New inference algorithms based on rules partition

Agnieszka Nowak-Brzezińska and Roman Simiński

Department of Computer Science, Institute of Computer Science, Silesian University, Poland http://ii.us.edu.pl

Abstract. This paper presents new inference algorithms based on rules partition. Optimisation relies on reducing the number of rules searched in each run of inference and reducing the number of unnecessary recursive calls and avoiding the extraction of too many new facts. The first part of the paper presents the definition of the model of knowledge base based on rules partition. The second part describes the modifications of both *forward* and *backward* inference algorithms using knowledge base with rules assigned to the groups given by the partition strategy. At the end, the paper consists of an example of knowledge base with the description of inference processes for this particular knowledge base.

Keywords: knowledge base, inference process, goal driven, data driven, rules partition

1 Introduction

Knowledge bases are still one of the most popular method of knowledge representation. They are constantly increasing in volume, thus the knowledge stored as a set of rules is getting progressively more complex. The ability to obtain large rules sets in easy and automatic way has got an enormous negative impact on the time of decision making — inference in large rules sets is time consuming process and the results of inference are in many cases hard to understand [1, 11, 7]. Effectiveness of both *forward* and *backward* inference algorithms is a significant problem when considering the large and complex rules bases, especially when we consider inference used in systems which work in the real-time environments and do not use a dialogue with user during inference.

This paper presents a new idea of partitioning of rule knowledge base and application of this idea in inference optimisation task. We introduce an approach, which assumes that the rule knowledge base is decomposed into the groups of rules, called *rules partitions*. Knowledge base in the form of rules partitions allows to improve the efficiency of inference algorithms. The groups of rules on which the knowledge base is partitioned let to reduce the search space for inference algorithms and to decrease the time needed to browse rules. Thus we may say that the knowledge base in the form of rules partition optimises inference processes. In this work we introduce two types of strategies for partitioning of knowledge base. First important strategy, divides the rules knowledge base into

groups of rules with similar premisses, the second divides the rules into groups of rules with the same conclusion. The first one is called *strategy based on similar premisses*. The second one is known as *decision strategy*. Both of the strategies will be used as a base for proposed modification of inference algorithms. The introduced partition strategies correspond to the research carried out so far by the authors on cluster analysis for the rules in the knowledge base [8–10] and the application of the concept of decision unit idea [12]. The proposed modification of knowledge base model is a kind of decomposition of knowledge base. Every created group of rules has its' representatives and during the inference processes the representatives of groups of rules are browsed instead of every rule one by one. Improving of the efficiency of forward inference is significant if the quality of groups' representatives is high enough. The modification of the classical backward inference algorithm is based on extracting information of internal rules dependencies. This information allows to perform only promising recursive calls of backward inference algorithm. Optimisation relies on reducing the number of rules searched for each run of inference and reducing the number of unnecessary recursive calls.

The first part of the work presents the rules partitioning approach. The second part consists of the description of interence algorithms modifications. At the end, the paper contains an example of knowledge base with the description of inference processes for this particular knowledge base.

2 Decomposition of rule knowledge bases

The proposed concept assumes division of knowledge base into coherent subgroups of rules. This section presents the basic concepts and notations of knowledge base partitioning, because this conception is a relatively new and it is not widely known. A significant part of this work contains detailed description of proposed approach.

2.1 Preliminary information

The rules-based knowledge base $\mathcal{R}$ consists of n rules: $r_1, \ldots, r_i, \ldots, r_n$ and is an unordered set. Each rule $r_i \in \mathcal{R}$ is stored as a Horn's clause: $r_i : p_1 \wedge p_2 \wedge \ldots \wedge p_m \rightarrow c$, where m is the number of literals[1] in conditional part of rule r_i ($m \geq 0$, $i = 1 \ldots n$) and c is the literal for the conclusion part of rule r_i. An attribute $a \in A$ may be a conclusion of rule r_i as well as a premise of it. For each rule $r_i \in \mathcal{R}$, the function $concl(r_i)$ returns the conclusion literal of rule r_i, whereas the function $cond(r_i)$ returns the set of conditional literals of rule r_i. The *set of facts* is denoted as $\mathcal{F} = \{f_1, f_2, \ldots f_s\}$, the facts are also stored as literals. The knowledge base $\mathcal{KB}$ is defined as a pair: $\mathcal{KB} = (\mathcal{R}, \mathcal{F})$.

[1] literals in the form of pairs of attribute and its value (a, v_i^a) (or equally $a = v_i^a$). Let A be a non empty finite set of conditional and conclusion (decision) attributes. For every attribute $a \in A$, the set V_a is called the set of values of attribute a.

2.2 Partition strategies

For every rules base $\mathcal{R}$ with n rules the number of possible subsets is 2^n. Any arbitrarily created subset of rules $R \in 2^{\mathcal{R}}$ is called *a group of rules* PR or *rules partition*. It is created by *partition strategy*, denoted by PS, which generates groups of rules PR: $PS(\mathcal{R}) \stackrel{\text{gen}}{\Rightarrow} PR$. Partition strategy PS for rules set $\mathcal{R}$ generates the partition of rules $PR \subseteq 2^{\mathcal{R}}$:$PR = \{R_1, R_2, \ldots, R_k\}$ where: k is the number of groups of rules formed by the partition PR, and R_j is j-th group of rules, $R \in 2^{\mathcal{R}}$ and $j = 1, \ldots, k$. Partition strategy PS is the general concept. It is possible to show a lot of different approaches for creating groups of rules. Each of them can be done with a different partition strategy. Every *simple partition strategy* involves the necessity of defining the *membership criteria* $mc : \mathcal{R} \times PR \rightarrow [0..1]$. We can define mc in a variety of ways, also we can consider specific form of this function ($mc : \mathcal{R} \times PR \rightarrow \{0, 1\}$), for example:

$$mc(r_i, R_j) = \begin{cases} 1 & \text{if } sim(r_i, R_j) \geq T \\ 0 & \text{in the opposite case} \end{cases} \tag{1}$$

where $sim(r_i, R_j)$ is defined in eq.3.

In general, the value 1 denotes the situation when rule $r_i \in \mathcal{R}$ is definitely included in the group R_j and 0 in the opposite case. The value between 0 and 1 denotes the degree of membership of rule r_i to the group R_j. It is possible to achieve many different partitions of rules. Generally, including *op*, as any operator from the set $[>, \geqslant, <, \leqslant, =, \neq]$, we can simply define partition PR as follows: $PR = \{R_j : R_j \in 2^{\mathcal{R}} \wedge \forall_{r_i \in R_j}\ mc(r_i, R_j)\ op\ T\}$, where value $mc(r_i, R_j)$ defines the membership of rule r_i to the group R_j which can be higher, higher or equal, equal, less, less or equal to the threshold value T ($0 \leq T \leq 1$).

The exceptional case of a simple strategy is the specific partition called further the *selection*. The *selection* divides the set of rules $\mathcal{R}$ into two subsets R and $\mathcal{R} - R$ in such a way, that all rules from R meet the membership criterion for some partition strategy PS, and all other rules do not meet such criterion. Partition $PR = \{R, \mathcal{R} - R\}$, which is the result of the *selection*, is important in practical terms. During the inference process, we search within the knowledge base one rule by one (which is time consuming) in order to find a set of relevant rules. *Selection* serves the same role, because it provides rules necessary to be activated during the inference process, but in a more efficient way.

There are two types of partition strategies: *simple* and *complex*. *Simple strategy* creates final partition PR by a single search of rules set $\mathcal{R}$ and by assigning each rule r_i to group R_j according to the value of the membership function $mc(r_i, R_j)$. The example of simple strategy is a selection of a pair of most similar rules or selection a group of rules to which a given rule is the most similar according to the value of threshold T and membership function mc. The strategy of creating *groups of similar rules* is the example of complex strategy because it usually multiply runs the simple strategy of finding two most similar rules or groups of rules. *Complex strategies* very often do not generate the final partition in a single step. Usually, they begin with a preliminary partition, which is modified further in accordance to the methods of partition. It is defined by

a combination of a few simple strategies, an *iteration* of one of them or the combination of both methods.

Partition strategy based on finding *groups of rules with similar premisses* is the example of complex strategy which uses simple strategy known as *strategy of choosing a pair of the most similar rules* many times. Finding the pair of most similar objects is a crucial step in every clustering algorithm. Thus the partition strategy which divides rules into the *groups of rules with similar premisses* has the result similar to the Agglomerative Hierarchical Clustering (AHC) algorithm but the hierarchy does not need to be saved [8, 9]. Only the result of the final partition is important. The process can be terminated when the similarity between merged clusters drops below given threshold (it was called the $mAHC$ algorithm in [6]). The result of partitioning the set of all rules from $\mathcal{R}$ using the $mAHC$ algorithm is k groups of rules $R_1, \ldots, R_l, \ldots, R_k$ and it is following:

$$R = \{R_1, \ldots, R_l, \ldots, R_k : \forall_{r_i, r_j \in R_l} sim(r_i, r_j) \geq T\}. \tag{2}$$

Each group R_l consists of rules for which mutual similarity is at least equal to T. To do this, at the beginning we have to use simple strategy forming $R_1, \ldots, R_k$ groups where every rule belongs to exactly one group. This strategy is used only once and is defined as: $R : \cup\{R_1 \cup \ldots \cup R_n : \forall_{R_i \in R_1, \ldots, R_n} |R_i| = 1 \wedge \forall_{R_i, R_j \in R_1, \ldots, R_n, R_i \neq R_j} R_i \cap R_j = \emptyset\}$. In the next step the strategy based on finding the most similar rules or groups of rules is multiply executed.

To determine the similarity between rules here the authors use the function $\mathcal{R} \times \mathcal{R} \rightarrow [0..1]$ defined as:

$$sim(r_i, r_j) = \frac{|cond(r_i) \cap cond(r_j)|}{|cond(r_i) \cup cond(r_j)|}. \tag{3}$$

The value of $sim(r_i, r_j)$ is equal to 1 if rules are formed by the same literals and 0 when they do not have any common literals. The more similar the rules are, the value of $sim(r_i, r_j)$ is closer to 1. It uses the similarity of conditional part of rules. However, we may also use the information about conclusions of rules while measuring the similarity of rules. This similarity measure works well with either rules, groups of rules, representatives of rules or set of facts. It is worth to mention that each group in created rules partition achieves its' representative called *Profile*. Instead of searching within the full structure of rules, only representatives of groups are compared. They are usually given by the set of common literals among all the literals belonging to all the rules forming a group of rules, a set of the most frequent literals or the set of distinctive ones. To find a relevant group of rules during the inference process the maximal value of similarity (see eq. 3) between the set of facts F (and/or hypothesis to be proven) and the representatives of each group of rules $Profile(R_j)$ is searched. Value of $sim(F, Profile(R_j))$ is equal to 0 if there is no such fact f_i (or hypothesis to prove) which is included in the representative of any group R_j. It is equal to 1 if all facts (and/or hypothesis) are included in the $Profile(R_j)$ of group R_j. The more facts with hypotesis included in the profile of group R_j the greater the

value of similarity. Exhaustive search of rules is done only within a given group. Details of inference process are given in the section 3.

Partition strategy for rules with the same conclusions creates the groups of rules from $\mathcal{R}$ by grouping rules with the same attribute in conclusions. It also can be called *decision oriented partition strategy*. The membership criterion for rule r_i and group R_j is given by the function mc defined in eq. 1. When we will use the *simple partition algorithm* (described below in the next section as *Alg01:createPartitions*) with the mc function defined in this way, we obtain *decision oriented partitions*. Each group of the rules generated by this algorithm may have the following form: $R = \{r \in \mathcal{R} : \forall_{r_i \in R}\ concl(r_i) = concl(r)\}$. The number of groups in the partition $k : 1 \leq k \leq n$ depends on the number of *different decisions* included in conclusions of such rules. When we distinguish different decision by the different conclusions appearing in the rules — we get one group for each conclusion. This partition strategy corresponds to the concept of the *decision units* presented in [9, 10]. In accordance to more general conception of rules partitions (presented here), decision units are just one of many possible partitioning strategy. In every group we have rules with the same conclusion: a fixed (a, v) pair. All rules, grouped within a rule set, take part in an inference process confirming the goal described by the particular attribute-value — for each $R \in PR$ the conclusion set $|Concl(R)| = 1$.

The attribute in knowledge base usually represents the particular *concept* or *property* from real word. The value of attribute expresses the determined *meaning* of the *concept* or *state* of the *property*. If we consider the specified (a, v) pair, we think about particular kind of concept or about particular property state. Similarly, in decision tables $DT = (U, A \cup \{d\})$ — considered in the Rough Set Theory — values of decision attributes d defines partition of objects from U into the *decision class* $CLASS_A(d) = \{X_A^1, X_A^2, \ldots X_A^{r(d)}\}$, where $r(d)$ is the cardinality of value set for attribute a and X_A^i is the i-th decision class of A [13]. Each group of rules from elementary partition is similar to i-th decision class X_A^i [14].

2.3 The partition algorithms

The input parameters for rules partition algorithm based on *simple strategy* are: knowledge base $\mathcal{R}$, mc function and the value of the threshold T. Output data is the partition PR. Time complexity of such algorithm is $O(n \cdot k)$, where $n = |\mathcal{R}|$, $k = |PR|$. For each rule $r \in \mathcal{R}$ we have to check whether the goal partition PR contains the group R with rule r (the value of the mc function has to be at least equal to T: $mc(r, R) \geqslant T$). If such a rule doesn't exist the given rule r becomes the seed of a new group which is added to the created partition PR. **Alg01: The simple partition algorithm**

Require: $\mathcal{R}$, mc, T;
Ensure: $PR = \{R_1, R_2, \ldots, R_k\}$;
procedure ***createPartitions***($\mathcal{R}$, **var** PR, mc, T)
var R, r;

begin
$PR = \{\}$;
for all $r \in \mathcal{R}$ **do**
if $\exists R \in PR : mc(r, R) \geqslant T$ **then**
$R = R \cup r$;
else
$R = \{r\}$;
$PR = PR \cup R$;
end if
end for
end procedure

The algorithm of creating the specific partition called *selection* is presented below. The input parameters are knowledge base $\mathcal{R}$, the function mc which defines the membership criterion and the threshold value T. Output data is the single group of rules R, which consists of rules from $\mathcal{R}$, that fulfill the condition of mc. In fact, each selection generates the partition which has two elements, but the second group is defined implicitly as $\mathcal{R} - R$. So the final partition of PR for the selection strategy is created as: $PR = \{R, \mathcal{R} - R\}$. Time complexity of this algorithm is $O(n)$, where $n = |\mathcal{R}|$.

Let us notice that presented algorithm does not have any knowledge about the specification of a given group of rules. It uses only the values of the membership function mc. If the parameter R is an empty set, the function of mc determines whether the rule r fulfills the condition given by the membership function for empty group R. **Alg02: Selection strategy algorithm**

Require: $\mathcal{R}$, mc, T;
Ensure: $R = \{r_1, r_2, \ldots, r_l\}$;
procedure *createSelection*($\mathcal{R}$, **var** R, mc, T)
var r;
begin
for all $r \in \mathcal{R}$ **do**
if $mc(r, R) \geqslant T$ **then**
$R = R \cup r$;
end if
end for
end procedure

An example of partition algorithm (different to selection strategy algorithm) is the algorithm based on clustering rules into groups of similar rules. For each rule it checks if a particular rule should be included in a given group. The result is a partition of k groups: $PR = \{R_1, R_2, \ldots, R_k\}$. This algorithm describes the AHC or $mAHC$ algorithms and it is presented as follows.

Alg03: Partition algorithm given by clustering

Require: $\mathcal{R}$, mc, T;
Ensure: $PR = \{R_1, R_2, \ldots, R_k\}$;
procedure *clustering*($\mathcal{R}$, **var** R, mc, T)
INPUT: R ;

repeat
carry out some processing
findTwoMostSimilarGroupsOfRules(R);
createNewGroup$(pOR,\ R\ )$;
until $\max\{sim(R_i, R_j)\} \geq T$
OUTPUT: R' ;

where R' denotes the created rules' partition. *findTwoMostSimilarGroupsOfRules*(R) is the procedure that compares groups' representatives and finds the two most similar. The *createNewGroup*(pOR, R) function adds the new, recently created group to the structure and reduces the size of the structure by removing the elements which formed the newly created group.

Partitions generated by the algorithms `Alg01, Alg02, Alg03` are *unique* and *complete* — each rule $r \in \mathcal{R}$ is considered, and qualified to one group R, that matches the criteria of *mc*. Presented modification of backward inference will utilise this features. For this reason algorithms which generates incomplete or non-unique partitions will not be presented in this work.

3 Inference in knowledge bases with partition of rules

The main goal of the decision support systems ($DSSs$) is to infer new knowledge from data stored in knowledge base. The inference is the process of proving (or falsifying) the hypothesis, knowing that the facts used in the process are true [1, 2]. The more rules in knowledge base, the greater the time needed to search withing a given knowledge base. A lot also depends on the number of facts given as input data. Choosing a proper methods of inference is very crucial and should be considered due to a goal of the given user of DSS. If the user wants to prove some hypothesis, the best choice is a *goal* driven inference process. When the user does not know what to prove, but has some facts given as input knowledge, the proper way is to use *data* driven inference process.

3.1 Forward inference

This type of inference is also called *data* or *facts driven*. Each rule is analysed whether all its' premisses are satisfied. If that is the case, the rule is activated and its' conclusions are added to the set of facts. The algorithm stops either where there are no more rules which can be activated or when the starting hypothesis is added to the set of facts [5–8]. Forward inference leads to a massive growth of new facts, but most of them are nonessential to prove the starting hypothesis and can slow down the inference process significantly. Unfortunately, when the DSS consists of more than 100 rules, the vast majority of time (over 90%) is consumed to find the rules which can be activated [2–4]. The inference process can be divided into three stages: matching, choosing and execution. At first, the premisses of each and every rule are analysed to match it to the current set of facts. Then the subset of the conflict set is chosen consisting of those rules, which are actually activated. At the end, previously selected rules are analysed

and the set of facts is updated. The matching phase usually takes the biggest amount of processing time during the inference. To reduce the time necessary to realise the matching phase, the authors propose to partition the rules into groups that match the facts set with minimal covering threshold. It is not necessary to build a full partition and to create the selection from all the rules which are relevant to set of facts with some minimal covering factor. Having the structure with rules' partition, the *forward* inference works as follows: in each iteration the representatives of each group are browsed and only a group with the highest similarity to the given set of facts is chosen for further analysis. This particular group is the browsed. Each rule belonging to this group is analysed and when the rules' premisses are fully covered by the set of facts, rule is activated. The conclusion of the activated rule is added to the set of facts.

The pseudocode of such inference process is following:

Require: $\mathcal{R}$, $\mathcal{F}$;
Ensure: $\mathcal{F}$;
  procedure ***forwardInference***($\mathcal{R}$, $\mathcal{F}$)
  var R;
  begin
  createFactsMatchingSelection($\mathcal{R}$, $\mathcal{F}$, R);
  while $R \neq \emptyset$ **do**
    applyRules(R, $\mathcal{F}$);
    excludeRules(R, $\mathcal{R}$);
    createFactsMatchingSelection($\mathcal{R}$, $\mathcal{F}$, R);
  end while
  end procedure

The algorithm uses temporary variable R, which is the set of rules that is the result of the previous selection. The procedure responsible for finding the most relevant group to the given set of facts (F) and thus - the rules to activate is *createFactsMatchingSelection*. It is responsible for creating the selection of rules that match the given condition defined by *mc*. The membership function *mc* checks whether the $sim(r_i, R_j)$ is greater or equal to T. Using the *createFacts-MatchingSelection* procedure reduces the time necessary to find relevant rules significantly. The original time complexity $O(n)$ is reduced to the $O(k)$ where $k << n$, k denotes the number of created groups. If the result of the selection is not empty ($R \neq \emptyset$) the procedure *applyRules* is commenced. It adds the new facts which are the conclusions of all rules activated from the set R to the set of facts. The *excludeRules* function temporarily removes activated rules from further searching. Afterwords, the $createFactsMatchingSelection$ procedure is called again. The whole presented process is repeated in *while* loop until the selection becomes empty.

It should now be clear that the most often used method of running the *mc* function is to calculate the similarity value between a given set of facts and a rule from the previously selected set R.

Among the advantages of the proposed modification of the classical forward inference process are: reducing the time necessary to search within the whole

knowledge base in order to find rules to activate and achieving additional information about fraction of rules that match the input knowledge (the set of facts).

3.2 Backward inference

In this section we present the backward inference algorithm, which use the conception of the *decision partitions* PR, created by the selection algorithm `Alg01`. The proposed modification utilises the following concepts: (i) the reduction of the search space by choosing only rules from particular rule group, according to a current structure of decision oriented rules partition and (ii) the estimation of the usefulness for each recursive call for sub-goals, only promising recursive calls of classical backward algorithm will be made.

Modified algorithm for backward inference will use the following input data: PR — the decision partition, $\mathcal{F}$ — the set of facts, g — the goal of the inference. The output data of the algorithm: $\mathcal{F}$ — the set of facts, including possible new facts obtained through inference, the function's result as boolean value, *true* if the goal g is in the set of facts: $g \in F$, *false* otherwise.

Alg04: Modified backward inference algorithm
function *backwardInference*(PR, g, **var** F) : **boolean**
begin
  if $g \in F$ **do return true else**
    $A \leftarrow \emptyset$, $truePremise \leftarrow$ **false**, **select** $R \in PR$ **where** $g \in Concl(R)$
    while $\neg truePremise \wedge \{R - A\} \neq \emptyset$ **do**
      select $r \in \{R - A\}$ **according to the current selection strategy**
      forall $w \in cond(r)$ **do**
        $truePremise \leftarrow$ ($w \in F$)
        if $\neg truePremise \wedge w \in In_C(R)$ **then**
          $truePremise \leftarrow$ *backwardInference*(PR, w, F)
        elseif $\neg truePremise$ **then**
            $truePremise \leftarrow$ *environmentConfirmsFact*(w)
        elseif $\neg truePremise$ **then**
            break
        endif
      endfor
      if $\neg truePremise$ **then** $A = A \cup \{r\}$ **endif**
    endwhile
  endif
  if $truePremise$ **then** $F = F \cup \{g\}$ **endif**
  return $truePremise$
end function

The proposed modification of backward inference algorithm utilises the information from decision partitions of rules to reduce the number of unnecessary

recursive calls and the number of rules searched for in each run of the inference. Only promising groups of rules are selected for further processing (**select** $R \in PR$ **where** $g \in Concl(R)$), where $Concl(R)$ is the *set of conclusions* for the group of rule R, containing literals appearing in the conclusion parts of the rules r from R. Only the selected subset of the whole rules of $(R - A)$ is processed in each iteration. Finally, only the promising recursive calls are made ($w \in In_C(R)$). $In_C(R)$ denotes *connected inputs of the rules group*, defined in the following way: $In_C(R) = \{(a, v) \in Cond(R) : \exists_{r \in \mathcal{R}} (a, v) = concl(r)\}$, where $Cond(R)$ is the *set of conditions* for the group of rule R, containing literals appearing in the conditional parts of the rules r from R. In each iteration the set R contains only proper rules matching to the currently considered goal, it completely eliminates the necessity of searching for rules with conclusion matching to the inference goal, it is not necessary to search within the whole set of rules $\mathcal{R}$ — this information is simply stored in the decision-units and does not have to be generated.

3.3 An example of knowledge base and inference processes

To illustrate the conception of partitioning and exploring the knowledge base, let us consider an example of knowledge base $\mathcal{R}$ which consists of following rules.

$$r_1 : (a,1) \wedge (b,1) \rightarrow (c,1) \; r_2 : (a,1) \wedge (b,2) \rightarrow (c,2) \; r_3 : (a,1) \wedge (b,3) \rightarrow (c,1)$$
$$r_4 : (b,3) \wedge (d,3) \rightarrow (e,1) \; r_5 : (b,3) \wedge (d,2) \rightarrow (e,1) \; r_6 : (b,3) \rightarrow (e,2)$$
$$r_7 : (d,4) \rightarrow (f,1) \qquad r_8 : (d,4) \wedge (g,1) \rightarrow (f,1) \; r_9 : (a,1) \rightarrow (d,4)$$

According to the knowledge base presented below and the set of facts $F = \{(a,1)\}$, in case of *goal driven* inference algorithm, for confirmation of the main goal of the inference described by literal $(f,1)$, it is necessary to find rules matching to the goal. Next — if such rules exist — we have to select the rule to be activated. In our simple example we select a group R_1 with rules r_7 and r_8 from which we are choosing r_7 as the rule with higher similarity value. Now we have to confirm the truth of the premise $(d,4)$. Because it doesn't appear in the facts set, it becomes a new sub-goal of inference and inference algorithm runs recursively. The classic version of the algorithm does not known whether the call is promising — i.e. it is unknown if there is a rule that proves the new goal of inference. For goal $(d,4)$ we use recursive call for $(a,1)$. It is a fact and when it is confirmed the other subgoal will also be confirmed $(d,4)$ as well as the main hypothesis $(f,1)$.

According to the rules' partition based on finding similar rules and the set of facts $\{(g,1),(d,4)\}$ the process of data driven inference algorithm goes as follows. Assuming that chosen partition strategy returns the following structure of knowledge base:

$$R_1 = \{r_7, r_8\} \; R_2 = \{r_1, r_3\} \; R_3 = \{r_4, r_5\} \; R_4 = \{r_2\} \; R_5 = \{r_6\} \; R_6 = \{r_9\}$$

we check the similarity[2] between the set of facts F and the representative of each group:

$$sim(F, Profile(R_1)) = \frac{2}{3},\ sim(F, Profile(R_2)) = 0,\ sim(F, Profile(R_3)) = 0,$$
$$sim(F, Profile(R_4)) = 0,\ sim(F, Profile(R_5)) = 0,\ sim(F, Profile(R_6)) = 0.$$

Next we choose the most similar group to the set of facts (which is R_1) and we check the similarity between every rule in this particular group:

$$sim(r_7, F) = \frac{1}{3}, sim(r_8, F) = \frac{2}{3}.$$

We activate rules with all known premisses (in our case it is only one rule r_8). Then we include new fact to F: $F = F \cup \{(f, 1)\}$. The new fact is the conclusion of activated rule r_8. Of course we block rule r_8 to make sure that it will not be analyzed later because it would only increase the time of inference process without gaining additional knowledge. We also block the whole group if all its' rules are already blocked. The result of the inference process is eventually given as a set of facts $F = \{(g, 1), (d, 4), (f, 1)\}$. If the rules partition is the result of using the partition strategy based on clustering algorithm, it is necessary to search the groups of rules using similarity measure based on searching using *profiles* of groups. We may use any algorithms for rules clustering but the most efficient one is, in authors' opinion, the AHC or $mAHC$ algorithm presented in [8, 9]. Therefore, it is quite easy to find a group which would be suitable for inference process. The time complexity of this operation is $O(\log n)$ where n is the number of elements grouped during the clustering.

4 Summary

We have introduced the approach, which assumes that the rule knowledge base is decomposed into the groups of rules, called *rules partitions*. In our opinion each knowledge base contains enough information, which allows to improve the efficiency of the classic inference algorithms. We introduce partitions of rules as the source of such information. Modification of the classical inference algorithms is based on information extracted from the rules of groups generated by the proposed, two partitioning strategy.

The proposed modification of forward algorithm consists of the reduction the time necessary to realise the matching phase of inference by choosing only the rules from particular rule group, which match the facts set with minimal covering threshold. The proposed modification of backward algorithm consists of the reduction of the search space by choosing only the rules from particular rule group, according to a current structure of decision oriented rules partition and the estimation of the usefulness for each recursive call for sub-goals. Therefore,

[2] the value for a pair F and $Profile(R_1)$ is equal to $\frac{2}{3}$ because there are two common literals $((g, 1), (d, 4))$ both in the set of facts and the representative of a group $R_1 = \{(g, 1), (d, 4), (f, 1)\}$.

only necessary rules are searched and only promising recursive call of the classical backward algorithm will made.

The concept of rules partitions is the new proposition, based on the previous research concerning on rules clusters [9] and decision units [12]. Proposition of the modified inference algorithm is the continuation of the previous research on optimisation of the inference in the knowledge based systems [8] .

Acknowledgements

This work is a part of the project ,,Exploration of rule knowledge bases" founded by the Polish National Science Centre (NCN: 2011/03/D/ST6/03027).

References

1. R. Akerkar and P. Sajja: Knowledge-Based Systems, Jones & Bartlett Learning, (2010)
2. D. Batory: The LEAPS Algorithms, `http://reports-archive.adm.cs.cmu.edu/anon/1995/CMU-CS-95-113.pdf`, (1995)
3. C. L. Forgy: On the efficient implementation of production systems, Carnegie-Mellon University, (1979)
4. C. L. Forgy: Rete: A Fast Algorithm for the Many Pattern. Many Object Pattern Match Problem, Artificial Intelligence, (1981)
5. J. P. Ignizio: An introduction to Expert Systems, McGraw-Hill, (1991)
6. D. P. Miranker: TREAT: A better match algorithm for AI Production Systems, Department of Computer Sciences, University of Texas at Austin,(1987)
7. D. Heckerman, An Empirical Comparison of Three Inference Methods, CoRR, 2013, http://arxiv.org/abs/1304.2357, http://dblp.uni-trier.de.
8. Nowak-Brzezińska, T. Jach, A., R. Simiński, T. Xieski: Towards a practical approach to discover internal dependencies in rule-based knowledge bases, Lecture Notes in Computer Science, LNCS 6954, Springer, 232-237, (2011)
9. A. Nowak-Brzezińska, A. Wakulicz-Deja: The way of rules representation in composited knowledge bases, Advanced In Intelligent and Soft Computing, Man - Machine Interactions, Springer-Verlag, 175-182, (2009)
10. A. Nowak-Brzezińska, R. Simiński: Knowledge mining approach for optimization of inference processes in rule knowledge bases, Lecture Notes in Computer Science, vol. 7567, Springer Berlin Heidelberg, (534-537), (2012)
11. W. Pedrycz: Knowledge-Based Clustering: From Data to Information Granules, Willey, (2005)
12. R. Siminski: Extraction of Rules Dependencies for Optimization of Backward Inference Algorithm, Beyond Databases, Architectures, and Structures, vol. 424, Communications in Computer and Information Science, Springer International Publishing, 191-200, (2014)
13. A. Skowron, J. Komorowski, Z. Pawlak, L. Polkowski: Handbook of Data Mining and Knowledge Discovery, Oxford University Press, Inc., (2002)
14. A. Skowron, Z. Suraj: Rough Sets and Intelligent Systems - Professor Zdzislaw Pawlak in Memoriam, Intelligent Systems Reference Library, Vol. 42, Springer Verlag, (2013)