

Rough Petri Net Model (RPNM) For knowledge Representation, Rules Generation and Reasoning

Hala S. Own

¹*National Research Institute of Astronomical and Geophysics, Helwan, Egypt*

E-mail: h_own@hotmail.com

halaown@gmail.com

Abstract

Rough Petri nets model (RPNM) for knowledge representation, rule generation, and reasoning is presented in this paper. An algorithm for verifying the consistency of a rough knowledge base through a set of reduction rules that preserve the properties of the RPNM is presented. In addition and based on the RPNM, an efficient algorithm is proposed to perform rough reasoning automatically.

Keyword

Reasoning, Petri net, rough sets, rule generation, knowledge-based system

1. Introduction

Knowledge representation and reasoning (KRR) is the study of how knowledge about the real world domain can be represented and what kinds of reasoning can be done with that knowledge [6,13]. Many of the broad topics within artificial intelligence and computer science in general such as reasoning, knowledge engineering, data acquisition, search techniques, and human-computer interaction, rely on representations of knowledge. The quality and design of these knowledge structures can determine if a particular endeavor will succeed and to what degree. Petri Nets (PNs) [3, 12,16] have ability to represent and analyze in an easy and visual way concurrency and synchronization phenomena, like concurrent evolutions, where various processes that evolve simultaneously are partially independent. Furthermore, PN approach can be easily combined with other techniques such as object-oriented programming, fuzzy theory, neural networks, rough sets, etc. [1,2,7].

With this observation we have chosen the rough Petri net as an alternative modeling and reasoning analysis

[4]. Knowledge base for rough set processing is stored as a table containing conditional and decision attributes. The table represents given knowledge in form of IF-THEN rules. However, some elements of knowledge can be uncertain and inconsistent. In such a case, decision table become inconsistent and some rules become uncertain, a RPN model will used in this paper to represent the rough production rules and to perform rough reasoning automatically

The paper is organized as follows: Section 2 gives a brief introduction of some fundamental concept of rough set. A formal description of the introduced RPNM is discussed in Section 3. In section 4, an attribute and rule reduction algorithms in RPNM are discussed. A RPNM verification methodology is described in section 5. A rough reasoning algorithm is introduced in section 6. Finally a conclusion and future work will be discussed in section 7.

2. Rough Sets

This section gives a brief overview of some fundamental concepts and features of rough set theory that are important to an understanding of the rough Petri net model given in this paper.

2.1 Preliminaries

Basically, rough set theory [13,14] deals with the approximation of sets that are difficult to describe with the available information. In a medical application, a set of interest could be the set of patients with a certain disease or outcome. In rough sets theory, the data is collected in a table, called decision table. Rows of the decision table correspond to objects, and columns correspond to attributes. In the data set, we assume we are given a set of examples with a class label to indicate the class to which each example belongs. We

call the class label the decision attributes, the rest of the attributes the condition attributes. Rough sets theory defines three regions based on the equivalent classes induced by the attribute values Lower approximation, upper approximation and boundary. Lower approximation contains all the objects which are classified surely based on the data collected, and upper approximation contains all the objects which can be classified probably, while the boundary is the difference between the upper approximation and the lower approximation. So, we can define a rough set as any set defined through its lower and upper approximations.

On the other hand, indiscernibility notion is fundamental to rough set theory. Informally, two objects in a decision table are indiscernible if one cannot distinguish between them on the basis of a given set of attributes. Hence, indiscernibility is a function of the set of attributes under consideration. For each set of attributes we can thus define a binary indiscernibility relation, which is a collection of pairs of objects that are indiscernible to each other. An indiscernibility relation partitions the set of cases or objects into a number of equivalence classes. An equivalence class of a particular object is simply the collection of objects that are indiscernible to the object in question. Some formal definitions of the rough sets are given as follows:

An information system $S = (U, A, V, f)$, where U is a finite set of objects, $U = \{x_1, x_2, \dots, x_n\}$, A is a finite set of attributes, the attribute in A is further classified into two disjoint subsets (C, D) , where C is a condition attributes and D is a decision attribute such that $A = C \cup D$, $V = \bigcup_{p \in A} V_p$ and V_p is a domain of attribute p and $f: U \times A \rightarrow V$ is a total function such that $f(x, q) \in V_q$ for every $q \in A, x_i \in U$.

The lower and upper approximations of a set $P \subseteq U$, are defined by equations (1) and (2), respectively.

$$\underline{PY} = \bigcup \{x : x \in U / IND(P), X \subseteq Y\} \quad (1)$$

$$\overline{PY} = \bigcup \{x : x \in U / IND(P), X \cup Y \neq \{\}\} \quad (2)$$

Assuming Q_1 and Q_2 are equivalence relations in U , the important concept called positive region $POS_{Q_1}(Q_2)$ is defined as:

$$POS_{Q_1}(Q_2) = \bigcup_{X \in Q_2} \underline{Q_1}X \quad (3)$$

A positive region $POS_{Q_1}(Q_2)$ contains all patterns in U that can be classified in attribute set Q_2 using the information in attribute set Q_1 .

The mathematical machinery of rough sets is derived from the assumption that granularity can be expressed by partitions and their associated equivalence relations on the set of objects, it is called indiscernibility relations. With respect to a given $q \in \Omega$, the functions partition the universe into a set of pair wise disjoint subsets of U :

$$R_q = \{x : x \in U \wedge f(x, q) = f(x_0, q) \forall x_0 \in U\} \quad (4)$$

Assume a subset of the set of attributes, $P \subseteq C$, two samples x and y in U are indiscernible with respect to P if and only if $f(x, q) = f(y, q) \forall q \in P$. The indiscernibility relation for all $P \subseteq C$ is written as $Ind(P)$ While $U / Ind(P)$ is used to denote the partition of U given $Ind(P)$ and is calculated as follows:

$$U / Ind(P) = \otimes \{q \in P : U / Ind(P)(\{q\})\}, \quad (5)$$

Where

$$A \otimes B = \{x \cap y : \forall q \in A, \forall y \in B, x \cap y \neq \{\}\}. \quad (6)$$

The degree of dependency $\gamma(C, D)$ between the condition attributes C and the decision attributes D is defined as:

$$\gamma(C, D) = \frac{Card(POS(C, D))}{Card(U)} \quad (7)$$

Where $Card$ denotes the cardinality of set S .

The degree of dependency provides a measure of how important condition attributes C is in mapping the dataset examples into decisional attributes D .

If $\gamma(C, D) = 0$, then classification D is independent of the attributes in C , hence the decision attributes are of no use to this classification. If $\gamma(C, D) = 1$, then D is completely dependent on C , hence the attributes are indispensable. Values $0 < \gamma(C, D) < 1$ denote partial dependency, which shows that only some of the attributes in P may be useful, or that the dataset was flawed to begin with, in addition, the complement of $\gamma(C, D)$ gives a measure of the contradictions in the selected subset of the dataset.

In an information system there often exists some condition attributes that do not provide any additional information about the objects in U . So, we should remove those attributes since the complexity and cost of the decision process can be reduced if those condition attributes are eliminated [13,14]. The rough set theory provides tools deal with this problem. Core and reduct are two important concepts in rough sets theory. A reduct is a central part of an information system which can discern all objects discernible by the original information system. Core is the common parts of all the reducts. The reduct definition is given as follows:

- Given a classification task mapping a set of conditional attributes C to a set of decision attribute D , a reduct is defined as any $R \subseteq C$, such that $\gamma(C, D) = \gamma(R, D)$.
- Given a classification task mapping a set of variables C to a set of labeling D a reduct set is defined with respect to the power set $\Pr(C)$ as the set $R \subseteq \Pr(C)$ such that $R = \{A \in \Pr(C) : \gamma(A, D) = \gamma(C, D)\}$. That is, the reduct set is the set of all possible reduct of the equivalence relation denoted by conditional and decisional attributes.
- Given a classification task mapping a set of conditional attributes C , to a set of decisional attribute D , and R is the reduct set for this

problem space, a minimal reduct is defined as any reduct R such that $|R| \leq |K|, \forall K \in R$.

The set of all attribute belonging to the intersection of all reduct of C with respect to D , is called the core of C , denoted as $Core(C, D)$

2.2 Decision rules

The major process of discovering knowledge in database is the extraction of rules from decision system. Rules are generated from reducts where reducts are transformed into rules by binding the condition attribute values of the object class from which the object reduct is originated to the corresponding attribute. A decision rough rule takes the form : $IF \beta THEN \psi$: (with $Confidence = w_i$),

Where

- β Represents the antecedent part of a given rule; it is define the rough region in the input space
- ψ Represents the consequent part of a given rule; it is define the rough region in the output space
- w_i Represents the confidence value that is associated with the constructed rules.

3. Rough Petri Net Model (RPN)

This section introduces the Rough Petri net to model the knowledge representation and verification. We assume that a decision table represents the knowledge base for an expert system. We extract rules from a decision system that represents the relationship between the values of conditional attributes and the decision attributes using the rough set methods. On the base of a set of all rules extracted from a given decision system we construct a Rough Net as a reasoning model.

3.1 RPN model: definition

Rough Petri Net (RPN) Model can be defined as a 10 – tuples as follows:

$$RPNM = (P, T, F, \eta, I, R_{in}, R_{out}, Cer, Cov, Str).$$

Where

- $P = \{p_1, p_2, \dots, p_n\}$ is the finite set of places
- $T = \{t_1, t_2, \dots, t_n\}$ is the finite set of transitions
- $F \subseteq (PxT) \cup (TxP)$ is the set of arcs
 $\eta = \{\eta_1, \eta_2, \dots, \eta_n\}$: where $\eta_i \in [0,1]$
represent the degree of dependency of a attribute represented by place P_i with respect to the class decision.
 $I : P \rightarrow \{0,1\}$ Is a rough marking function it represents the distribution of token over places. The rough marking function illustrates the degree of completion of the rough event as a result of the processes of the rough reasoning rules.
 $I_i, i = 1,2,3, \dots, n$ where $I_i = 1$ if there is a token in P_i , $I_i = 0$ if P_i is not marked. The initial marking is denoted by I^0 .
- R_{in} is a mapping $P \times T \rightarrow \{0,1\}$
Corresponding to the set of directed arcs from proposition to rules.
 $R_{in}(P_i, T_j) = 1$ if there is a directed arc from P_i to T_j for $i = 1,2, \dots, n$ and $j = 1,2, \dots, n$
and $R_{in}(P_i, T_j) = 0$ if there is no directed arc from P_i to T_j .
- R_{out} is a mapping $P \times T \rightarrow \{0,1\}$
corresponding to the set of directed arcs from rule to proposition:
 $R_{out}(P_i, T_j) = 1$ if there is a directed arc from T_j to P_i for $i = 1,2, \dots, n$
and $j = 1,2, \dots, n$
and $R_{out}(P_i, T_j) = 0$ if there is no directed arc from T_j to P_i .
- $cer : T \rightarrow [0,1]$, is a certainly factor.
- $cov : T \rightarrow [0,1]$, is a coverage factor.
- $str : T \rightarrow [0,1]$, is a strength factor.

An essential feature of RPN model is that it can be executed; here we will introduce the enabling and firing rule of RPN model. A rule T_i is enabled means that the condition of the rule has been satisfied and the associated rule is activated. This condition is fulfilled if and only if each input proposition is marked by token. Enabling rule T_i at a marking I , resulting a new marking I' such that:

- A token will be added to each output place of the enabling transition.
- For each $P_i \in R_{out}(T_i)$ $\eta(P_i) = \max(\eta_j)$
, $j = 1,2, \dots, k$ such that $P_j \in R_{in}(T_j)$
- $Cer(T_i) = cer(T_i) * \eta(P_i)$ where $\eta(P_i) = \max(\eta_j)$
, $j = 1,2, \dots, k$ such that $P_j \in R_{in}(T_j)$
- $Cov(T_i) = Cov(T_i) * \eta(P_i)$ where $\eta(P_i) = \max(\eta_j)$
, $j = 1,2, \dots, k$ such that $P_j \in R_{in}(T_j)$
- $Str(T_i) = str(T_i) * \eta(P_i)$ where $\eta(P_i) = \max(\eta_j)$
, $j = 1,2, \dots, k$ such that $P_j \in R_{in}(T_j)$

The rough Petri net model is given in the following steps:

The RPNM

Input: attributes for rules

Processing:

- **Step1:** Represent the attributes of the rules as a set of places.
- **Step2:** Construct a set of input object distribution transitions, where these transitions are used to distribute the desired input attributes to the common propositions of the second antecedent part of the rule.
- **Step3:** Calculate the firing strength for each rule according to execution rules of RPNM.
- **Step 4:** determine the winning rule that has highest confidence among the activated rules.

Output: Wining rules

3.2 Worked Example

Let us now consider an example of decision table shown in Table (1). Where disease, age and sex and test are condition attributes, whereas test is the

decision attribute. Table (2) shows the calculated strength, certainly and coverage factors for the decision Table (1). The generated rough rules are given in Table (3).

Table 1. Decision table

Fact	Disease	age	Sex	Test	Support
1	Yes	Old	Man	+	400
2	Yes	Middle	Woman	+	80
3	No	Old	Man	-	100
4	Yes	Old	Man	-	40
5	No	Young	Woman	-	220
6	Yes	Middle	Woman	-	60

Table 2. Strength, certainly and coverage factors

Fact	Strength (Str)	Certainly (Ser)	Coverage (Cov)
1	0.44	0.92	0.83
2	0.09	0.56	0.17
3	0.11	1.00	0.24
4	0.04	0.08	0.10
5	0.24	1.00	0.52
6	0.07	0.44	0.14

Table 3. Rough rules

NO	Rules
1	Disease(yes) AND Age (old) => Test(+)
2	Disease(yes) AND Age (middle) => Test(+)
3	Disease(no) => Test(-)
4	Disease(yes) AND Age (old) => Test(-)
5	Disease(yes) AND Age (middle) => Test(-)

Now we will illustrate how RPNM model can be used to model the structure properties and describe the dynamic behavior of the following rough rule.

$$\text{Disease (yes) AND Age (old) } \Rightarrow \text{Test (+)} \quad (\text{A})$$

Figure (1) shows the RPN model of the given rough rule (A). The description of places and transitions of the RPN model are give as follows:

- P1: represent the antecedent proposition 1 with its degree of dependency η_1
- P2: represent the antecedent proposition 2 with its degree of dependency η_2
- T1: represents the min composition operation of the antecedent propositions of the rule with coverage, strength, and certainly

factors $Cov(t_1), Str(t_1)$ and $Cer(t_1)$ respectively.

- P3: represent the degree of truth of test is +.

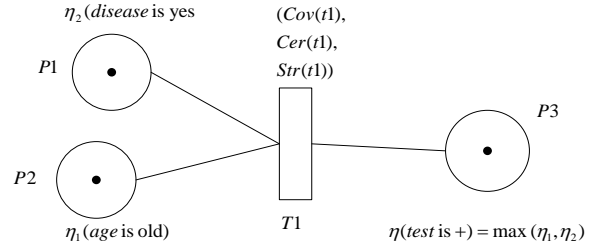


Figure 1. RPN model of the rough rule

$\ll \text{Disease (yes) AND Age (old) } \Rightarrow \text{Test (+)} \gg$

4. Attribute and Rule Reduction in RPNM

An algorithm to finding a subset of attributes/rule (reduct) with the same classificatory power as the entire set of attributes in an information system is briefly described in this section.

Attribute Reduction Algorithm

The place P_i can be suppressed (deleted with its arcs) according to the following rules [5]:

- **Rule 1-** the output transition of place P_i have no other input places than P_i and P_i is pure (i.e. there are no transition which are both input and output transition of P_i).
- **Rule 2-** If the place P_i is implicit i.e. the marking in this place never forms an obstacle to the firing of its output transition.
- **Rule 3-** The marking of P_i can be deduced from the marking of other places by relation
$$M(p_i) = \sum_{k \neq i} (\alpha_k M(p_k)) + \beta$$
 where α_k is a rational number positive or null, and β is a rational number.

Rule Reduction algorithm

A transition T_i and its input and output arcs can be suppressed according to the following rules:

- Rule 1- the set of input place of a transition T_i is identical to the set of its Output places (i.e. if $\exists T_k \neq T_i$ such that $\text{Post}(P_i, T_k) \geq \text{pre}(P_i, T_i)$ for every place $P_i \in R_{in}(T_i)$).
- Rule 2- If $\exists T_k$ such that $R_{in}(T_k) = R_{in}(T_k)$ and $R_{out}(T_{ki}) = R_{out}(T_{kj})$.
- Rule 3- If T_i is a self loop transition (i.e. $R_{in}(T_i) = P_i = R_{out}(T_i)$). Suppressing T_i includes first to delete its input and output arc to and from P_i , and delete the transition T_i if it is isolated.

5. RPNM Verification Methodology

In this section we will use RPNM as knowledge representation formalism where structural and behavioral properties of the net can be used to verify the knowledge base integrity. In rough set environment, the issues of concern when talking about integrity checking are the definition of concepts such as inconsistency, redundancy and completeness. Petri net provide a tool to investigate all the properties of the net which is reachability tree [16]. Reachability tree is made up of nodes which correspond to the reachable markings and each directed edge represents transition firing and connect one node to another resulting in the passing from one marking to another. Here an algorithm to generate the reachability tree of RPNM will be introduced.

Generate the reachability tree of RPNM Algorithm

Input: a vector V of nodes denoted the initial marking.

Processing:

Step-1: Label the initial marking M_0 as the root

Step-2: For all enabled transition

$$t_i, i = 1, 2, 3, \dots, n, n =$$

number of enabled transitions under the current marking M

Step-3: Loop:

3.1 Generate a new marking and update V according to the new marking, and add an edge labeled by $T = \{t_1, t_2, \dots, t_n\}$ from the original node to that node.

3.2 If there is a marking $M_j = M_i$ on the path from M to M_i then M_i has no successor.

3.3 If there is a marking M_i on the path from M to M_i such that $M_i > M_j$, then w is placed for each of the components greater than the component M_i . This component w remain with no change for any firing transition.

3.4 If the new marking $M' = M$, then the set of transitions do not change the current marking. We add an edge labeled T from the current node to itself.

3.4 Go back to step 2.

Output: reachability tree

Figure (2) shows the rough petri net model of the rough rules given in Table (3). Figure (3) shows the reduction of the three rules given above.

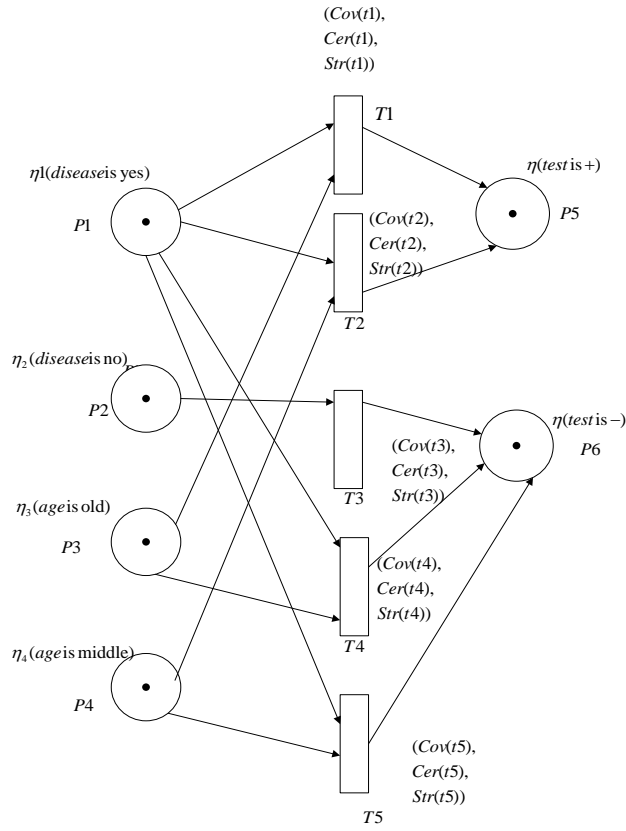


Figure 2. RPNM model of rough rule given in table 3.

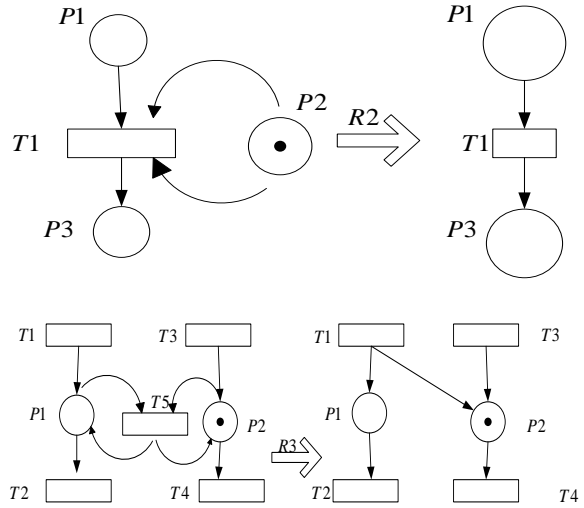
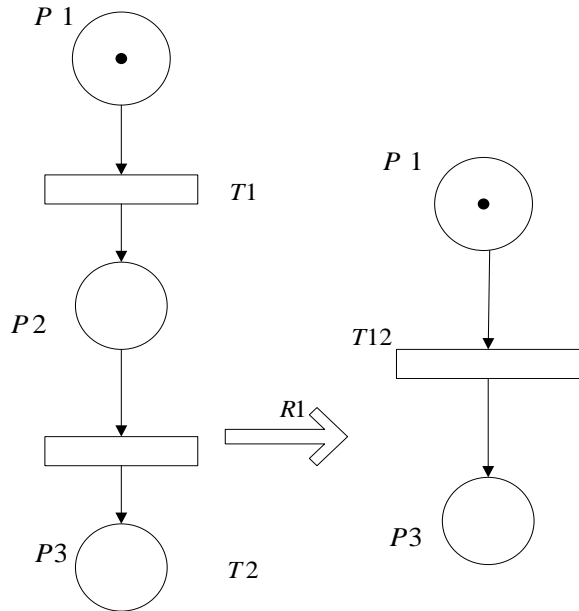


Figure 3. Examples of the 3 rules reduction

6. Rule Verification and Reasoning Algorithm

We will use RPNM reachability tree to detect structure errors including incompleteness, conflict and redundancy. These types of errors represent the important errors that occur in the practical rough set systems [8,9,10,11,15].

- **Redundancy rule:** This error occurs when both the antecedent and consequent parts of the rules are equivalent [14]. We can detect redundancy from the reachability tree. When the marking M' is the same as M , the transition firing of marking M' is redundant to the previous transition.
- **Conflict rules:** Conflicting rules have equivalent antecedents and conflicting consequences. We can detect it if two places in the marking vector M corresponding to two contradictory attributes are both has ω value. Then the rule resulting in the second ω is inconsistent with the rule resulting in the first ω .
- **Incompleteness Rule:** Zeros in the terminal node of the reachability graph corresponding to decision indicate missing rules in the rule base.

6.1 Worked Example

To help understanding these definitions, we will introduce the following inconsistency form via an illustrated example. Table (4) considers the

representation a classification of banana. Assume that we generate the following rules from table (4).

- R1: If Size is 10 AND color is yellow then Class is Fruit
- R2: if Size is 10 AND color is yellow then Class is Juice
- R3: if Size is 10 AND color is brown then Class is Fruit
- R4: If Size is 12 AND Smell is good then Class is Juice
- R5: If Size is 12 AND Smell is bad Then Class is Juice

Table 4. representation a classification of banana

#	Size	Smell	color	Filed#	Class
X1	10	Good	Yellow	1	Fruit
X2	10	Good	Yellow	1	Juice
X3	12	Good	Yellow	1	Juice
X4	5	Bad	Brown	1	Juice
X5	12	Good	Yellow	1	Juice
X6	10	Bad	Black	2	Fruit
X7	14	Bad	Black	1	Fruit
X8	11	Good	Brown	2	Fruit
X9	12	Bad	Yellow	1	Juice

As an example of indiscernible, R4 and R5 is unnecessary because it can not affect the conclusions of these rules. These rules can be reduced into the following rule:

- R6: If Size is 12 Then Class is Juice.

As an example of the conflicts, R1 and R2 succeed in the same situation but produce conflicting results.

Based on the model and execution rules of RPNM, we can study the reasoning algorithm to obtain the final reasoning results from the initial truth degree of propositions.

Reasoning Algorithm

Input: Decision table $\Gamma = \langle U, \Omega, V_q, f_q \rangle_{q \in \Omega}$

Processing:

- **Step 1:** build RPNM for the generated rule.
- **Step 2:** while there is an enabling transition loop
 - Apply firing execution rule these step generate a new marking M_i
 - Check if $M_i = M_{i-1}$, if true go to step 3, otherwise go to step 2

- **Step 3:** Apply attribute and rule reduction. Drive a reduct model
- **Step 4:** Verify the model and drive a verified model by:
 - Remove the inconsistent rules
 - Remove the conflict rules
 - Remove redundant rules

Output: Reasoning model corresponding to the decision system

7. Conclusion and Future Work

Rough Petri nets model (RPNM) for knowledge representation, rule generation, and reasoning have presented in this paper. Based on the rough net model, an algorithm was proposed to perform rule verification and reasoning automatically. It can determine whether an antecedent-consequence relationship exists between two propositions. The formal description of the model and the rule verification algorithm are shown in detail with examples. In the future work an investigation of using the RPNM with the proposed reasoning algorithm to predict a class decision in some medical applications

8. References

- [1] Brauer W., Reisig W. and Rozenberg, G. Eds., "Petri Nets: Applications and Relationships to Other Models of Concurrency", Berlin: Springer-Verlag, 1987.
- [2] Bugarin A. J. and Barro, S. "Fuzzy reasoning supported by Petri nets", IEEE Trans. Fuzzy Syst., vol. 2, pp. 135-149, Apr. 1994.
- [3] Fryc B., Pancerz K. and Suraj Z. "Approximate Petri Nets for Rule-Based Decision Making", In: Proceedings of Rough Sets and Current Trends in Computing: 4th International Conference, RSCTC 2004, Uppsala, Sweden, June 1-5, 2004, pages 733-742. Volume 3066 of Lecture Notes in Computer, 2004.
- [4] Giordana A. and Saitta L. "Modeling production rules by means of predicate transition networks", Information Sciences: an International Journal, v.35 n.1, p.1-41, March 1985
- [5] Hayes-Roth F. "Rule-based systems", Communications of the ACM, v.28 n.9, p.921-932, Sept. 1985 .
- [6] Jouni Järvinen, "Knowledge Representation and Rough Sets", citeseer.ist.psu.edu/434302.html, 1999
- [7] Li X., Yu W. and F. L. Rosano, "Dynamic knowledge inference and learning under adaptive Fuzzy Petri net

framework”, IEEE Syst., Man, Cybern, vol. 30, pp.442-450. Nov. 2000.

[8] G.Looney C. and Alfize A. R. “Rule-based reasoning as boolean transformations”, IEEE Transactions on Systems, Man and Cybernetics, v.17 n.6, p.1077-1082, Nov./Dec. 1987

[9] Murata T. and Matsuyama K., “Inconsistency check of a set of clauses using Petri net reductions”, Franklin Institute, vol. 325, no. 1, pp. 73-93, 1988.

[10] Nguyen T. A., “Verifying consistency of production systems,” in Proc. Third IEEE Conf. Artificial Intell. Appl., Orlando, FL, Feb. 1987, pp. 4-8.

[11] Nguyen T. A., Perkins W. A., Laffey T. J., and Pecora D., “Checking expert system knowledge bases for consistency and completeness,” in Proc. Ninth Int. Joint Conf. Artificial Intell. Los Angeles, CA, Aug. 1985, pp. 375-378.

[12] Pancierz K., and Suraj Z., “Synthesis of Petri Net Models: A Rough Set Approach”, *Fundamenta Informaticae* XX, IOS Press, 2003.

[13] Pawlak Z., “Rough Sets- Theoretical aspect of Reasoning about Data” Kluwer Academic Publishers, 1991.

[14] Pawlak Z., Grzymala-Busse J., Slowinski R., Ziarko, W., “Rough sets” *Communications of the ACM*, vol. 38, no. 11, pp. 89-95, 1995.

[15] Peters J. F., Ramanna S., Borkowski M., Skowron A., and Suraj Z., “Sensor, filter and fusion models with rough Petri nets”, *Fundamenta Informaticae*, vol. 47, no.3,4, August 2001, 307-323.

[16] Zbigniew Suraj, Barbara Fryc, Zofia Matusiewicz and Krzysztof Pancierz "A Petri Net System - an Overview" *Fundamenta Informaticae* vol. 69, no. 3, pp.101-119