting a simple fact that operates as a switch.

The operator can prompt the system for some information by using the 'why' function. The why function basically retrieves the trace of the current decision making process and the suitability values of the candidate algorithms. Based on this information the operator can make decisions and eventually enter new declarative knowledge into the knowledge base. The modification of the deduction rule base has to be done off-line through editing the deduction rules file.

4.C. SYSTEM LINK AND ADAPTATION MANAGER

The system link connects the adaptation manager and the expert system. Whenever the adaptation manager decides that a test for a replacement concurrency control algorithm is needed it sends an initialization message to the expert system.

The expert system starts operating and requests step by step the parameter values from the adaptation manager as it proceeds in its decision making. The adaptation manager returns on each request the requested parameter value and the reliability value for this particular parameter.

After terminating its operation (a candidate algorithm was found or another termination criterion was met) the candidate algorithms and the associated suitability values are returned to the adaptation manager via the system link.

The adaptation manager maintains the functions to monitor the distributed data base system, to decide about when to activate the expert system, to evaluate the gains and costs of a change of algorithms and finally to trigger the change.

The parameter values that represent the system performance state are generated in the system interface and stored in a file that is periodically accessed by

the adaptation manager. The adaptation manager loads these parameter values as they come in and checks for changed values. Depending on the parameter the values may vary in a predefined range.

Whenever the adaptation manager recognizes that the response time goes below a predefined level it starts the expert system. The expert system receives an identification of the currently running algorithm and the value for the response time.

The test of the available concurrency algorithms for suitability at the expert system returns a value per algorithm that describes the suitability measure for this algorithm. The distance between the suitability value for the currently running and the value for the most suitable concurrency control algorithm serves as heuristic measure for the replacement decision. If this distance value is high this indicates a high profit of a change to the replacements algorithm. A small distance value indicates that the costs of the change outweight the gains.

Finally, if the decision was made to replace the currently running concurrency control algorithm with the most suitable replacement algorithm then a message is sent to the concurrency controllers of all sites via the systems interface. The message simply contains the identification of the replacement algorithm.

4.D. DYNAMIC BEHAVIOR

A dynamic behavior of the expert system was defined as a requirement in the specifications in Subsection 3.A. Dynamic behavior of the expert system means that changes in the parameter set during a test run are recognized and that the expert system reacts to these changes by rolling back some of the decisions that are already made.

The dynamic behavior is required in order to avoid that a replacement decision is already obsolete when it is executed in the distributed data base system. An unnecessary change would waste computing time and degrade the short term as well as the long term system performance. The dynamic behavior assures that the change decision is always based on the most recent data and with this achieves a maximum of reliability.

Furthermore, the dynamic behavior acts as a safty check that avoids that the adaptation mechanism reacts to short term changes in the system performance state. These short term changes can be a load peak followed by a return to the previous state or a time period with very many, frequent changes in different parameter values due to different user applications. In both cases it is not suitable to replace the concurrency control algorithm because the cost of the change will exceed the short term gains of this change.

The dynamic behavior is implemented as a check-and-backtrack routine. The expert system maintains a list with all parameter values that were used in the previous steps of a test run. On each step this list is compared with the current parameter list of the adaptation manager. If a difference of sufficient significance is detected the decision making process is stopped and rolled back to the point where the changed parameter was introduced to the decision making. The process is restarted at this point with the new parameter value.

In the case of a short term change of the system performance the expert system would roll back to the initialization of the expert system (From there no restart is executed). The change of the performance state has to be short enough so that the new parameter values arrive before the test in the expert system finishes successfully.

If the system performance state is characterized

by frequent changes during some time period the expert system would perform a back and forth jumping without terminating. Only after reaching a balanced state the expert system would terminate proposing an algorithm for this stable state. This back and forth jumping is desired because it prevents the expert system from making a change decision that was based on an instable systems environment. The distributed data base system is not burdened with the algorithm change effort without gaining much advantage from this change.

4.E. DECISION MAKING UNDER UNCERTAINTY

The decision making in the expert system finding the most suitable algorithm for the concurrency control mechanism is a process that is characterized by uncertainty. The parameter values that describe the system environment are of limited correctness, completeness, reliability and recency and it is questionable whether they describe the system environment is sufficient detail [2], [3]. The deduction rules are used to model relation ships between parameters and algorithms that are not precisely known [1], [2]. Finally the relationships between the parameters and the algorithms are not clearly identified.

The expert system provides the approach to achieve decisions in this environment. The decision (or the decision alternatives) are associated with belief values (here called suitability values) that describe the certainty that the derived solution is the desired one. The suitability values (belief values) are subject to modification in each step of the decision making process. Several approaches were developed like the Dempster-Schafer theory [12], [13], the bayesian theory [14], [15] and the certainty factors in MYCIN [16].

In this work another heuristic model was developed

that is similar to the bayesian model. The modification of the suitability value in one decision making step depends on several factors in our model.

The evidence factor describes the reliability of the parameter value. The system interface defines this value describing reliability, completeness, recency and correctness of the parameter value. The value is returned with the actual parameter value from the adaptation manager when the parameter value is requested.

The deduction rule factor represents the confidence or relevance that is assigned to this deduction rule. The expert that defines a rule attaches this value to this rule.

In our implementation even two deduction rule factors were attached. The one value is used for increasing suitability values and the other for decreasing suitability values. The attachment of two factor values permits a to refine decision making. A particular parameter value might indicate that the suitability for one group of candidate algorithms can be increased significantly. At the same time the suitability decrease for another group of algorithms might be mild. This different strength of modification is defined through using two deduction rule factors.

The relationship factor models the relationship between a particular algorithm and a particular parameter. The suitability of an algorithm is defined to be high if the parameter value is within a predefined range and low if it is in another range. For example a high value for the 'ratio update/read_only' parameter would indicate that the optimistic algorithm is most probably not very suitable. The relationship value for this case would be below 1. The ranges for the parameters in relationship to the algorithms are represented by facts in the knowledge base.

With these three factor and the previous suitabi-

lity value for a candidate concurrency control algorithm the new suitability value for the algorithm is computed. Figure 9 exhibits the arithmetic algorithm for the computation.

```
if REL1 < REL2
    if  T2 > 1
        M = REL2 * (T2 - 1)/(1 - REL1) + (1 - T2 * REL1)/(1 - REL1)
    if  T2 <= 1
        M = T2 + (1 - T2)/REL1 * REL2

if REL1 >= REL2
    if  T1 > 1
        M = REL2 * (T1 - 1)/(1 - REL1) + (1 - T1 * REL1)/(1 - REL1)
    if  T1 <= 1
        M = T1 + (1 - T1)/REL1 * REL2

PROB = SUIT_OLD/(1 - SUIT_OLD)

SUIT_NEW = M * PROB/(1 + PROB)

REL1 = reliability of the parameter     REL2 = relationshipfactor
T1   = first deduction rule factor      T2   = second deduction rule factor
SUIT_OLD = old suitability value        SUIT_NEW = new suitability value
```

Figure 9 Arithmetic Algorithm

4.F. KNOWLEDGE BASE

The knowledge base contains the deduction rules, the facts and general data. Figure 10 displays the knowledge base.

The main location of knowledge are the deduction rules. They are matched and unified by the inference machine during the decision making process.

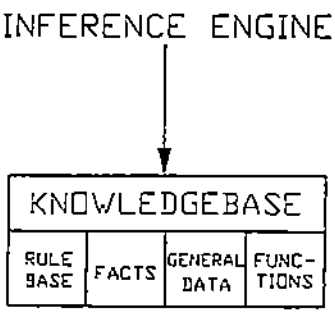

Figure 10 Knowledge Base

Two types of deduction rules can be distinguished. The 'assigned' rules name a concrete concurrency control algorithm as a first condition. This indicates that they are only applicable when this algorithm has the highest suitability value of all candidates. The general rules can be matched by the inference engine independent of the candidate algorithm with the currently highest suitability value.

The fact base contains the mainly the relationship factor values and the general knowledge stores the administrative knowledge of the system.

5. IMPLEMENTATION

5.A. IMPLEMENTATION OVERVIEW

In this Section the implementation details are presented.

The prototype expert system is implemented in PROLOG. PROLOG appears to be a powerful tool for implementing prototype systems in general and of expert systems in particular {25}, {26}, {27}, {28}, {29}.

The built-in inference machine frees the systems developer from all work on the flow control aspect of programming and permits to build the code in modules. The matching and unification mechanism of PROLOG finds the path through the provided code modules. This feature of PROLOG reduces the likelihood of programming errors and makes the PROLOG programs easy to modify and extend.

A scenario was developed and implemented as a subset of the entire control problem. The prototype expert system is tested on this scenario with several test cases.

First, the parameters are defined that are used to describe the system environment. The concurrency control

algorithms are selected and described that will be subject of the suitability tests.

The content of the implemented knowledge base is presented. Some information about the design of their knowledge base are provided.

Finally the expert system is exhibited running on some test cases. The major aspects of these runs are commented.

4.B. PARAMETERS AND ALGORITHMS

Twelve parameters were selected for the description of the current status of the distributed data base system. They are briefly defined below:

1. response time. This is the parameter that describes the performance of the distributed data base system in general and allows to decide whether the concurrency control algorithm has to be changed.
2. transaction conflicts. The relative number of conflicts between transactions occurring in one time unit.
3. roll back cost. The cost of rolling back and restarting a transaction with T/O or optimistic concurrency control after this transaction failed.
4. blocking delay. The average waiting time for transactions in the case of a transaction conflict in a locking scheme.
5. update/readonly ratio. The ratio between readonly and update transactions in the system at one time.
6. multi-programming level. The number of transactions processed in the system at a time.
7. transaction size. The mix of different transaction sizes is described (all short transactions or mix of short and long transactions).
8. arrival rate. The average number of new transactions arriving in the distributed data base.

9. semantics. Knowledge about the distributed data base system, its environment and its performance.
10. communication overhead. This parameter describes the amount of communication activity of the system.
11. CPU and I/O utilization. This parameter describes the current utilization of the local processors.
12. concurrency control overhead. The relative amount of processing work that is required for a concurrency control algorithm.

Figure 11 displays the parameters.

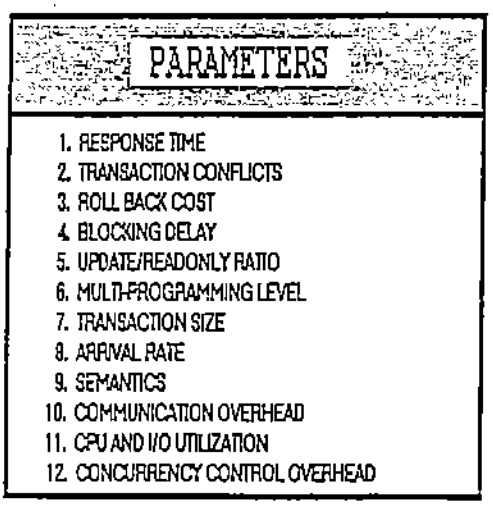

Figure 11 Parameters

Seven concurrency control methods were chosen to be subject to testing in the expert system. Four of these methods belong to the group of two-phase-locking methods (2, 3, 6, 7), two are timestamp-ordering - T/O - methods (4, 5), and one is an optimistic method (1). The seven methods are briefly introduced below:

1. cycle graph detection for read/write and write/write, this method executes all transactions and creates after the execution a cycle graph. It then searches this cycle graph and if conflicts are detected one transaction is

rolled back using a history protocol.

2. centralized 2PL, the locktable is maintained at one central site of the distributed data base system,
3. primary 2PL, primary 2PL is a special form of the centralized 2PL scheme. The locktable for a data item is maintained at one site but different data items have different central sites.
4. basic T/O, timestamp control avoids conflicts between transactions by comparing timestamps. If conflicts are detected one of the conflicting transactions is rolled back.
5. basic T/O for read/write and Tomas Write Rule for write/write, the Tomas write rule permits certain conflicts between two write operations because they do not affect the data consistency.
6. decentralized 2PL, the locktables are maintained at all sites. This means that write transactions have to acquire locks at all sites that hold a copy of the target data item.
7. majority vote locking, this scheme is similar to 6. The difference is that both read and write transaction need to get the locks on the majority of the sites In 6 a read transaction only needs one local lock and the write all locks.

Figure 12 exhibits the concurrency control algorithms.

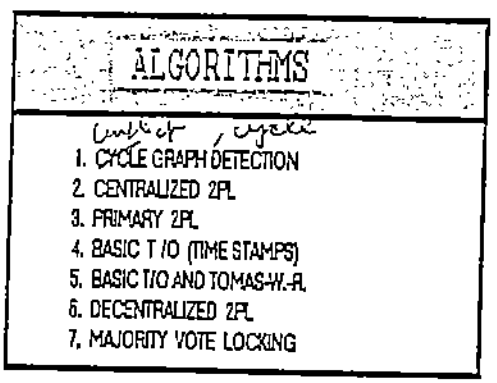

Figure 12 Concurrency Control Algorithms

5.C. CURRENT KNOWLEDGE BASE

The rule base contains at the time of about 50 rules. This prototype rule set is able to solve sufficiently a limited number of test cases. Figure 13 exhibits one deduction rule.

```
rule2 : if
   candidate X
     and
   update_readonly equals Y - Z
     and
   number_of_conflicts equals P - K
     and
   Y is_less_equal 0.5 - 1
        then
   number_of_conflicts request R - V
     and
   update_readonly equals Y - V
     and
   number_of_conflicts equals R - V
        with
   strength(0.8,1.2).
```

Figure 13 Deduction Rule

The fact base contains a fact with the name of the currently running concurrency control method and a set of relationship factors. These relationships are only implemented for three ranges per parameter (value is high, medium or low). Figure 14 displays a subset of these facts.

```
update_readonly(alg1,0.4,0.7).
update_readonly(alg1,0.65,1.2).
update_readonly(alg1,1.0,0.5).
update_readonly(alg2,0.4,1.3).
update_readonly(alg2,0.65,0.5).
update_readonly(alg2,1.0,,0.35).
```

Figure 14 Facts

5.D. EXPERT SYSTEM IN OPERATION

Initially the adaptation manager is in a wait loop waiting for the next parameter set from the system interface. The expert system is not in operation. At some point the system interface receives some new data

from the distributed data base and generates a new parameter set that is passed to the adaptation manager. The parameter set is displayed in figure 15. The system interface was not part of this work and is therefore only simulated with UNIX commands and prepared data by a programmer.

```
parameters([
   [update_readonly,number_of_conflicts,response_time,
           rollback_cost,cc_overhead,number_of_messages,
           semantics,transaction_size,blocking_delay,arrival_rate],
   [0.65,0.3,0.6,0,0.5,0.5,0,0.3,0.6,0.5],
   [0.8,0.85,0.7,0.6,1.0,0.5,0.1,0.7,0.9,0.4],
   [0.9,0.8,0.85,1.0,0.8,0.7,0.9,0.9,0.9,0.85]
]).
```

Figure 15 Parameter Set

The adaptation manager loads this parameter set and detects that the response time has fallen beyond the predefined level (0.65). It is decided to run a test for a more suitable concurrency control algorithm on the expert system and an initialization message is sent to the expert system. Figure 16 displays this message. The currently running concurrency control algorithm is algorithm 1 (optimistic).

```
forward(response_time equals P-Z,
        [alg1/0.5,alg2/0.5,alg3/0.5,alg4/0.5,
        alg5/0.5,alg6/0.5,alg7/0.5],
        New,0,[],Solution),
```

Figure 16 Initialization Message

The expert system starts operation and goes through five inference steps (and loads five parameter values) before is reaches the decision that algorithm 6 (distributed 2PL) is the most suitable with 0.767 suitability. Figure 17 exhibits the trace of this decision making as it is stored by the historic function.

The adaptation module measures the distance between the suitability values of currently running and replacement algorithm. The distance high and a message is issued to the system interface to trigger the change.

```
| ?- shell.
parameters consulted 0 bytes 0 sec.
parameters consulted 560 bytes 0.150007 sec.
parameters consulted 0 bytes 0 sec.
parameters consulted 0 bytes 0 sec.
parameters consulted 0 bytes 0 sec.
parameters consulted 0 bytes 0 sec.

The best algorithm is alg6 with certainty 0.767852

Would you like to see how?
|:y.
The best algorithm is alg6 with certainty 0.767852
from rule36 with condition(s):
    arrival_rate equals 0.7 - 0.6799
and
from rule35 with condition(s):
    rollback_cost equals 0.75 - 0.6
and
from rule34 with condition(s):
    update_readonly equals 0.85 - 0.719999
and
from rule33 with condition(s):
    number_of_messages equals 0.9 - 0.559999
and
from rule32 with condition(s):
    number_of_conflicts equals 0.8 - 0.679999
and
from rule31 with condition(s):
    response_time equals 0.65 - 0.9
```

Figure 17 Trace of one Run

The test is repeated under dynamic conditions. Currently running algorithm and initial parameter set remain the same. During the third step the parameter set is replaced by new data. The new set is displayed in figure 18.

```
parameters([
   [update_readonly,number_of_conflicts,response_time,
           rollback_cost,cc_overhead,number_of_messages,
           semantics,transaction_size,blocking_delay,arrival_rate],
   [0.65,0.7,0.6,0,0.5,0.9,0,0.3,0.9,0.5],
   [0.8,0.85,0.7,0.6,1.0,0.5,0.1,0.7,0.9,0.4],
   [0.9,0.8,0.85,1.0,0.8,0.7,0.9,0.9,0.9,0.85]
]).
```

Figure 18 Second Parameter Set

The expert system backtracks and makes 5 additional inference steps before the algorithm 1 (optimistic) is selected. Figure 19 displays the trace of this run. The result does not lead to a change of parameters because the adaptation manager decides that the distance between currently running and replacement algorithm is too small. The predicted performance gain is not satisfactory to justify the replacement costs.

```
; ?- shell.
parameters consulted 560 bytes 0.13334 sec.
parameters consulted 0 bytes 0 sec.
parameters consulted 0 bytes 0 sec.
parameters consulted 0 bytes 0 sec.
parameters consulted 560 bytes 0.150007 sec.
parameters consulted 0 bytes 0 sec.
parameters consulted 0 bytes 0 sec.
parameters consulted 0 bytes 0 sec.
parameters consulted 0 bytes 0 sec.
parameters consulted 0 bytes 0 sec.

The best algorithm is alg1 with certainty 0.926165

Would you like to see how?
|: y.
The best algorithm is alg1 with certainty 0.926165
from rule10 with condition(s):
    update_readonly equals 0.45 - 0.809999
      and
    blocking_delay equals 0.9 - 0.809999
and
from rule5 with condition(s):
    update_readonly equals 0.45 - 0.809999
      and
    number_of_messages equals 0.9 - 0.35
and
from rule4 with condition(s):
    update_readonly equals 0.85 - 0.719999
      and
    number_of_conflicts equals 0.8 - 0.679999
and
from rule2 with condition(s):
    number_of_conflicts equals 0.45 - 0.679999
and
from rule1 with condition(s):
    response_time equals 0.56 - 0.9
```

Figure 19 Second Trace

6. CONCLUSIONS

A model for an expert system for the adaptation control for the concurrency mechanism of a distributed data base system was proposed. The main features of this system can be summarized as follows:

- expert decision making, several candidate algorithms for the concurrency controller are tested in parallel on the same parameter values. The decisions are based on expert knowledge.
- decision making under uncertainty, due to the incomplete and unorganized knowledge the decisions are associated with uncertainty.
- dynamic behavior, the expert system reacts to performance changes that occur during the decision making process.
- on-line, real-time operation without human assistance.

A prototype expert system was implemented in PROLOG and tested on a scenario that represents a subset of the problem space.

There are two directions for further research that are of interest. First the knowledge base of this system needs a lot of work. The performance comparison of concurrency control algorithms is of interest. Second other areas in distributed system could be subject to expert decision making, too. In the area of network partitioning several applications are possible.

BIBLIOGRAPHY

{1} Bernstein, P. A. and Goodman, N., Fundamental Algorithms for concurrency control in distributed Data Base Systems, Tech. Report CCA-80-05, Computer Corporation of America, February 1980.

{2} Bhargava, B. and Riedl, J. A Model for adaptable Concurrency Control, CSD-TR-609, Department of Computer Sciences, Purdue University, June 1986.

{3} Bhargava, B. and Riedl, J, The design of an adaptable distributed System, IEEE 10th COMPSAC Chicago, Il., Oct. 1986.

{4} Papadimitriou, C. H., The serializability of concurrent data base updates, J. Association of Computing Machinery, vol. 26, pp. 631 - 653, Oct. 1979.

{5} Bhargava, B. and Hua, C. T., A causal Model for analyzing distributed Concurrency Control Algorithms, IEEE Transactions on Software Engineering, vol. SE-9, no. 4, Jul. 1983.

{6} Waterman, D. A., A Guide to Expert Systems, Addison-Wesley Publishing Company, Reading, Ma., 1986.

{7} Hayes-Roth, F., Waterman, D. A. and Lenat, D. B., (editors), Building Expert Systems, Addison-Wesley Publishing Company, Reading, Ma., 1984.

{8} Hayes-Roth, F., Rule-based Systems, Communications of ACM, vol. 28, no. 9, Sept 1985.

{9} Brownston, L., Farrell, G. F., Kant, E. and Martin N., Programming Expert Systems in OPS5, An Introduction to Rule-based Programming, Addison-Wessley Publishing Company, Reading, Ma., 1985.

{10} Whinston, P. H., Artificial Intelligence, Addison Wesley publishing Company, Reading, Ma., 1984.

{11} Nielsson, N. J., Artificial Intelligence, Springer Verlag, 1982.

{12} Zadeh, L. A., Review of Glenn Schafer: A mathematical Theory of Evidence, The AI Magazine, Fall 1984.

[13] Zadeh, L. A., A simple View of the Dempster-Schafer Theory of Evidence and its Implication for the Rule of Combination, The AI Magazine, pp. 85 - 90, Summer 1986.

[14] Bratko, I., PROLOG, Programming for Artificial Intelligence, Addison-Wesley Publishing company, Reading, Ma., 1986.

[15] Duda, R., Gaschnig, J. and Hart, P., Model Design in the Prospector Consultant System for Mineral Exploration, In: Expert Systems in the Microelectronic Age, Michie (editor), Edinburgh University Press, 1979.

[16] Buchanan, B. G. and Shortliffe, E. H., Rule-based Expert Systems, The MYCIN Experiments of the Stanford Heuristic Programming Project, Addison-Wesley Publishing Company, Reading, Ma., 1982.

[17] Bhargava, B., Performance Evaluation of the optimistic Approach to distributed Data base Systems and its Comparison to Locking, in: Proceedings 3rd IEEE International Conference on Distributed Computing Systems, Miami, Fl., Oct. 1982.

[18] Bhargava, B., Resilient Concurrency control in distributed Data base Systems, IEEE Transactions on Reliability, vol. R-32, no. 5, Dec. 1983.

[19] Reed, D. P., Implementing Atomic Actions on Decentralized Data, ACM Transactions on Computer Systems, vol. 1, no. 1, Feb. 1983.

[20] Bhargava, B. and Leu, Pei-Jyun, Multidimensional Timestamp Processing, Proceedings 10th IEEE COMPSAC Conference, Chicago, Il., Oct. 1986.

[21] Bhargava, B. and Lilien, L., A Review of Concurrency and Reliability Issues in Distributed Data Base Systems, in: Bhargava, B. (editor), Concurrency and Reliability in Distributed Systems, Van Nostrand & Reinhold, 987.

[22] Liskov, B., On linguistic Support for distributed Programs, Proc. 9th ACM SIGACT - SIGPLAN Symposium. Principles of Programming, Albuquerque, NM, pp. 7 - 19, January 1982.

[23] Popek, G. J. and Walker, B. J., The LOCUS distributed System Architecture, The MIT Press, 1985.

[24] Parker, D. S., Popek, G. J., Rudsin, G., Stoughton, A, Walker, B. J., Walton, E., Chow, J. M., Edwards, D., Kiser, S. and Kline, C., Detection of mutual Inconsistencies in distributed Systems, IEEE Transactions on Software Engineering, vol. SE-9, no. 3, May 1983.

[25] Pecoria, V. J., EXPRS, A Prototype Expert System Using PROLOG for Data Fusion, The AI Magazine, pp. 37 - 41, Summer 1984.

[26] Clark, K. L. and McGabe, F. G., PROLOG: A Language for Implementing Expert Systems, In: Hayes, J. E. and Michie, D. (editors), Machine Intelligence, vol. 10, pp. 455 - 475, 1980.

[27] Mizoguchi, F., PROLOG Based Expert System, New Generation Computing, vol. 1, no. 1, pp. 99 - 104, 1983.

[28] Subrahmanyam, P. A., The "Software Engineering" of Expert Systems: Is PROLOG appropriate?, IEEE Transactions on Software Engineering, vol. SE-11, no. 11, pp. 1391 - 1400, Nov. 1985.

[29] Bobrow, D. G., If PROLOG is the Answer, What is the Question? or What takes to support AI Programming Paradigms, IEEE Transactions on Software Engineering, vol. SE-11, no. 11, pp. 1401 - 1408, Nov. 1985.