Forward and Backward Chaining Techniques of Reasoning in Rule-Based Systems

Bareen Haval Sadiq Mzori

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
July 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Nazim Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Rashad Aliyev
Supervisor

Examining Committee

1. Prof. Dr. Rashad Aliyev
2. Assoc. Prof. Dr. Benedek Nagy
3. Asst. Prof. Dr. Ersin Kuset Bodur

ABSTRACT

The forward and backward chaining techniques are well-known reasoning concepts used in rule-based systems in Artificial Intelligence. The forward chaining is data-driven, and the backward chaining is goal-driven reasoning methods.

The aim of this thesis is to present the implementation of above concepts. The matching process between facts and rules, and the conflict resolution strategy in forward chaining are used. The depth-first search in both forward and backward chaining is performed. The backtracking process in backward chaining employs the Prolog programming language which is also discussed in this thesis. Some examples for better understanding the forward and backward chaining techniques are provided.

Keywords: forward and backward chaining techniques, matching process, conflict resolution, depth-first search, backtracking, Prolog language

ÖZ

İleri ve geri zincirleme teknikleri Yapay Zeka kural tabanlı sistemlerde kullanılan analiz kavramları iyi bilinmektedir. İleri zincirleme veri odaklı ve geri zincirleme hedef odaklı analiz yöntemleridir.

Bu tezin amacı, verilen kavramların uygulamasını sunmaktır. Gerçekler, kurallar ve ileri zincirleme uyuşmazlık çözümü stratejisi arasındaki eşleştirme işlemi kullanılır. İleri ve geri zincirleme derinlik öncelikli arama yapılır. Bu tezde, Geriye zincirleme olarak geriye izlemeli arama süreci de tartışılmıştır. Programlama dili olarak Prolog kullanır. İleri ve geriye zincirleme tekniklerini daha iyi anlamak için bazı örnekler verilmiştir.

Anahtar Kelimeler: Ileri ve geri zincirleme teknikleri, eşleştirme işlemi, uyuşmazlık çözümü, derinlik öncelikli arama, geriye izlemeli arama süreci, Prolog dili

ACKNOWLEDGMENT

First of all, I would like to thank my supervisor Prof. Dr. Rashad Aliyev, without help of whom this thesis could not have been finished. I thank him for his support.

I would also like to thank my father and mother for their permanent support. I am grateful forever to them for helping me find my path in life.

Many thanks to my husband Hariwan, and without him I could not be here. I thank him for supporting me to reach my dream and make it possible to come true. I also thank my lovely daughter Baran who is the most precious person for me in this world. Finally, thanks to my best friend, Sarkar, for the support he gives me when I need it.

TABLE OF CONTENTS

ABSTRACT……..iii

ÖZ.....iv

ACKNOWLEDGMENT...…….....v

LIST OF TABLES…..……...…..viii

LIST OF FIGURES…….……...……ix

1 INTRODUCTION……1

2 REVIEW OF EXISTING LITERATURE ON FORWARD AND BACKWARD CHAINING TECHNIQUES……..…..…6

3 RULE-BASED SYSTEMS. FORWARD CHAINING TECHNIQUE.....11

3.1 Rule-based systems…….……11

3.2 Forward chaining technique……..…….…….13

3.3 Matching in forward chaining.....14

3.4 Conflict resolution in forward chaining……...……...20

3.5 Depth-first search in forward chaining…….....…….…….24

3.6 Advantages and disadvantages of forward chaining…….……..27

4 BACKWARD CHAINING TECHNIQUE……...…….28

4.1 Basic principle of backward chaining …….……...…28

4.2 Resolution strategy in backward chaining……..…31

4.3 Backward chaining algorithm……...…….....32

4.4 Depth-first search in backward chaining……32

4.5 Using backward chaining in Prolog language……...…….....35

4.5.1 Backtracking and unification in backward chaining…….....…….36

4.6 Advantages and disadvantages of backward chaining…….....39

5 CONCLUSION……..41

REFERENCES……..42

LIST OF TABLES

Table 1: Forward chaining procedure for reaching goal ……22

LIST OF FIGURES

Figure 1: Representation of a structure of forward chaining ……...19

Figure 2: Graph representation of forward chaining from starting premise node to goal leaving node……...20

Figure 3: Depth-first search in forward chaining……...25

Figure 4: Representation of a structure of a backward chaining.....30

Figure 5: Depth-first search in backward chaining.....33

Figure 6: Graph representation of backward chaining from goal leaving node to starting premise node……..35

Chapter 1

INTRODUCTION

As it is known, the intelligence differ human from other species. In fact, the level of smartness is different from human to human, and it is a capability for making decisions, solving problems, fitting with a new statue, and the ability to learn [1].

Artificial Intelligence (AI) can make machines acting and thinking rationally like human. Some properties should be available in order the computer system will be intelligent, such as learning process from experience, taking intelligent actions while dealing with environment, or in other words, being adaptable to the hazardous environment, performing any type of tasks without intervention of human, describe intelligent behavior etc.

The AI software is used in our daily routine, for example, smartphones, computers, cars and so on. The other real-world applications of AI are medicine, industry, pattern and speech recognition, game playing, finance, marketing, robotics etc.

The thought of a smart machine is not new. It began in early ages with fantasies and myths. It started with philosophers who raised the possibility of building of smart "intelligence" machines. Through centuries, this idea was passed and developed from Indian, Chinese and Greek philosophers to philosophers such as Aristotle, Euclid and

al-Khwarizmi. In the 17th century Thomas Hobbes, Leibniz and René Descartes discovered the possibility that all logical thought can be represented as algebra or geometry, and their work became the guiding principle of AI studies. In the 20th century, the mathematical logic was used to provide the fundamental breakthrough that made AI to be comprehensible. The works of scientists, David Hilbert and Alan Turing who are also considered pioneers in AI led to what is called computer science.

In the 1940s and 1950s, the efforts to develop machines that could have intelligence, began by scientists Herbert Simon, Allen Newell and Cliff Shaw. They conducted experiments in writing programs to imitate human thought processes. The result of their work was the logic theory, but the year 1956 marked the start of AI. The first conference in AI was organized in Dartmouth [2].

After the development of the field of AI the new system called expert system came to life, and the first expert systems were formed in the 1970s, and then propagated in the 1980s. Expert systems were the first active forms of AI software. Although at the beginning there was concern about acceptance of expert system as AI program in a society, because it used a particular range of knowledge to solve narrow problems. But this idea vanished after two new concepts were proposed in 1972. Roger Schank formed the concept of "script" in 1972, and Minsky introduced the concept of "frame" in 1975. A combination of these two improved concepts gave the ability to realize the inference strategies in expert systems.

Expert system is a computer system enabling the imitation of the decision-making capability of human. They are designed to solve intricate problems by reasoning about knowledge. In general, there are three primary components in ruled-based expert system: user interface, knowledge base, and inference engine.

User interface plays the role of a bridge between the user and the expert system. This is a responsibility of a system engineer to build the user interface for a particular problem. The user interface permits the user to submit a query to the expert system in order to get an advice. So the query and the advice are considered as input and output of the expert system, respectively.

Knowledge base is very important part of the expert system. It is simply a technology that stores the rules and facts, basically in the form of IF…THEN statements. These statements are provided by human expert. The knowledge base demonstrates the facts that the system needs to solve the problem and to reach the goal.

Inference engine is responsible for providing the rules (logical) to the knowledge base and then generate new facts by using these rules. One of the techniques used to reach this goal is the production rules and their types: the forward chaining and the backward chaining. In addition, the inference engine needs a strong memory to save all the initial and new facts provided by the interface engine.

Rule-based systems are used in a variety of problems. For example, the CLIPS expert system was created by NASA in 1985 that led to the development the JESS expert

system, which was the JAVA application of CLIPS. These two expert systems were built with the rule-based production idea. The rule-based systems are also effectıvely used in finance, law, medicine, business, manufacturing, education, computer science and computer games [3-4].

Production rules are a set of statements represented in the form of IF…THEN statements. For example, "IF A1 and B1 are true THEN C1 is true, and the action E1 must be taken". The conditions A1 and B1 are called the left-hand side of the rule (LHS) and C1 and E1 are called the right-hand side of the rule (RHS). So if the LHS elements correspond to the facts in the working memory then the rule is activated and the RHS is executed. Another case is that the rule can be negated which means that the LHS will be only executed when no fact in the working memory matches. The actions can update the working memory by adding, modifying, or removing the facts [3-4].

The process of the output of one rule activating another rule is called chaining.
Chaining technique is to break the task into small procedures and then to inform each procedure within the sequence by itself. Two types of chaining techniques are known: forward chaining and backward chaining.

Forward chaining is a data-driven reasoning, and starts with the known facts and tries to match the rules with these facts. There is a possibility that all the rules match the information (conditions). In forward chaining, firstly the rules looking for matching facts are tested, and then the action is executed. In the next stage the working

memory is updated by new facts and the matching process all over again starts. This process is running until no more rules are left, or the goal is reached. Forward chaining is useful when a lot of information is available. Forward chaining is useful to be implemented if there are an infinite number of potential solutions like configuration problems and planning.

The opposite of a forward chaining is a backward chaining, i.e. in contrast to data-driven reasoning forward chaining, a backward chaining is a goal-driven reasoning method. The backward chaining starts from the goal (from the end) which is a hypothetical solution and the inference engine tries to find the matching evidence. When it is found, the condition becomes the sub-goal, and then rules are searched to prove these sub-goals. It simply matches the RHS of the goal. This process continues until all the sub-goals are proved, and it backtracks to the previous step where a rule was chosen. If there is no rule to be established in an individual sub-goal, another rule is chosen. The backward chaining reasoning is good for the cases where there are not so much facts and the information (facts) should be generated by the user. The backward chaining reasoning is also effective for application in the diagnostic tasks [5].

Chapter 2

REVIEW OF EXISTING LITERATURE ON FORWARD AND BACKWARD CHAINING TECHNIQUES

[6] investigates a natural condition ensuring a property that a logic program always has an extension. This condition is an extension of Rieter's normality in default logic theory in combination with information system. The general forward chaining construction is given which is applied to nonmonotonic rule system.

The paper [7] is devoted to the constructing of stable models of logic programs by using the forward chaining technique. It is proven that each model of the program which is stable is also a stable submodel of the original program. The proposed algorithm calculates the subset of the base of the program, and this property of algorithm differs the new methodology from the existing ones. The forward chaining construction makes it possible to use the suggested semantics in both logic programs and default theories.

In [8] the progress on some existing works on forward chaining in terms of increasing the speed of this technique is conducted, and two additional properties are offered. The first property is an augmentation of rule processing which is performed by reference tables, and the second property is the implementation of empirical ordering strategy for hypotheses where data collected from the system are used to

optimize the performance of the system. The additional properties are supported by experiments carried out by the authors.

In [9] the concurrently executable operations with varying durations, and metric quantities are used by forward chaining technique in order to generate a complex and lengthy plans to solve the planning problems. Combining both features with some other properties such as search control knowledge is very powerful to planning resources.

The method presented in [10] is used to optimize the naïve forward chaining algorithm by using the conceptual graphs rules. Due to this algorithm the number of rules checks and the cost of each check are reduced, and goal-oriented form of forward checking is achieved. Moreover, this process is realized without any extra runtime cost.

In [11] the development process of inference engine of an expert system working in ternary grid knowledge model is discussed. The strategy used for this reason is a forward chaining model with recursive process. The advantage of the proposed model over iterative forward chaining model in terms of improvement the efficiency of the inference process is mentioned.

An expert system with stratified forward chaining model based inference engine is presented in [12]. The expert system designed with this tool uses production rules, and the user interface in this expert system is web based. The proposed chaining is

implemented on forward, backward, and mixed chaining techniques. The special kind of inference called the deontic inference making the presence of deontic rules is discussed.

The improvement of inference engine of expert system which is based on backward chaining with recursive process and is able to work in ternary grid expert system is discussed in [13]. The experimental results show that the inference strategy works properly and more dynamic in comparison with existing models. The efficiency of the developed model is also convenient in detecting the new rules to provide new solutions.

In [14] the backward-chaining evolutionary algorithms (EA) are given which are faster than EA algorithms, and are applied to any form population based search, crossover, mutation etc. The recursive evaluation of individuals is performed by the proposed algorithms. The theoretical and empirical behavior and benefits of the algorithms are analyzed by the experiments using standard and backward chaining algorithms.

The description of the implementation process of backward chaining in CLIPS programs is given in [15]. The efficiency of the implementation of this chaining technique consists in gaining the simple and short forms of CLIPS programs. The extended version of CLIPS program called ECLIPS is implemented, and the comparative analysis of both programs is done.

The authors of the paper [16] investigate the implementation of matrix calculation in knowledge based system in the conclusion process of backward chaining. The advantage of this approach is to increase the speed in reaching the conclusion of the system. By using the third dimension in S matrix it is possible to simplify the conclusion process. The convenient production system is constructed.

In the paper [17] the hybrid algorithm is provided which is used for effective backward chaining procedure on very big datasets, and these datasets are expressed by OWL Horst ruleset. This algorithm is tested on different datasets. The effectiveness of the algorithm is evaluated by the performance analysis of the considered approach.

Using backward-chaining and partial order planning for the problems of practical planning is sometimes very inconvenient, because they are not always successful control strategies for taking the difficulties of real-world problems into consideration to solve them. To overcome these difficulties of above control strategies, the adapting Hierarchical Task-Network planning is described that is used for total-order control strategy [18].

In many cases the linear logic programming languages are implemented using the backward chaining technique. The combination of backward chaining with forward chaining provides better results in many applications. In [19] the proof theoretic foundation of the combination of both forward and backward chaining techniques is proposed.

In [20] the combination of forward and backward chaining techniques for a constraint logic programming (CLP) framework is considered. The combination of above techniques takes the advantages and disadvantages of each type of chaining techniques. According to answers and constraints, the completeness of the language is established.

Chapter 3

RULE-BASED SYSTEMS
FORWARD CHAINING TECHNIQUE

3.1 Rule-based systems

The term knowledge means the understanding of a specific domain. The person who has some kind of knowledge and passing it to another person is called an expert. The expert must have a wide knowledge of facts and rules, and use experience in a particular domain, and this person is called a domain expert.

Any rule-based system can be created in the presence of some components. First of all, a set of facts which are used in a working memory of a system must be available. Secondly, a set of rules must be there encompassing the possible actions to be taken. Finally, the condition must be used which is determining whether a solution is found or there is no solution at all.

It is very hard to represent the human thinking process in the form of algorithm, because this process is an internal process in the brain of a human. The expert can express his/her knowledge as rules to describe and then solve problems. In artificial intelligence, the rules are used to represent the knowledge, and the knowledge can be better represented in the form of IF-THEN linguistic rules. In the IF part, the logical

conditions are used. In THEN part some actions or results are given. The structure of a rule is easy to create and understand. For example:

IF it is raining

THEN take an umbrella

The rules are made of two parts. The IF part (left hand side or LHS) is called the condition (antecedent) and the THEN (right hand side or RHS) part is called the action (consequent).

If the conditional part of the rule (antecedent) is true, then the conclusion part (consequent) is asserted. This can be a goal or a new fact to be added to the working memory.

The rule can have more than one condition connected with (conjunction) AND, (disjunction) OR, or a mixture of both. For example:

IF < condition 1 >	IF < condition 1 >
AND < condition 2 >	OR < condition 2 >
.	.
.	.
.	.
AND < condition n >	OR < condition n >
THEN < action >	THEN < action >

The condition part of a rule consists of the object and its value. The object is represented in a linguistic form, and linked to its value by using an operator.

Rule-based representation of knowledge for designing the production systems was proposed by Newell and Simon in the early seventies which is the foundation of the current rule-based systems. The idea came from how the human could solve a problem by implementing human knowledge represented in the form of production rules.

3.2 Forward chaining technique

In a forward chaining, the system's purpose is to determine results from premises.

Forward chaining starts from the set of facts (the available data) and then checks if the given rules are satisfied. If so (there is a matching) then the rule is executed (fires). This process continues until the goal is found (assuming that just one answer is required), or there are no new facts to be added.

An important notice is that the fact is not "new" if it is a renaming of a known fact. Renaming is that two sentences are renaming each other if they are the same except the names of the variables are different.

The rules are expressed in a close form to human language but in reality the rule-based system must be represented in a way that the limited machine-processes can understand. There is no standard syntax rule. LHS of the rule consists of object name followed by the objects attributes and values.

The forward chaining algorithm is represented as the sequence of the following steps:

- Initial facts are inputs from the user to be set into the database (working memory);
- Check LHS of the production rules;
- If the logical condition part of a rule (IF part) matches, then the rule "fires";
- Execute RHS actions;
- Retract old conditions/facts;
- Input new conditions/facts;
- Do other input-output actions, unifications etc.
- Repeat until no other rules fire.

3.3 Matching in forward chaining

Matching of patterns in chaining process is also known as unification. So the application of the chaining procedure depends on unification.

In the unification a set of binding of variables is considered. In a unification process two atomic sentences are compared and a unifier is returned if one of the given sentences exists.

The literals are shown as a list in which the name of the predicate is the first element, and the rest are arguments. The arguments can be a single element or a list of elements.

To unify two literals, it must be examined whether their first elements are same. If yes, continue. Otherwise, they will not be unified.

Consider the following simple unifications.

1) likes (murat,x)

   likes (murat,pizza)

   Unification= {x|pizza}

2) likes (murat,x)

   likes (y,pizza)

   Unification= {x|pizza, y|murat}

3) likes (murat,x)

   likes (x,pizza)

   Unification= {fail}

In the matching step the rules are compared with the facts that are in the working memory and the rules that are satisfied are determined. Firstly, the syntax rule is defined, and afterwards it is specified how the facts match the conditions of the rules. The system must consider the rules in conjunct ordering to minimize the total cost. The problem of the matching process is that the rule-based system takes almost 90% of its running time for the matching process (between rules and facts in the working memory). Another problem is that the pattern matching process repeatedly happens in all the iterations of the fact list, and so a new fact could be added, or some old facts could be removed. The satisfied rules must be updated in all the iterations and maintained. In most cases, the rule actions change only some facts in the fact list which is called a temporal redundancy.

The simple example of the forward chaining is given below:

R1: IF A=5 AND B=7

THEN C=10

R2: IF C=10 AND D=12

THEN E=15

R3: IF E=15

THEN F=18

The following facts are present: A=5, B=7, and D=12. The values of A and B help infer the value of C from the rule 1, then the rule 2 fires due to the inferred C and the known fact D (so E is inferred), and finally the rule 3 fires to infer F. So using forward chaining the goal F is reached.

The solution of the problem is matching the rules to the facts one by one exhaustively in addition of using index techniques. The algorithm that represents the solution is the RETE algorithm.

RETE algorithm is a pattern matching algorithm created by Dr. Charles L. Forgy at Carnegie Mellon University. RETE means “net” in Latin. The RETE algorithm is very useful for matching facts against the patterns in a rule. Writing the rules in a particular way is more powerful than writing them randomly, and this will be performed by implementing the RETE algorithm.

Below some examples of forward chaining are provided. The inference mechanism will provide an appropriate conclusion of the knowledge base. Some examples are also represented in decision tree form.

Let’s see an example (knowledge base) with a forward chaining:

R1: IF X is human THEN X is mammal

R2: IF X is a mammal THEN X is a life form

R3: IF X is a life form THEN X is mortal

Fact: Murat is human.

Goal: Is Murat a mortal?

The unification process determines that X is Murat. So the rule 1 fires (Murat is a mammal), then rule 2 fires (Murat is a life form). Finally, the rule 3 fires (Murat is a mortal).

Another knowledge base is given below:

R1: IF hungry THEN make food

R2: IF make food THEN need ingredients

R3: IF need ingredients THEN go to refrigerator

R4: IF refrigerator is full THEN grab ingredients

R5: IF tomato AND lettuce THEN make salad

R6: IF meat THEN make barbecue

R7: IF chicken THEN make fried chicken

R8: IF refrigerator is empty THEN sleep

Find the conclusion, if the following facts are present: hungry, refrigerator is full, meat.

Firstly the rule R1 fires, then the rules R2, R3, R4 fire continuously after each other, and finally the rule R6 fires. So the conclusion is to make barbecue.

Another knowledge base is provided below, and it is to prove the conclusion.

R1: IF A OR B

THEN C

R2: IF D AND E AND F

THEN G

R3: IF C AND G

THEN H

The following facts are presented:

B, D, E, F.

Goal: prove H.

The structure of a forward chaining example is given in the figure 1.

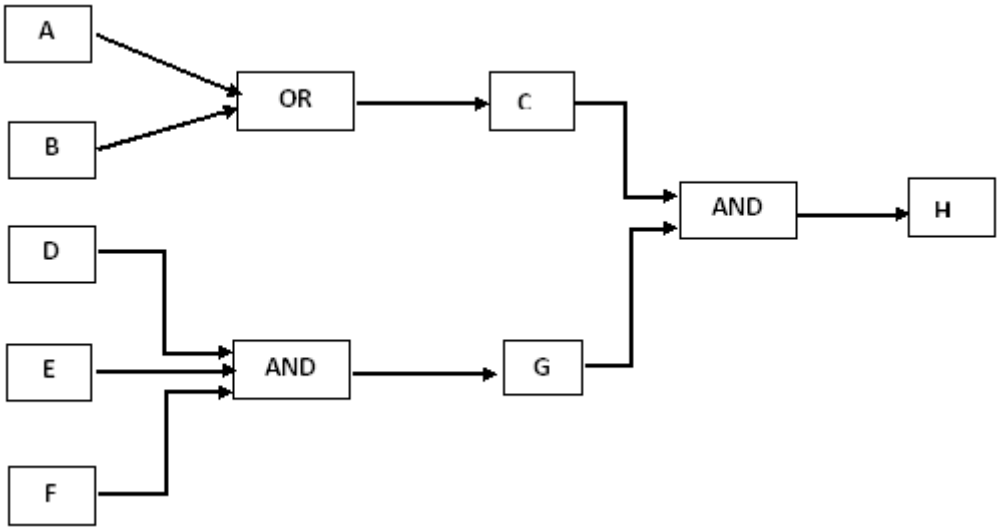

Figure 1: Representation of a structure of forward chaining

From the first rule C is inferred, from the second rule G is inferred, and finally, from the third rule H is inferred.

The forward chaining can also be represented in a graph form with nodes and links. Suppose the following rules are available:

R1: IF A THEN B

R2: IF B AND C THEN D

R3: IF D AND E THEN F

R4: IF F AND G THEN H

For the above rules the forward chaining is illustrated in a graph form starting from premise node to a goal leaving node (Figure 2).

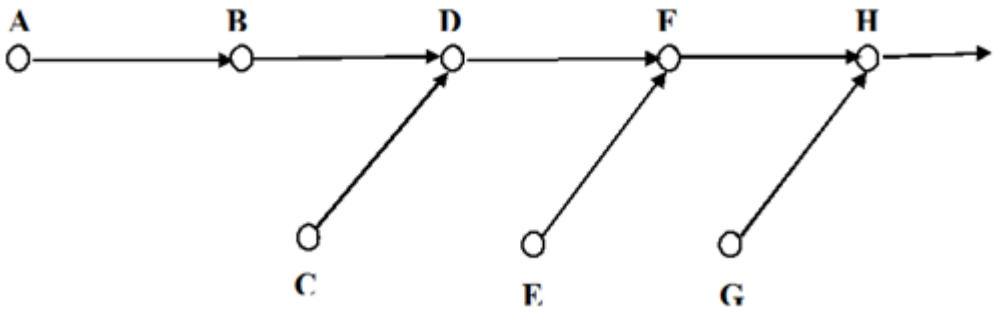

Figure 2: Graph representation of forward chaining from starting premise node to goal leaving node

3.4 Conflict resolution in forward chaining

If there is a possibility of firing two or more rules in which IF parts are satisfied to be executed at the same time (the rules have the same condition), this process is called a conflict set. In this case, the inference engine must choose the rule to be fired from the set. Choosing which rule is to be fired when more than one rule can fire in a given round is a function of a conflict resolution strategy.

A simple example with a knowledge base causing a conflict set is given below:

R1: IF it is hot

THEN wear short

R2: If it is hot

THEN go swimming

R3: If it is hot

THEN put sun-cream

If the fact is present that “It is hot”, then all three rules will be triggered, so we need to find out which action should be taken. This problem can be solved by using the following strategies:

1) The rule with the highest priority fires first if there are few rules in the knowledge base. The rules should be arranged in a suitable order in the knowledge base. Each rule is given the priority, and the rule with the highest priority is selected.

Assume the following rules are available:

R1: IF A AND B AND C THEN D

R2: IF A THEN E

R3: IF D AND E THEN F

R4: IF E AND F THEN G

R5: IF A AND G THEN H

R6: IF F AND G AND H THEN I

The following facts are given: A, B, and C. The purpose is to prove whether or not it is possible to reach the goal I. The forward chaining procedure can be used to reach the goal, and it is represented in table 1.

Table 1: Forward chaining procedure for reaching goal

Facts available	Triggered rules	Rules to be fired
A,B,C	1,2	1
A,B,C,D	2	2
A,B,C,D,E	3	3
A,B,C,D,E,F	4	4
A,B,C,D,E,F,G	5	5
A,B,C,D,E,F,G,H	6	6 (Stop. The goal is reached)

2) If a specific rule gives more information than a general one, then this specific rule fires, and this method is called the longest matching strategy. This method is most helpful in non-standard cases.

3) The rule that was most recently added to the database, fires. The new elements in the working memory are to fire before the older ones. By this, the system will track through a single chain of reasoning, instead of keeping on sketching new conclusions from the old data.

The conflict resolution can be defined as a set of pairs of the following components:

< IF…THEN rule, matching of element from a working memory >

When it is decided which rule fires first, the conflict resolution makes it sure that the same rule can't be executed twice.

This strategy helps the forward chaining systems to get a reasonable performance, but the construction of the rules is the most important point because if they are not written in a proper way, we will have limited control over what will happen with the preconditions defined as precisely as possible when different rules should fire.

To control the performance of the system, the special working memory elements are used. For example, we might decide that there are some basic stages of processing in doing some tasks, and certain rules should fire at a given stage - we could have a special working memory element and add to the preconditions of all the relevant rules. When this stage is complete, the working memory element is removed.

There are two types of a conflict resolution: general conflict resolution and problem specific conflict resolution.

The rule selection process is very important, because an unaware selection of rules makes various rules to be fired, and many of these rules might not be useful to reach the goal. As a result, the user is asked many unnecessary questions. Therefore, the system performance will be degraded.

In rule-based systems the knowledge is considered by meta-rules. The meta-rules are used to make the rule selection process more efficient. The meta-rules are used to determine the strategy of the conflict resolution. They represent the knowledge regarding how the system will work.

The meta-rules can define the following property: if the knowledge of one expert is more reliable and trusted in comparison with rules of another expert, this is called a meta knowledge. A meta knowledge can be also interpreted as a knowledge about knowledge.

The meta-rules are given higher priority.

After matching the rules and facts, and using the conflict resolution, the system will choose a rule to find the goal and in the last stage this rule will fire. This process might cause a modification and the user will be asked questions in addition to the new calculations and new actions. After all, the goal will be presented to the user.

3.5 Depth-first search in forward chaining

Sometimes the rules in forward chaining can be represented in a tree form from the root to the depth. Below the following rules are given:

R1: IF A THEN B

R2: IF B THEN C AND D

R3: IF C THEN E

R4: IF D THEN F

R5: IF F THEN G AND H

The depth-first search in a forward chaining for above rules is represented in the figure 3.

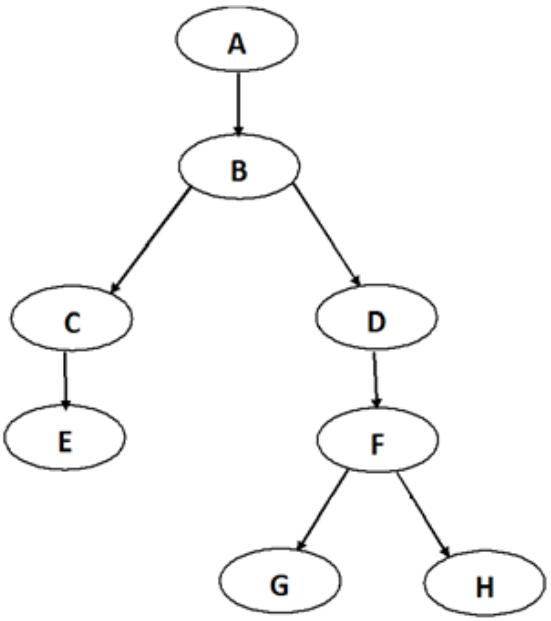

Figure 3: Depth-first search in forward chaining

Forward chaining usually has rules that are useless in the system when it tries to find the goal. Let's consider the following rule set:

IF A AND B THEN D

IF A AND C THEN E

IF A AND F THEN G

IF D AND E THEN H

IF D THEN J

IF E AND F AND H THEN K

The following facts (inputs) are present: A, B, and C. The reasoning goal is J.

Below two solutions of the forward chaining process are given: the sub-optimal reasoning and the optimal reasoning.

Sub-optimal reasoning:

IF A AND B THEN D

IF A AND C THEN E

IF D AND E THEN I

IF D THEN J

Optimal reasoning:

IF A AND B THEN D

IF D THEN J

As it can be seen, the sub-optimal solution is not efficient, and it will become worst if the number of rules increases. The heuristics are used to solve this problem. In general, the heuristics are classified into weak and strong forms. The weak heuristics prefer a simple rule sets that have a small number of conditions (IF part), and also analyze the syntactic difference between the current state and the goal state which is called a means-end analysis. The strong heuristics are known as meta-rules, intended being used for a specific domain of knowledge, and normally are more useful than weak heuristics.

The best meta-rules are the automatic strong heuristics. They learn dynamically to select the rule in each stage while the system is in the run mode.

3.6 Advantages and disadvantages of forward chaining

There are some advantages of forward chaining which are given below:

1) Runs great when a problem naturally begins by collecting data and searching for information that can be collected from it to be used in future steps;

2) Forward chaining has the capability of providing a lot of data from the available few initial data or facts;

3) Forward chaining is a very popular technique for implementation to expert systems, and systems using production rules in the knowledge base. For the expert system that needs interruption, control, monitoring, and planning, the forward chaining is the best choice;

4) When there are few facts and initial states, the forward chaining is very useful to be applied.

There are also some disadvantages of a forward chaining, and they are given below:

1) New information will be generated by the inference engine without any knowledge about which information will be used for reaching the goal;

2) The user might be asked to enter a lot of inputs without knowing which input is relevant to the conclusion;

3) Several rules may fire that have nothing to reach the goal;

4) It might produce different conclusions which are the causes of a high cost of the chaining process.

Chapter 4

BACKWARD CHAINING TECHNIQUE

4.1 Basic principle of backward chaining

Backward chaining works in reverse to forward chaining, and starts from the goal and tries to find data to prove its goal. Therefore, it is also called a goal-driven reasoning. After starting from the given goal, the search of THEN parts of the given rules (action part) (RHS) is conducted, and if the rule is found and its IF part (condition) matches the data in the database, then the rule is executed (fired). Otherwise, if the condition does not match the data (facts) in the database, the inference engine sets the rule that is working on a stack and makes a new subgoal to prove the condition in the current rule. The knowledge base keeps looking for rules to prove the subgoal. The process of stacking the rules is repeated until the knowledge base has no rules to prove the subgoal.

Consider the following rules:

R1: IF A AND B THEN C

R2: IF C THEN E

R3: IF A AND E THEN H

Facts:

A, B

Goal:

Prove H

At first, the system looks for a rule that proves the goal, in other words, searches the RHS of the rules, which is rule 3. Then the LHS of the rule is considered which is the IF part that contains the condition of the rule. In this stage the system tries to satisfy the condition, and if the facts match the condition of this rule, then the rule fires. If not, the system makes a subgoal that contains the new rule and tries to prove it. So in rule 3 the system looks at the LHS, and finds A and E in the database, but we only have A. Therefore, the system now will make proving E its new subgoal. E is proved in rule number 2, the LHS of rule number 2 is C. The new subgoal is proving C. The system is out the old subgoal in a stack. C is proved in rule number 1. The condition in rule number 1 can match the facts in the database. Now the system can prove the subgoals in the stack, and the goal is proved.

The structure of a backward chaining for the above problem is represented in the figure 4.

Backward chaining concept is used to build many types of expert systems, and especially the interactive systems simulate conversion between users and an expert person. Using backward chaining technique, the system will know what to ask and

when to ask the proper question. It helps to dismantle the complex problem into easy, small, fixed sections, and the system use them automatically if needed.

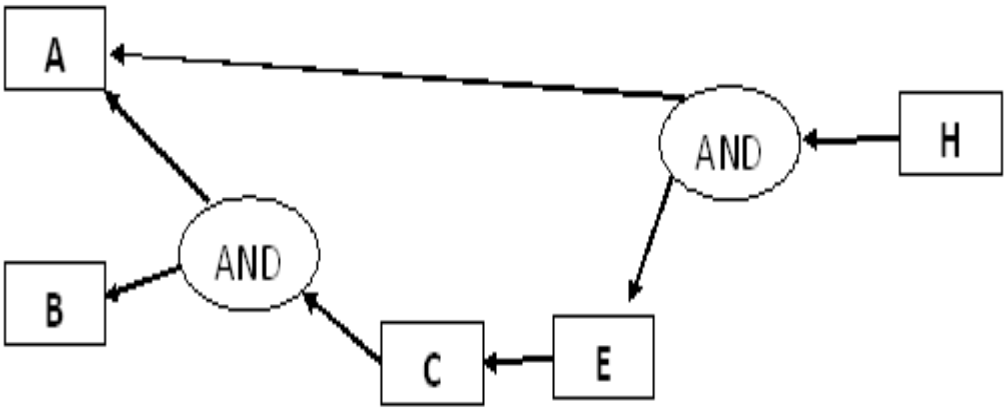

Figure 4: Representation of a structure of a backward chaining

Backward chaining is easy to implement, and therefore it is the default method for many expert systems tools. If a variable is given and the system must determine the value of this variable, and there is a rule that determines that value, then backward chaining will make the rule fire automatically to obtain the value. In case if executing the rule needs more information to fire, the system will have to execute some additional rules.

The systems that use backward chaining are also known as goal driven systems. They have a list of goals and try to execute. The list dynamically adds new goals to the top of the list and pushes old goals down the list. The idea is that the system works on the top goal of the list once, after it is executed and drops it from the list, and the next goal becomes the top goal and tries to execute it. This procedure is done until there is no goal to be executed in the list.

In backward chaining, the system starts from the high-level rules and can be expanded to any level needed to reach the decision by adding blocks of rules that cover specific decision details. These rules can be reused by the system in any multiple places. If in the future some new criteria are needed to prove a top goal, additional rules can be added to the block of rules and the system call and test them making it very easy to add rules and to expand a system.

The backward chaining allows the system to ask the user some information about the rules they have. Sometimes the user is not in a position to answer these questions or does not have the required information. In this case adding some more rules to the system will help the execution process, and these rules might be asked for information, and the user might be able to answer.

4.2 Resolution strategy in backward chaining

A resolution strategy in backward chaining is applied to combine sub-goals after some axioms are searched in order to make a goal deduction.

How does a resolution strategy work in backward chaining? Assume the following rules are available:

R1: IF A AND B AND C

THEN D

R2: IF D

THEN E

The axioms are given that A, B, and C are true. The goal is to deduce E if a sub-goal D is true.

In order to prove that E is true we need to make sure that the sub-goal D is true. The sub-goal D is proved to be true since the facts A, B, and C are true. So E is deduced from the fact that D is true.

The backward chaining works as long as no subgoal remains to be taken into account. If all the subgoals are satisfied, the goal is reached.

4.3 Backward chaining algorithm

The backward chaining algorithm can be represented in the following form:

1) The rule that matches the goal is selected.

IF the condition (IF part) is empty, ask the user for information.

ELSE

WHILE not end, AND we have the selected rules DO

2) Add the conditions of the rules

IF the condition is not met, THEN put the condition as a goal to solve

END WHILE.

4.4 Depth-first search in backward chaining

Suppose the following rules are given, and a depth-first backward chaining for these rules is illustrated in figure 5.

R1: IF A AND B AND C THEN D

R2: IF E AND F THEN A

R3: IF G AND H THEN B

R4: IF I OR J THEN C

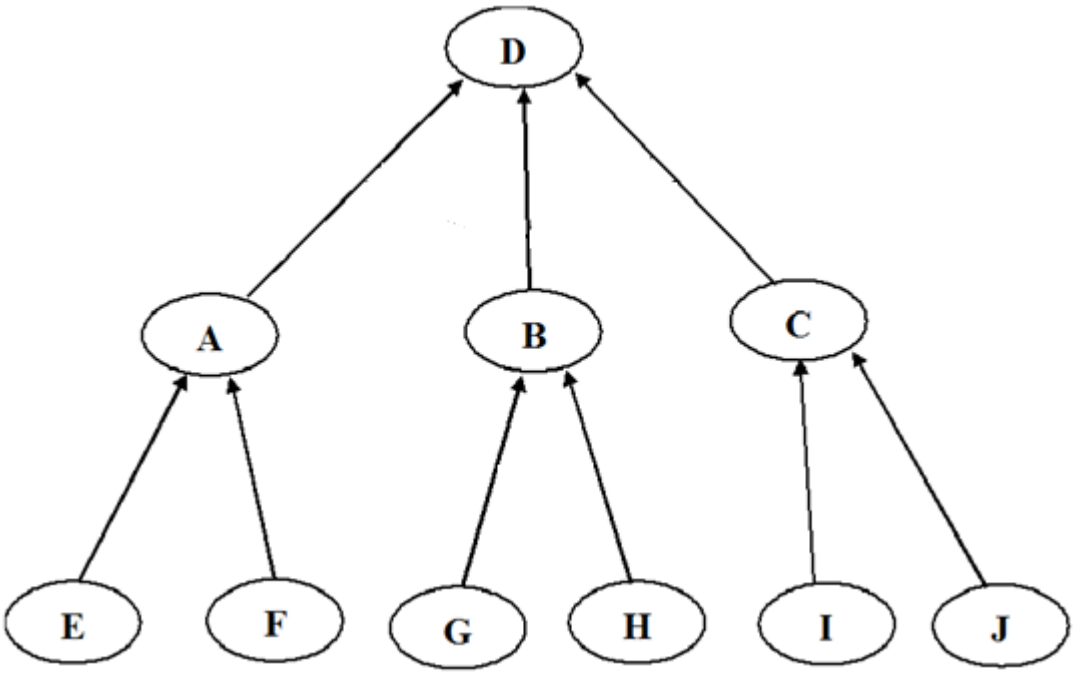

Figure 5: Depth-first search in backward chaining

Take a look at the following rules:

R1: IF customer has the priority

THEN customer gets discount for 20%

R2: IF customer makes a purchase for more than $200

THEN customer has priority

R3: IF customer is a member of the fan club

THEN customer has priority

Fact(s) given:

Customer makes a purchase for more than $200.

Goal: Will the customer get 20% discount?

The system looks for a rule that matches the goal, which is R1.

Then the system looks to the condition of R1 and try to satisfy it, and this condition becomes the new sub-goal (top goal). In this stage, the system will look for information to satisfy the condition.

It is to find a rule that its THEN part matches the new top goal. There is no information but THEN part of R2 matches the sub-goal. The condition of R2 becomes the new sub-goal and in the database there is a fact that satisfies the condition (makes a purchase for more than $200). So the rule R2 fires, and the system drops it from the list of sub-goals and the old sub-goal becomes the new sub-goal which is R1. In this stage the condition in R1 is satisfied and R1 fires. The final decision is the customer gets discount for 20%.

Like forward chaining, the backward chaining can be also represented in a graph form. Suppose the following rules are available:

R1: IF A THEN B AND C

R2: IF B THEN D AND E

R3: IF C THEN F

R4: IF D THEN G

In figure 6 the graph representation of backward chaining from goal leaving node to starting premise node is given:

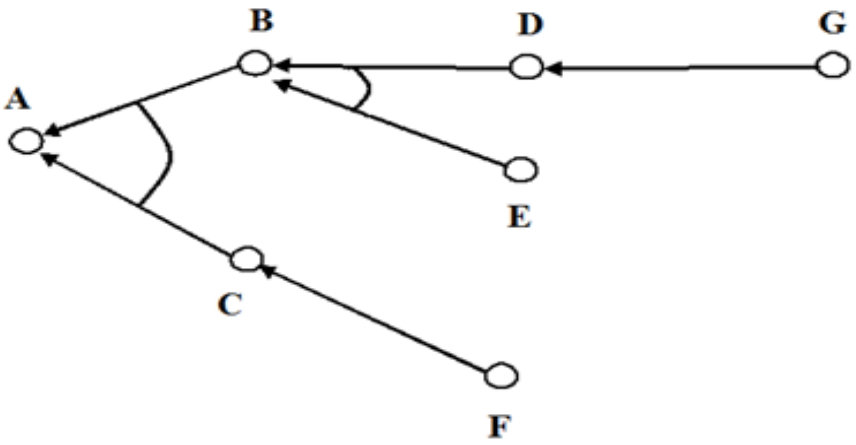

Figure 6: Graph representation of backward chaining from goal leaving node to starting premise node

4.5 Using backward chaining in Prolog language

Prolog is a logic-based programming language which allows backward chaining to deal with rules, and is successfully used in building the expert systems. Prolog is very efficient for using in pattern matching, backtracking execution etc. These features make Prolog the powerful tool in building an expert system. In Prolog the code of the program can be written in easy way (more declarative than procedural). This is possible because of the built-in pattern matching and backtracking provides an automatic control on the program.

Prolog uses a built-in backward chaining. The rules in Prolog are used for representing the knowledge, and the inference engine of Prolog is used to determine the results. Prolog uses backward chaining to get the conclusion; the rules have goals and sub-goals. The structure of the rules and the strategy of the inference are very important for various expert system applications. The only improvement that must be done is the dialog with the user.

Using a backward chaining, Prolog executes the division of an entire problem into the small problems, and afterwards these problems are solved.

The Horn clauses are the basis of any program in Prolog. The inference mechanism in Prolog is working according to backward chaining. The backtracking process is performed to find the unification in Prolog. The relations and predicates are used in Prolog.

Inference in Prolog is based on backward chaining.

4.5.1 Backtracking and Unification in Prolog

Backtracking is one of most important properties of Prolog intended to find all the solutions of the problem, and if some alternatives fail, Prolog can reach the necessary alternative.

The backtracking process in Prolog runs automatically, if it is important to reach the goal.

Unification in Prolog plays a role of pattern matching between the goal and head part of clause. For example, if likes(john,X) is to be unified with likes(john,strawberry), then X is to be unified with strawberry.

The unification process occurs if the terms are same or if these terms consist of variables which are instantiated with terms in which the final terms are equal. The meaning of above words can be interpreted in the following form:

book and book unify

57 and 57 unify

man(socrat) and man(socrat) unify

But

john and socrat don’t unify

man(john) and man(socrat) don’t unify

Unification is the substitution procedure which is performed by Prolog to derive new relationships. Suppose the following database of facts is given:

takes(john,mathematics).

takes(jack,history).

takes(mary,biology).

takes(mia,history).

takes(vincent,physics).

If the query is - takes(john,X), the database is searched and unification leads to X=mathematics. For the another query - takes(X,biology), database is searched and unification process occurs to lead to the solution X=mary.

In order to understand a backtracking process and unification in Prolog, the following problem is given.

likes (john,X) :- vegetable(X), color (X,orange).

vegetable(carrot).

vegetable(broccoli).

vegetable(spinach).

color(carrot,orange).

color(broccoli,green).

color(spinach,green).

Query: likes (john,X).

In order to find X, both conditions - vegetable (X) and color (X,orange) must be satisfied. From vegetable (X) we find that X= carrot, X=broccoli, and X=spinach. Next, color (X,orange) must be checked, but after unification two values of X which are X=broccoli, and X=spinach fail, since their colors are not orange. The only alternative satisfying both conditions of the query is X=carrot. So the solution of the problem is found: likes (john,carrot).

Another example about how a backtracking process works in Prolog rules is considered below:

celebrates_party(X):-

has_birthday(X), in_good_mood(X).

has_birthday(john).

has_birthday(jack).

has_birthday(nick).

in_good_mood(nick).

If the query is: celebrates_party(X), it is necessary to find a clause of has_birthday. It binds X to john. Then the goal in_good_mood(john) is checked. Since it does not match, i.e. there is no such data in database, it fails, and Prolog uses backtracking to come back to check whether there is another solution of the problem. The same backtracking process happens with jack. Finally, the subgoal has_birthday(nick) is checked. This needs the goal in_good_mood(nick) to match database. Since it is available, celebrates_party(X) succeeds with X=nick.

4.6 Advantages and disadvantages of backward chaining

Below some advantages of backward chaining are given:

1) The system will stop processing once the variable has its value. It's a “floor system”;

2) The system that uses backward chaining tries to set goals in order which they arrive in the knowledge base;

3) The search in backward chaining is directed;

4) While searching, the backward chaining considers those parts of the knowledge base which are directly related to the considered problem or backward chaining never performs unnecessary inferences;

5) Backward chaining is an excellent tool for specific types of problems such as diagnosing and debugging.

6) Compare to forward chaining, few data are asked, but many rules are searched.

There are also some disadvantages of backward chaining:

1) The goal must be known to perform the backward chaining process;

2) The implementation process of backward chaining is difficult.

Chapter 5

CONCLUSION

Rule-based systems are involved in human's daily routine, so understanding and implementation of these systems are important. Most expert systems are developed using production rules.

In this thesis, the properties of two reasoning strategies, which are important components in rule-based systems, are described: forward and backward chaining strategies. The main difference in implementation of these control strategies consists in the fact that in forward chaining the system is working from the facts to the conclusion, and this approach is called data-driven; but in backward chaining the system is working from the conclusions to the facts, and this approach is called goal-driven. The selection of the appropriate chaining strategy is up to the nature of the problem. It is necessary to mention that the behavior of inference procedure of backward chaining is similar to human expert.

This thesis investigates the matching and conflict resolution strategy in forward chaining, the depth-first search in both reasoning strategies, the conflict resolution and backtracking in backward chaining etc. The provided examples are very useful to understand the application principles of forward and backward chaining techniques.

REFERENCES

[1] Legg, S., & Hutter, M. (2007). Collection of Definitions of Intelligence. *The Journal of Frontiers in Artificial Intelligence and Applications, Vol.157,* pp. 17-24.

[2] Buchanan, B. (2005). A (Very) Brief History of Artificial Intelligence**.** *AI Magazine Volume 26, Number 4,* pp. 53-60.

[3] Lagun, E. (2009). Evaluation and Implementation of Match Algorithms for Rule-based Multi Agent Systems using the Example of Jadex. Master's Thesis. University of Hamburg.

[4] Griffin, N., & Lewis, F. (1998). A Rule-Based Inference Engine which is Optimal and VLSI Implementable. *IEEE International Workshop on Tools for AI Architectures, Languages and Algorithms,* pp. 246-251.

[5] Kingston, K. Knowledge based system development tools. *Encyclopedia of Life Support Systems (EOLSS).*

[6] Marek, V., Nerode, A., & Remmel, B. (1994). A context for belief revision: forward chaining-normal nonmonotonic rule systems. *Annals of Pure and Applied Logic 67,* pp. 269-323.

[7] Marek, W., Nerode, A., & Remmel, B. (1999). Logic programs, well-orderings, and forward chaining. *Annals of Pure and Applied Logic 96,* pp. 231-276.

[8] Schofield, M., & Saffidine, A. (2013). High Speed Forward Chaining for General Game Playing. *Proceedings IJCAI-13 Workshop on General Game Playing.*

[9] Bacchus, F., & Ady, M. (2001). Planning with Resources and Concurrency A Forward Chaining Approach. *Proceeding IJCAI'01 Proceedings of the 17th international joint conference on Artificial intelligence - Volume 1,* pp. 417-424.

[10] Baget, J. (2004). Improving the Forward Chaining Algorithm for Conceptual Graphs Rules. Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), pp. 407-414.

[11] Erdani, Y. (2011). Developing Recursive Forward Chaining Method in Ternary Grid Expert Systems. *IJCSNS International Journal of Computer Science and Network Security, Vol. 11 No. 8,* pp. 126-130.

[12] Bezzazi, E. (2007). On some inferences based on stratified forward chaining: An application to e-Government. *Advances in Information Systems Development,* pp. 447-458.

[13] Erdani, Y. (2012). Developing Backward Chaining Algorithm of Inference Engine in Ternary Grid Expert System. *International Journal of Advanced Computer Science and Applications, Vol. 3, No. 9, 2012,* pp. 241-245.

[14] Poli, R., & Langdon, W. (2006). Backward-chaining evolutionary algorithms. *Artificial Intelligence 170,* pp. 953–982.

[15] Homeier, P., & Le, T. (2008). ECLIPS: An Extended CLIPS for Backward Chaining and Goal-Directed Reasoning. *Technical report.*

[16] Hell, M., Kulenovic, Z., & Antonic, R. (2005). Matrix calculation backward chaining in rule based expert system.*4th Research/Expert Conference with International Participation "QUALITY 2005",* pp. 163-170.

[17] Urbani, J., Van Harmelen, F., Schlobach, S., & Bal, H. (2011). QueryPIE: Backward reasoning for OWL Horst over very large knowledge bases. *Proc. 10th Int. Semantic Web Conf. (ISWC'11).*

[18] Nau, D., Smith, S., & Erol, K. (1998). Control Strategies in HTN Planning: Theory versus Practice. *AAAI-98/IAAI-98 Proceedings,* pp. 1127-1133.

[19] Harland, J., Pym, D., & Winikoff, M. (2000). Forward and Backward Chaining in Linear Logic (Extended Abstract). *Electronic Notes in Theoretical Computer Science 37.*

[20] Haemmerle, R. (2014). On Combining Backward and Forward Chaining in Constraint Logic Programming. *PPDP'14.*