We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

8

Using Petri nets for modeling and verification of Hybrid Systems

Ricardo Rodriguez[1,2], Otoniel Rodriguez[3],
Gerardo Reyes[4] and Vianey Cruz[4]
[1]*Technological University of Ciudad Juarez*
[2]*Technological Institute of Ciudad Juarez*
[3]*Autonomous University of Morelos State*
[4]*National Center of Research and Technological Development*
Mexico

1. Introduction

Artificial Intelligence (AI) is a science that came out in the decade of 50s. It emerges having as main objectives the understanding of intelligence and the building of intelligent behaviour systems that carry out complex tasks with a superior or equal level competence to a human performing. This science uses theoretical and experimental Computational techniques in order to emulate human intellectual activities: learning, perception, reasoning, creation or manipulation (Negnevitsky, 2005; Nikolopoulos, 1997).

The AI counts on several disciplines that have arisen in some application fields and it is approached to problems in industrial and commercial sectors. Such is the case in Knowledge-Based Systems (KBS) that have received great attention and have became to be essential tools in business, science, engineering, manufacturing and many others fields.

A KBS is a computational system, it has knowledge, ability and experience that only belong to a person or specialist people group in some area of the human knowledge, in such way, this system can solve specific problems in intelligent and satisfactory way (Negnevitsky,2005).

KBS provide assistance to humans in decision making and normally can explain the reasoning used to obtain a diagnostic or suggestion, and why the questions were inferred as well. These systems can manage large knowledge amount established in declarative form to be used in the intuition and experience. Such knowledge can be integrated with some knowledge representation: rules, facts, heuristics, and in some cases includes uncertainty.

A KBS allows the user to introduce facts or information to the system and obtain as results, advices or experiences. The relevant components of a KBS are: the knowledge base and the inference engine. KB represents the knowledge in application domain. Inference engine gives reasoning capability to solve a particular problem.

The knowledge obtained of a human expert is transformed into a representation frame and the inference is done using it. However, knowledge is not always totally incorporated because of its nature. It causes incompleteness problems in the KB.

The knowledge base is the heart of a KBS and it is coded in some representation language. A particular case of KBS is the Neural-Symbolic Hybrid System (NSHS) that allows interaction of the connectionist and symbolic knowledge contained in it (Nikolopoulos, 1997; Cruz, 2004; Santos, 1998). Hybrid systems allow integrating two or more knowledge representations in one system. It is in order to obtain a knowledge integrating that allows improving the global efficiency of the system. NSHS use an Artificial Neural Network to improve the knowledge that provides a symbolic system. Each one of these subsystems maintains its own representation language and a different mechanism to infer its solution.
When a NSHS is bought, there is a specialist that provides his experience in the application field. Therefore, it is coherent and without contradictions. However, in these systems mistakes can potentially arise as a result of the modeling complexity and the knowledge among to represent.
These systems have been used mainly in classification problems. The knowledge base of a NSHS and any knowledge-based system can contain mistakes since several experts provide their experiences in the field of application, which can be contradictory. Therefore, the verification and validation of knowledge in this kind of systems are critical processes in their construction, and may be focused on the knowledge base or the inference engine.
As the NSHSs gain acceptance, it increases the necessity of ensuring the automatic validation and verification of the knowledge contained in them. Verification has as a principal objective to ensure the consistency and completeness of the KB. Although, it does not warrant that its behaviour corresponds with the human expert knowledge. In the verification processes defined characteristics of a NSHS are evaluated. Such analysis can be restricted to one specific element of the system. As a matter of fact it can be in the inference machine, KB, or user interfase, and it can be focused on system specific stages, for example, in its deductive behaviour (Negnevitsky, 2005).
When the verification process has been finished, the validation process takes place analysing the proper of KB and the possibility of obtaining right solutions to the domain problems.
In some application environments, a NSHS might not be easily accepted nether be started up, at least, it can be conveniently and meticulously proved that works according to expectations (Nikolopoulos, 1997).
The present chapter presents an enhanced Petri net model for the detection and elimination of structural anomalies in the knowledge base of the NSHS. In addition, a modeling process is proposed to evaluate the obtained results of the system versus the expected results by the user. The validation and verification method is divided in two stages: 1) it consists of three phases: rule normalization, rule modeling and rule verification. 2) It consists of three phases: rule modeling, dynamic modeling and evaluation of results. Such method is useful to ensure that the results of a NSHS are correct. A set of tests is presented to demonstrate the effectiveness of the results obtained with this method. The cases of study were obtained from KBs extracted from Neural-Symbolic Hybrid Systems.
The chapter is organized in the following form: section 2 explains some related background in knowledge verification and validation; section 3 describes a Hybrid System Framework with combined knowledge; section 4 presents some aspects of error detection and system modeling; section 5 gives some information about Petri nets modeling; section 6 describes our verification method applied to error detection; section 7 presents our knowledge validation approach; experimental results and discussions are provided in section 8; finally, in section 9 we conclude the chapter and point out the direction of future research.

2. Related Work

A number of works dealing with knowledge verification and modeling have been proposed in the past. Such works have been based on the comparison of rule pairs. However, recent proposals use techniques such as Petri nets, directed graphs and directed hipergraphs (He et al., 2003). In these approaches, nodes are used to represent simple clauses of a rule and directed arcs to represent causal relations.

Petri nets have been used in the study of RBS due to the possibility of capturing dynamics and structural aspects of the system. The rule base can be verified by Petri net analysis techniques. Those techniques have been used in several works. In Yang et al. (1998) an error verification method in a Rule Base (RB) based in an incidence matrix is proposed. This method does not admit negated propositions. It makes a previous ordering of the RB for the verification and it does not need an initial marking of the net for the verification. In He et al. (2003) a reachability graph of a Petri net (PN) for structural anomalies detection in a knowledge base (KB) is presented. This technique is known as *w*-Net, where *w* indicates the amount of tokens existing in each place. Nevertheless, in this technique it is necessary to know the initial marking of the net to detect errors. In Wu & Lee (1997) a variant of classic Petri net named high level extended Petri net is proposed. This model allows the logical negation and the use of variables and constants in the antecedent as well as consequence of the rules. Execution of the model is made by means of input conditions. It uses a reachability approach based on a color scheme for validation.

3. Knowledge Representation in Hybrid Systems

Neural-Symbolic Hybrid Systems are computer programs based on artificial neural networks that also interact with symbolic components (Cloete & Zurada, 2000). These types of systems integrate the connectionist and symbolic knowledge, in such a way that the knowledge contained in each one of these is complemented (Cruz et al., 2005; Cruz et al., 2006; Negnevitsky, 2005; Santos, 1998).

The symbolic knowledge is a set of theoretical knowledge from a particular domain. This knowledge should be translated into a formal representation in order to be used in a computer system. Some knowledge representations are: semantic networks, predicates logic, proposition logic, etc. The representation mostly used is the production rules. A disadvantage of the symbolic representation is that sometimes the characteristics of the objects can not be totally described. It is due to such representation can not make an exhaustive description of the object in all its modalities or contexts.

On the other hand, a different type of knowledge is known as "practical", integrated by a group of examples about an object or problem in different contexts or environments. For example in object recognition, an image base of the objects can be used to describe it in different contexts, positions, environments and with different focus of external quality. In some cases, a numerical representation of RGB colour, high or wide can be important. According to the above mentioned a hybrid approach can be the solution to object recognition problems.

NSHS are a type of Knowledge -Based Systems that can be used in many applications where failures can be expensive in services, properties, or even the life (Cruz, 2004; Tsai et al., 1999). It is important the verification and validation of these systems before their implementation to ensure this way their reliability. The necessity to develop knowledge

verification and validation systems will increase to guarantee the quality and reliability of such systems.
The extracted knowledge of the system is composed of the production rules. Users of a NSHS can interact with the knowledge base because of its representation is extracted and understandable.

3.1 Development Stages of a NSHS

NSHS transfers knowledge represented by a set of symbolic rules toward a connectionist module. The obtained neural network permits a supervised training, starting from a base of examples. In the next step, an extraction algorithm is developed to obtain the knowledge of a neural network into production rule form. Finally, the rules extracted must be verified and validated to be sure that the knowledge obtained in the extracting process is suitable to solve the problem (Cruz et al., 2006; Santos, 1998; Villanueva et al., 2006). The stages of a NSHS are shown in the figure 1.

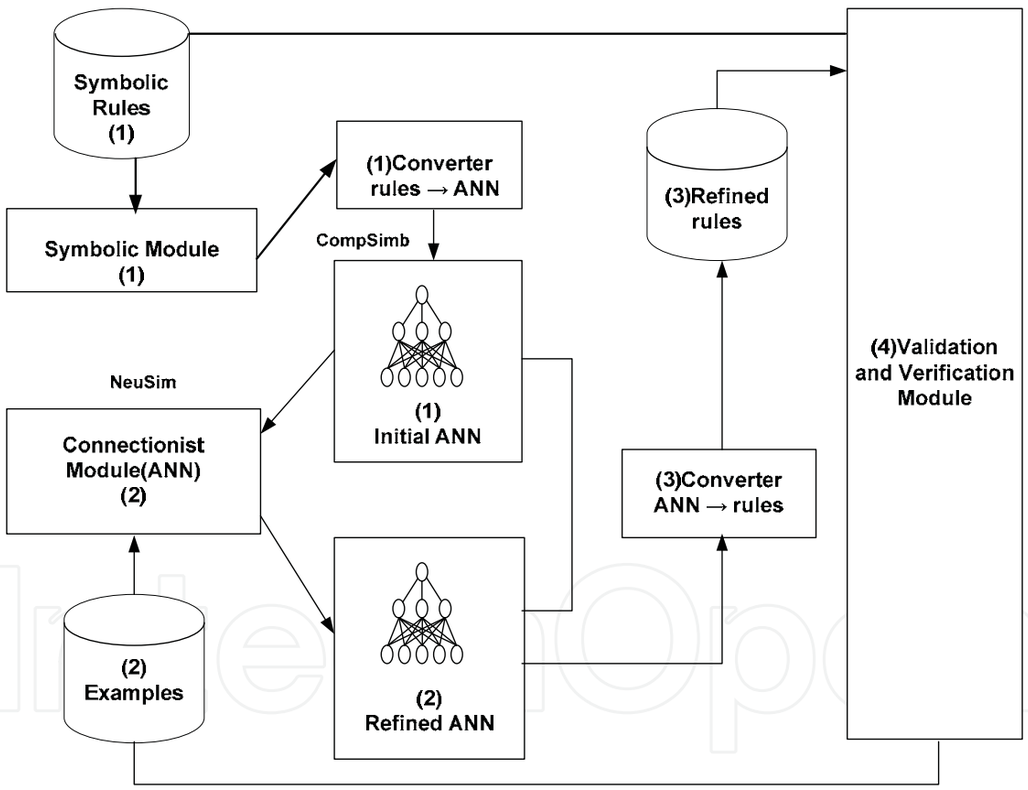

Fig. 1. Stages of a NSHS.

1) Insertion
In this stage, the knowledge extracted from a human expert is symbolically represented (Symbolic Module). This symbolic representation is in form of IF…THEN rules. Subsequently, it is converted to an Artificial Neural Network (ANN) named "initial" ANN. This stage is known as "rules compilation in a neural network" (Cruz, 2004).

2) Refinement
In this stage, a module that receives the initial ANN is implemented, which is subjected to a learning process starting from a base of examples. This module is named connectionist module; at the end of this stage, an artificial neural network is obtained, which is named "refined" ANN.
3) Extraction
In this stage, the extraction of the knowledge contained in the refined neural network is done. At the end of the module, symbolic rules called refined rules are obtained. This process is carried out due to the necessity to interpret and to evaluate such knowledge.
4) Verification and validation
In this stage, the coherence of extracted knowledge in symbolic rules is verified. Just later, tests to analyze the operation of the system as a whole are done.

4. Object of Knowledge Base Verification and Validation

The extracted knowledge of a NSHS is represented in production rules and stores the accessible experiences for the system. Different relations exist in the RB, for example: the conclusions of a rule can act as conditions for other rules and different rules can share common conditions. The production rules describe an IF-THEN relationship of the form $CC \Rightarrow CA$. Where CC is a collection of conditions, CA is a collection of actions or conclusions and the "$\Rightarrow$"symbol acts as a logical implication. The propositions CC can be joined by $\wedge / \vee$ that represent the logical connectives AND/OR respectively. Propositions CA can be joined only by the connective $\wedge$. A negative proposition $\neg p$ in CC is true if p does not exist in the work memory. A negative proposition in CA causes elimination of p in work memory.

4.1 Knowledge Verification Aspects

The main goal of verification is to obtain the consistent, complete and correct system. Because of that, anomalies on KB must be detected (Ramaswamy et al., 1997; Ramirez & De Antonio, 2001; Tsai et al., 1999; Yang et al., 2003). An anomaly is referred to common fault patterns according to an analysis technique (Ramirez & De Antonio, 2007; Tsai et al., 1999). This KB can contain errors due to:

1) The existence of several human experts providing their experiences in the application field.
2) The inserted knowledge can not be represented properly because of communication problems between human experts and the knowledge engineer.
3) The information may be missed during knowledge insertion due to matching of same one to the neural network,
4) The base of examples might be redundant due to a bad selection of samples.
5) Information may be missed or gained during the integration process of numerical and symbolic knowledge (Cruz et al., 2006; Santos, 1998; Villanueva et al., 2006).

It is necessary to consider some verification principles in order to make a suitable definition of the anomaly types that can be found into an extracted knowledge base of a NSHS.

a) Anomalies are defined according to the declarative meaning of the KB, instead of any other procedural meaning.
b) Anomalies are detected by means of analyzing the KB syntax, and should be semantically understood as well.
c) Anomalies are considered as symptoms of possible errors into the KB. Not all anomalies are errors.
d) Anomaly detection methods are just applied to the KB of the KBS and consider some properties of the inference engine.
e) The syntax and semantics of the anomalies should be defined in terms of syntax and semantics of the knowledge representation language used to expressed the KB.

The anomalies that can be found in a KB are shown in figure 2.

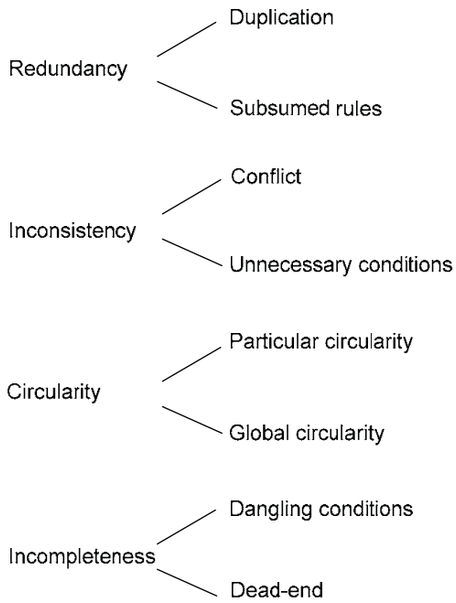

Fig. 2. Anomalies that can be found in a NSHS.

Redundancy: It occurs when there are unnecessary rules in a rule base. Redundant rules increase the size of the rule base and may cause additional inferences. There are two kinds of redundancies:

a) Duplication (equivalent rules). If $R1: a \wedge b \rightarrow c$ and $R2: b \wedge a \rightarrow c$, R1 and R2 are totally equivalents.

b) Subsumed rules (rule contained in another one). If $R1: a \wedge c \rightarrow b$ and $R2: a \rightarrow b$, R1 subsumes to R2.

Inconsistency: It occurs in conflictive facts. Here two types of inconsistencies are approached:

a) Conflict. A set of rules are conflicting if contrary conclusions are derived under a certain condition. An example of rules in conflict is the following: $R_1: a \rightarrow b$ and $R_2: a \rightarrow \neg b$.

b) Unnecessary conditions. A pair of rules $R, R' \in \Re$ has an unnecessary condition if they have the same consequent and the antecedents are only different in that some of the

propositions in R are negated in R'. An example of rules with unnecessary conditions: $R_1 : a \wedge b \rightarrow c$ and $R_2 : a \wedge \neg b \rightarrow c$.

Circularity: It occurs when several inference rules have circular dependency. Circularity can cause infinite reasoning that must be broken. The following are examples of circularity.

a) Particular circularity. If $R1 : a \rightarrow b$ and $R2 : b \rightarrow a$, a cycle is formed in pairs of rules.

b) Global circularity. If $R1 : a \rightarrow b$, $R2 : b \rightarrow c$, $R3 : c \rightarrow a$ then chaining of circular rules appears.

Incompleteness: It occurs when there are missing rules in a rule base. If $R1 : \rightarrow a$, $R2 : a \wedge c \rightarrow b$, $R3 : a \rightarrow d$ and $R4 : b \rightarrow$ Here we present two types of Incompleteness.

a) Rule with dangling conditions. It occurs if the condition will never be matched by some conclusion. In R2 the condition *c* is never matched with any conclusion of the rest of the rules, therefore *c* is a dangling condition.

b) Rule with dead-end. It appears when a conclusion is not a goal and is not used as condition in any other rule. In R3, *d* is never matched by any condition of the rest of the rules; therefore *d* is a conclusion with a dead-end.

4.2 Knowledge Validation Aspects

The validation process allows analyzing the quality of the KB and the possibility of obtaining right solutions to the problems of the domain (Knauf et al., 2002; Tsai et al., 1999). This process is done in order to evaluate a system during or at the end of the development process to establish if it satisfies the specified requirements.

An important feature of the KBS is that its specifications and requirements are dynamically changing. It is more difficult to develop the verification and validation in systems which requirements or some other elements are frequently changing than in systems with static characteristics. There are different validation approaches such as: reachability, reliability, safety, completeness, consistency, robustness and usability.

A NSHS is only accepted when is convincingly and meticulously verified to work according to expectations.

The general evaluation frame consists of the following steps:

a) A testing criterion must be established, as reachability, reliability, safety, completeness, consistency, robustness and usability.

b) Test cases (inputs) and awaited outputs according to the selected inputs must be generated.

c) A test method to exercise the software must be applied.

d) The outputs of the software must be evaluated.

Tests are in general an intense work and a process prone to faults. Difficulties arise from different test criterions, great input and output space and legal input case generation (Knauf et al., 2002; Nikolopoulos, 1997; Vermesan, 1998).

Test criterion. It defines the objective of comparing a system versus a specification. Different types of tests are defined according to different types of test criterions.

Test case generation. Proper test inputs can be specified according to the type of problem that we can solve. It is possible to use the literature of the domain to generate the inputs as test cases.

Expected output generation. An expected output consists of a response. It is possible to ask to the expert of the domain to predict the expected outputs and generate them. Furthermore, expected outputs can be generated according to explicit solution specifications, saved test cases or literature of the domain.
Test method to exercise the software. The cost of the test method not only depends on the cost of the test case generation and on the evaluation cost. It also depends on the fact that a valid result is not found.
Evaluation of software outputs. It consists on evaluating if the generated output set belongs to the expected outputs in the solution of the problem (Nikolopoulos, 1997).

5. Petri nets as a modeling tool

Petri nets came in the literature on 1962 with the phD thesis of Carl Adam Petri (Murata, 1989). Petri nets are mathematic and graphic modeling tools that can be used in several types of dynamic systems. Petri nets allow describing and studying information from concurrent, asynchronous, distributed, parallels, nondeterministic, and/or stochastic systems (Nikolopoulos, 1997; Murata, 1989; David et al., 2005).

5.1 Petri nets graphic representation

A Petri net is a particular type of directed graphic and it is represented graphically by bipartite graph (there are two types of nodes and the arcs just can connect nodes of different types).Two types of Petri nets nodes are named places and transitions. Places represent variables that define the system state and transitions represent the transformer of such variables. Places are represented by circles, transitions by bars and marks are represented by a point into the circle which defines the place that contains it. Places and transitions are connected by directed arcs (Murata, 1989; Wu et al., 2000). The figure 3 shows the graphic representation of the Petri net elements.

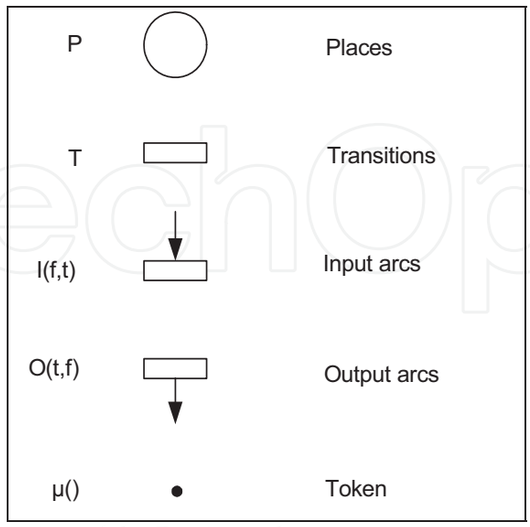

Fig. 3. Graphic Representation of the Petri net elements.

Figure 4 shows a Petri net with places: P= {P1, P2, P3, P4} and transitions: T = {T1, T2, T3, T4}.

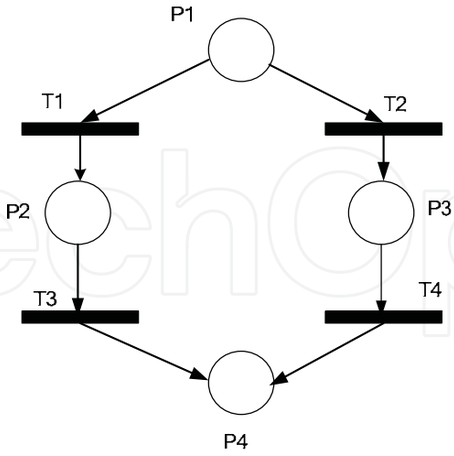

Fig. 4. Places, transitions and arcs of a Petri net.

An arc directed from a place P_i to a transition T_j defines an input place of the transition. Multiple inputs to the transition are indicated by multiple arcs from the input place to the transition. An output arc of a transition is indicated by an arc directed from the transition to the place. Multiple outputs are represented by multiple arcs (Murata, 1989).
Places can be used as "tokens" containers. The number of tokens contained in a place is named mark. Therefore, the net marking is defined as a column vector where the elements are the number of tokens contained in the places (David et al., 2005). Figure 5 shows the net marking: $M=\{m1, m2, m3, m4\}$, in other words: $M=[1,1,0,0]^T$.

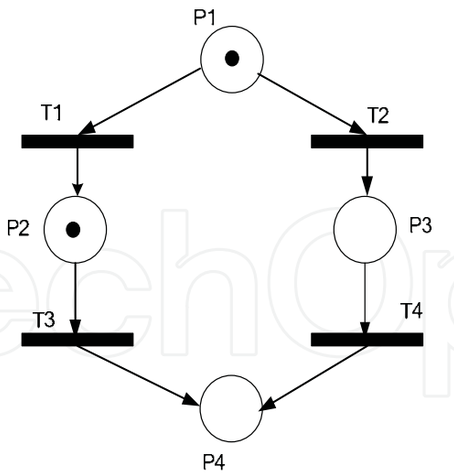

Fig. 5. Net marking.

Tokens can move inside the net, changing the state of the same one. In order to move the token, transitions must be fired (David et al., 2005).
A transition can be fired if it is enabled. A transition is enabled if into each one of its input places there is, at least, as many tokens as the arc weight that connects with them. A source

transition is always enabled. This type of transitions does not have input places (David et al., 2005).
Figure 6 shows three enabled and disabled transitions examples. Figure 6a presents t1 enabled due to its places P1 and P2 are in compliance with the enable conditions. In figure 6b, t2 is not enabled due to the place p5 has less tokens than the weight of the arc that connect it with the transition. In the figure 6c the source transition t3 is enabled.

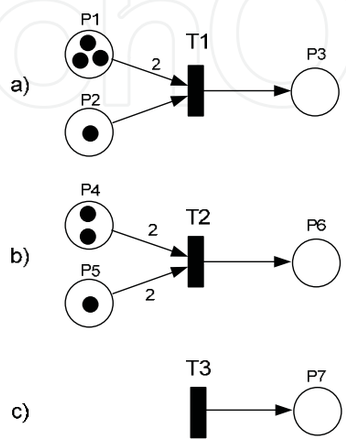

Fig. 6. Enabled and disabled transitions.

Firing of an enabled transition eliminates from each one of its input places as many tokens as the weight of each arc connecting such places with the transition. Also deposits in each output place as many tokens as the weight of each arc connecting them with the transition (David, et al. 2005; Murata, 1989).
Figure 7 shows a Petri net with two enabled transitions t_1 and t_3. Figure 8 shows the Petri net after the firing of the transition t_1.

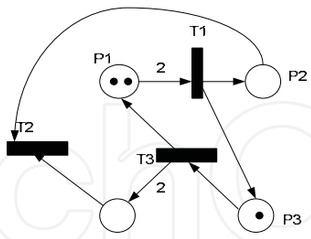

Fig. 7. Petri net with t1 and t3 enabled.

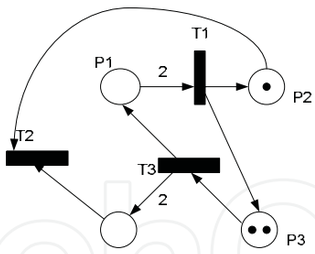

Fig. 8. Petri net after the firing of t1.

5.2 Petri nets formal definition

Definition: The structure of a Petri net is: $N = \{P, T, F, W, M_0\}$.

where:

1) P is a set of places, T is a set of transitions and F is a set of flow relations. It means: $P \cap T = 0, P \cup T \neq 0, F \subseteq (PxT) \cup (TxP)$.
2) W is a mapping of $F \rightarrow \{1,2,3...\}$, it is a weight function.
3) M_0 is the initial marking $M_0 : P \rightarrow \{0,1,2,3...\}$, it is the initial marking.

5.3 Types of Petri nets

Nowadays, several types of Petri nets have been developed to be able of being applied to solve specific problems, and there are several classifications (David et al., 2005; Nazareth et al., 1991). Table 1 shows some types of Petri nets, as well as some of its characteristics. In this classification, Low level Petri net (Place/Transition) is emphasized as well as the most common extensions (Colored, Stochastic and Hierarchical).

Name of net	*Type of net*	*Characteristics*
Pure Petri net	Classic net	Do not contain self cycling. Input places of t are not at the same time output places of T.
Ordinary Petri net	Classic net	If the weight of an arc is 1.
Finite Capacity Petri net	Classic net	Each place limits the number of tokens that it is able to contain.
Regular Petri net	Extended net with time	It has associated a determinate firing time
Stochastic Petri net	Extended net with time	It has associated a stochastic firing time.
Hierarchical Petri net	Extended net with hierarchy	A hierarchy of subnets is provided.
Colored Petri net	Extended net with color	It is very linked to its modeling language and allows working with tokens of different colors to represent values and types of data which the modeling system works with.

Table 1. Petri nets clasification.

Classic Petri nets are also known as Low Level Petri nets, due to the modifications and restrictions made to the net are just arc connection conditions and the number of tokens that can store the places. However, Classic Petri net does not allow data and time modeling. Therefore, several extremely big and complex extensions for modeling real world have been proposed.

Extensions with time came up due to the necessity of describing the temporal behavior of the system. This type of nets can be divided into two classes: Deterministic time nets (regular Petri net) and Stochastic time net (Stochastic Petri net). The first one, include Petri nets that have associated a determined firing time in its transitions, arcs or places. The second one, include Petri nets that have associated a stochastic firing time in its transitions, arcs or places.

Hierarchical extensions have been used to manage the size problem that Petri nets face to model real systems. It is based on a restructuring mechanism of two or more processes that can be represented by subnets. In one level a simple description of the processes is done, while in another one it is done a detailed description of its behavior. This is the case of the client/server schemes where subnets are conveyed with each other using a place type graph.

Extensions with colors are known as colored Petri net. This type of nets combines the Classic Petri nets advantages and the high level programming languages, because of that this type of Petri nets allows the representation of different data types in the model by means of tokens that flow inside the net. In a colored Petri net the concept of token color is used. This allows having tokens of different colors, where each token color represents a piece of information. The model of a colored Petri net is more compact than the equivalent model in a classic Petri net (Chavarría & Li, 2006; David, et al. 2005; Nikolopoulos, 1997).

There are other Petri net extensions that have been developed, such as: hybrid and continuous Petri nets, as well as fuzzy Petri nets. Continuous Petri nets are a model where the mark number in the places is a decimal point number instead of an integer. On another hand, hybrid Petri nets are characterized by having a discrete part and a continuous part.

5.4 Enhanced Petri net model

Traditional Petri nets have inherent disadvantages. Some of these disadvantages are: deficiency of flexible descriptions for negative relations and necessity to formalize the original KB before beginning the verification process, such is the case of the example of a logical system or production systems, due to the difficulty of expressing logical disjunctions (He et al., 2003; Nazareth, 1993; Wu & Lee, 1997; Tsai et al., 1999; Yang et al., 1998). In order to overcome these problems an enhanced Petri net model is proposed by us.

Definition: An enhanced Petri net is a sextuple.

$$N = \{P, T, F, C, I_-, I_+\}, \quad F = F_b + F_r \tag{1}$$

Where:

1) *P* is a set of places, *T* is a set of transitions and *F* contains the set of inhibitor or activator relations between *CC* and *CA*. Therefore, $F = F_b + F_r$.

2) *C* means that for any $p \in P, C(p)$ is a collection of possible colors in P. For any $t \in T, C(t)$ is the collection of possible colors in the transition T.

3) I_- and I_+ are negative and positive functions of PxT respectively. For any $(p,t) \in PxT$. I_- and I_+ are the previous and later transition matrices respectively.

The elements *P, T, F_b, F_r* are named predicate symbols (places), implications (transitions), activator arcs and inhibitor arcs, respectively. For a transition $t \in T$, a positive place $p_b \in P$ of *t* is a place that connects to t with an activator arc and presents a positive relation between p_b and *t*. A negative place $p_r \in P$ of *t* is a place that connects to *t* with an inhibitor arc and presents a negative relation between p_r and *t*. The elements of the EPN modeling are shown in the figure 9.

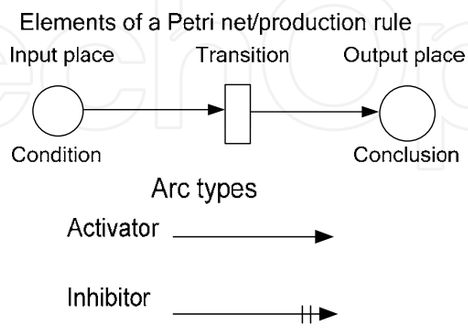

Fig. 9. Elements of the EPN modeling.

For the rule R_i, t_i is its transition. The premises $C_1(r_i) \wedge C_2(r_i) ... \wedge C_n(r_i)$ are *t and the conclusion $A_1(r_i)$ is t*. The representation of the rule is shown in figure 10.

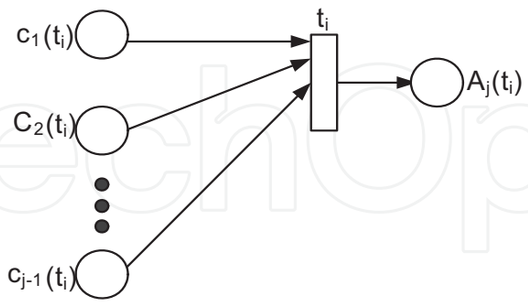

Fig. 10. EPN modeling Example.

As a PN, EPN can be mathematically represented by its incidence matrix which shows the interactions between places and transitions. In an incidence matrix A_{mxn} the n columns represent places and the m rows represent transitions of N. The table 2 shows values that can have the incidence matrix of the enhanced Petri net.

Value	*Means*
-1	The place $p_j \in P$ is an input place to the transition $t_i \in T$.
0	There is not an arc connecting the place $p_j \in P$ with the transition $t_i \in T$ and vise versa.
1	The place $p_j \in P$ is an output place of the transition $t_i \in T$.

Table 2. Values that can have the EPN incidence matrix.

6. Improper Knowledge Detection

In this section we propose a PN based mechanism to detect and eliminate structural errors of a KB extracted from a NSHS, which consists of three phases: *rule normalization*, *rule modeling* and *rule verification*. For the rule base verification, a static analysis of the EPN model is done by means of obtaining its incidence matrix.

Rule normalization

This step is done in order to simplify the rule base analysis. In this phase rules are translated into an *atomic form*:

$$R_i = C_1(r_i) \wedge C_2(r_i) ... \wedge C_n(r_i) \Rightarrow A_1(r_i) \tag{2}$$

It guarantees that each of its parts (CC/CA) cannot be decomposed into subparts. In an atomic rule the left condition only permits the conjunction of zero or more conditional clauses and just one element as action is permitted. The kinds of rules possible to normalize are:

1) $P_1(r_i) \wedge P_2(r_i) \wedge ... \wedge P_{j-1}(r_i) \rightarrow P_j(r_i) \wedge P_{j+1}(r_i) \wedge ... \wedge P_k(r_i)$

2) $P_1(r_i) \vee P_2(r_i) \vee ... \vee P_{j-1}(r_i) \rightarrow P_j(r_i) \wedge P_{j+1}(r_i) \wedge ... \wedge P_k(r_i)$

In order to obtain an atomic rule, first, each one of the elements of production rule is transformed into disjunctive form. An element in this form consists of one or more disjunctions, each one of these being a conjunction of one or more propositions. The rules are converted to disjunctive form using the distributive property of AND over OR, the idempotency and the contradiction. It means we have to use logical equivalences. The logical equivalences used in this work are shown in table 3.

Idempotency	$A \wedge A \equiv A$ $A \vee A = A$
Complementary element	$A \wedge \neg A \equiv F$ $A \vee \neg A \equiv V$
Distributive	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
Commutative	$A \vee B \equiv B \vee A$ $A \wedge B \equiv B \wedge A$

Table 3. Logical equivalences.

Next, each subpart of the rule corresponding to each disjunction is separated. Also, a rule can have conjunctions in CA, which indicates it has multiple actions. In this way, separated rules are obtained corresponding to each action with the same set of premises that the original rule has.

In this chapter, we do not discuss the rules normalization with disjunctions in CA because the conclusion disjunctions do not make an explicit implication.

Rule modeling to EPN

In this step, rule mapping of the KB to EPN modeling is shown in the section Enhanced Petri net model. The rule modeling to EPN is made once the rules were normalized to their atomic form. Mathematical representation of the EPN is also obtained.

Rule verification

The following notations will be used during the rule verification process: *CC(i)* is a set of conditional clauses of *i-th* transition. *CA(i)* is a set of conclusion clauses of *i-th* transition. Also:

$C_{-1,i} = \{ P_k \mid P_k$ *is the place in the j-th column in matrix A such that* $A_{i,j} = -1\}$

$C_{1,i} = \{ P_k \mid P_k$ *is the place in the j-th column in matrix A such that* $A_{i,j} = 1\}$

$C_{0,i} = \{ P_k \mid P_k$ *is the place in the j-th column in matrix A such that* $A_{i,j} = 0\}$

Property 1. Redundancy. For t_i and t_k $i \geq 1, k \leq m, i \neq k$; two transitions represent the rules *i* and *k* of the rule base, respectively, and execute the same action. There are two kinds of redundancy. If t_i and t_k satisfy any of the following conditions. $1) CC(i) \subset CC(k)$ or $2) CC(i) = CC(k)$ then the rules *i* and *k* are redundant. Duplication and subsumtion, are two kinds of redundancy.

Proof:

1) If $CC(i) \subset CC(k)$ then $|CC(i)| < |CC(k)|$. If the rows of A_{mxn} represent transitions and the set of conditions represents the places, and such places $p_b \in CC(i)$ and $p_r \in CC(i)$ are equivalent in *CC(k)* then the rule *i* is subsumed in the rule k, i.e., $A_{i,j}$ =1 y $A_{k,j}$ =1, $1 \leq j \leq n$ and for the rest of places *j*, $A_{i,j} < A_{k,j}$.

2) $CC(i) = CC(k)$ indicates both rules *i* and *j* have the same conditions. In such a way, places $p_b \in CC(i)$ and $p_r \in CC(i)$ are equivalent in *CC (k)*, although the conditions are presented in different order, *i* and *j* are equivalent rules, i.e., $A_{i,j} = A_{k,j}$. for all *j*, $1 \leq j \leq n$.

Property 2. Inconsistency. For t_i and t_k $i \geq 1, k \leq m, i \neq k$; two transitions represent the rules *i* and *k* of the rule base, respectively. If t_i and t_k satisfy $CC(i) = CC(k)$ and $CA(i) \neq CA(k)$ then rules *i* and *k* are conflictive rules. If t_i and t_k execute the same action and either $p_b \in CC(i)$ or $p_r \in CC(i)$ are not equivalent in *CC(k)*, then that condition is necessary in t_i and t_k.

Proof:

1) If $A_{i,j} = A_{k,j}$. For all j $1 \leq j \leq n$ in $C_{-1,k}$ and $C_{-1,i,}$ and the places $p_b \in CC(i)$ and $p_r \in CC(i)$ are equivalent in *CC(k),* but $C_{1,k} \neq C_{1,i}$.

2) If $A_{i,j} = A_{k,j}$. For all j $1 \leq j \leq n$ in $C_{-1,k}$ and $C_{-1,i}$ and the places $p_b \in CC(i)$ and $p_r \in CC(i)$ are equivalent in *CC(k)* and $C_{1,k} = C_{1,i}$. However, $p_b \in CA(i)$ and $p_r \in CA(i)$ are different in *CA(k).*

3) If $A_{i,j} = A_{k,j}$. For all j $1 \leq j \leq n$ in $C_{-1,k}$ y $C_{-1,i,}$ and someone places $p_b \in CC(i)$ and $p_r \in CC(i)$ are not equivalent in *CC(k).*

Property 3. Circularity. For $t_i, t_k, \ldots, t_m$, $i \geq 1, k \leq m, i \neq k, \ldots, \neq m$; *MT*:* memory work. All the transitions add their respective places to *MT** (if they were not in). In such a way if $p_b \in MT^*$ or $p_r \in MT^*$ and if it is deduced again in some transitions, then the place *p* causes circularity.

Proof:

1) t_i and t_k are two transitions represent the rules *i,k* of the rule base. *MT**= $\{ \{p_b \mid p_r\} \in p \mid p$ is the *j-th* column of *A* such that $A_{ij} = 1\}$ and *MT**= $\{ \{p_b \mid p_r\} \in p \mid p$ is the *j-th* column of *A* such that $A_{ij} = -1\}$. It checks t_k, $\{A_{k,j} = 1\} \in MT^*$. Then there is contradiction in the rules *k* and *i*, and we consider particular circularity.

2) t_i,t_k,…,t_m are transitions representing the rules *i*,*k*,…,*m* of the RB. *MT**=$\{\ \{p_b \mid p_r\} \in p \mid p$ is the *j-th* column of *A* such that $A_{ij} = 1\}$ and *MT**=$\{\ \{p_b \mid p_r\} \in p \mid p$ is the *j-th* column of *A* such that $A_{ij} = -1\}$. It checks in $t_{m,,}$ $\{A_{k,j} = 1\} \in MT*$. Then there is global circularity.

Property 4. Incompleteness. t_i y t_k, $i \geq 1, k \leq m, i \neq k$ are two transitions representing the rules *i* and *k* of the rule base, respectively; P_j is in wherever place of the transitions matrix.

Proof:

1) $\exists P_j \in C_{-1,i}$ and $\{p_b \mid p_r\} \in p_j$ for some *i*, such that $p_j \notin C_{-1,k}$ for all *k*. P_i is the condition of some rule, but it is not the conclusion of any rule. Then the corresponding rule has a dangling condition P_j.

2) $\exists P_j \in C_{1,i}$ and $\{p_b \mid p_r\} \in p_j$ for some *i*, such that $p_j \notin C_{-1,k}$ for all *k*. P_j is the condition of some rule, but it is not matched as condition of some other rule. Besides, p_j is not a goal. The corresponding rule of p_j is a rule with dead-end.

7. Knowledge Validation

In this section, we propose a mechanism to analyze the resultant rules from the verification process of a NSHS. This mechanism presents a multiple color scheme in the enhanced Petri net (see subsection Enhanced Petri net model). The reachability of the production system is probed on the basis of the dynamic logic inference made on the enhanced Petri net. The method consists of three phases: *rule modeling to EPN, dynamic modeling of the EPN, evaluation of results.*

Rule modeling to EPN

This step is done in order to generate the enhanced Petri net as shown in section Enhanced Petri net model. From which the verified knowledge base is received.

First, The KB is mapped to the EPN model and the mathematical representation of the same one is obtained. Then, the *transposed incidence matrix* is generated, and it is represented as A^T.

The places of the enhanced Petri net model are classified into two categories: $P = \{P_C, P_R\}$. For analysis conveniences, P_C is divided into three sub-sets, $P_C = \{P_{CE}, P_{CI}, P_{CG}\}$. P_{CE} is the places collection which can obtain information through inputs by the user. P_{CI} is the places collection produced in the inference process and P_{CG} is the places collection being the conclusion of the system. On the other hand, P_R is the collection of transition places used by the model to avoid firing the same transition once the system has already fired it. It is due to the initial and deduced facts (in the inference process) are kept when the transition fire and can be used by the system has already fired it multiple rules. The reason is that the input places of each transition were held as output places of the same transition (see figure 11).

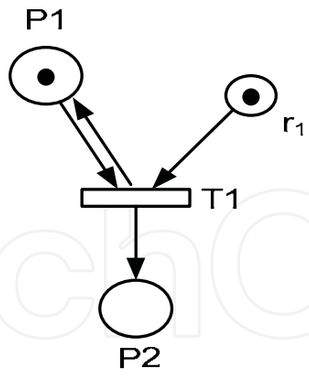

Fig. 11. Representation of the places: PCE= {P1} and PR={r1}.

Closed world assumption. This Enhanced Petri net model works under closed world assumption, which says that if a fact is unknown, any query about it is falsified. In fact, it assumes that all positive information has been specified. Any other fact not specified is assumed as false. The negation acts as if some additional rules are added to insert all the negative information when the NSHS is consulted (Wu & Lee, 1997).

Unknown and know facts. The known facts act as input and output of the rules. The Petri net model identifies them as tokens. When the transition fires, the model deletes a token that represents an input fact from some positive input place of the transition. On the other hand, when the rule fires and it has negative propositions in the right side of a rule, it falsifies the existence of the token in that place. In order to preserve the known facts, the model must preserve the negative propositions on the left side of the rule if these exist (Wu & Lee, 1997; Wu et al., 2005).

Refraction. The known facts reside in the work memory after rules fire. In order to do that, the Petri net model was modified to Enhanced Petri Net, attaching input places of a transition as well as output places of the same one (Nazareth, 1993; Wu & Lee, 1997; Wu et al., 2005)

Dynamic modeling of the EPN

This step is done in order to model dynamically the EPN. The developed validation method uses negation in reasoning of Close World Assumption and an initial marking from facts known by the user, which will be incorporated to the system. It uses the transposed matrix, composed by the input and output places of the transitions. The method incorporates matrixes and a reachability problem studied by means of an equation matrix set.

This modeling is made from goals and facts known by the user.

The facts known by the user act as inputs and outputs from rules. The Petri net model identifies them as tokens in the P_{CE} places. The color set: $D = \{b, r_d, r_f, f\}$ is used to the marking of the EPN and the possible tokens in places are: b, r_d, r_f and f. Color b means that the clause or conclusion represented by the place is true. Colors r_d and r_f mean that the clause or conclusion represented by the place is *deduced false* and *defaulted false*, respectively. Tokens in P_R have the color f which means the rule has been already fired or not. Formally, the marking of the EPN is an indexed vector with respect to the places, which gives to each place p a defined *Multi-Set* (MS). The *initial marking* of the EPN is defined by using a formal notation based on sums as follows: $M_0 = p_1(1b+1f) + p_2(1b+1f)$,.

The marking $M_0(P_R) = [1f]$, means that exist no rules fired initially.

Goals of the system are also provided by the user and they represent the P_{CG} places. They act as conclusions of the system and the goals expected by the user from the dynamic modeling. Rules forward chaining is used to the dynamic modeling of rules. Dynamic equations control dynamic behavior of the system. A *t* transition could fire if it is enabled with a color *(X(t))* under a marking μ. If *t* is fired then the μ marking is changed to the reachable marking $\mu' = \mu'_b \cup \mu'_r$ by means of:

$$\begin{aligned}
&\mu'_b(p) = \mu_b(p) - I_{b-}(p,t)(X(t)) + I_{b+}(p,t)(X(t)) \\
&\mu'_{r_d}(p) = \mu_{r_d}(p) - I_{r-}(p,t)(X(t)) + I_{r+}(p,t)(X(t)) \\
&\mu'_{r_f}(p) = \mu_{r_f}(p) - I_{b+}(p,t)(X(t)) - I_{r+}(p,t)(X(t)) \\
&- I_{b^-}(p,t)(X(t)) - I_{r^-}(p,t)(X(t))
\end{aligned} \tag{3}$$

All $p \in P$, where the operators "+" and "-" are operations of "addition" and "subtraction" on multi-set, respectively. Where *(X(t))* is considered a fire element and μ' the following marking of μ. The transition firing model in the validation phase is a dynamic logic inference and an explanation logic reasoning.

Figure 12(a), presents the modeled Petri net that simulates the behavior of the rule: $R1: A \rightarrow B$. In the model, an additional node named rule place is provided.

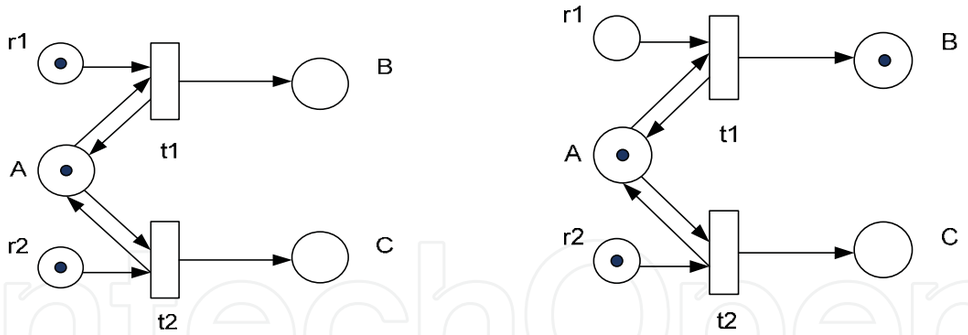

Fig. 12. Modeling of the transition t1: (a) Initial marking, (b) Firing of *t1*

Figure 12(b) expresses that the condition A is true, and then the transition of the corresponding rule *r* is enabled because "the rule place" has a token. It expresses that the conclusion B is true after the transition firing and that the condition *A* keeps the token even after the transition firing.

Finally, the inference mechanism is stopped when it cannot fire any other transition (there are no enabled transitions) or when some deduced fact is a goal proposition. The obtained marking is known as the final marking of the chaining and it represents the reachability of the net from initial marking to final marking. In each test, the system obtains the following data: initial facts used in the net, visited rules, fired rules, deduced facts and finally, the

obtained results notification (valid or invalid results). If the final marking contains some place considered as a goal then this test is labeled as valid result. Otherwise, the test is labeled as an invalid result.

Evaluation of results

This step is done in order to evaluate all the performed tests with different initial marking. It allows the evaluation of the reliability of the obtained results with such initial markings.

The system is considered reliable when valid results are obtained from applied test input. If no valid results are obtained in any performed test, the quality of the input test is analyzed. In case of having applied properly test inputs, we deduce the system contains inconsistent, partially erroneous, or incomplete knowledge. In reason to, it is convenient to perform the verification process again.

8. Experimental results

Based on our approach addressed in previous sections, the validation and verification example 1 of a KB used by a NSHS is showed.

Example 1. We propose a KB extracted from an example base which contains information about 24 patients who were examined to diagnose if they should use contact lenses according to some of their symptoms. The following data are the principal features of the example base.

Attribute information:

Age of the patient: Young, Pre-presbyopic, Presbyopic
Tear production rate: Reduced, Normal
Astigmatic: No, Yes
Spectacle prescription: Myope, Hypermetrope

Classes: Hard Contact, Soft Contact, No Contact

Distribution of the classes:

Hard Contact: 4 instances
Soft Contact: 5 instances
No Contact: 15 instances

The following KB was extracted from the example base mentioned above. In this case, the set of rules is normalized. It can be easily mapped into an EPN.

R1: ***If*** *-(Tear(Reduced))* ***Then*** *Hard_Contact*
R2: ***If*** *Astigmatic* ***Then*** *Hard_Contact*
R3: ***If*** *-(Spectacle(Hypermetrope)) And -(Age(Are- presbyotopic)) And -(Age(Presyopic))* ***Then*** *Hard_Contact*
R4: ***If*** *-(Tear(Tormal))* ***Then*** *Hard_Contact*
R5: ***If*** *-(Astigmatic) And -(Tear(Reduced))* ***Then*** *Soft_Contact*
R6: ***If*** *Spectacle(Hypermetrope) And Age(Young) And Age(Are-presbyotopic)* ***Then*** *Soft_Contact*
R7: ***If*** *-(Age(Presyopic)) And -(Spectacle(Myope))* ***Then*** *Soft_Contact*
R8: ***If*** *Tear(Tormal) And NewUnit1* ***Then*** *Soft_Contact*
R9: ***If*** *Tear(Reduced)* ***Then*** *No_Contact*
R10: ***If*** *-(Age(Young))* ***Then*** *No_Contact*
R11: ***If*** *-(NewUnit1)* ***Then*** *No_Contact*
R12: ***If*** *Astigmatic* ***Then*** *No_Contact*
R13: ***If*** *Spectacle(Hypermetrope)* ***Then*** *No_Contact*

R14: ***If*** *Age(Presyopic) And Age(Are-presbyotopic) And -(Spectacle(Myope)) And Tear(Tormal)* ***Then*** *No_Contact*
R15: ***If*** *Astigmatic And Age(Are-presbyotopic) And Tear(Reduced) And Spectacle(Hypermetrope)* ***Then*** *NewUnit1*
R16: ***If*** *-(Spectacle(Myope))* ***Then*** *NewUnit1*
R17: ***If*** *-(Tear(Tormal))* ***Then*** *NewUnit1*

8.1 Results of the verification process

For this analysis, the propositions: *Hard_Contact, Soft_Contact and No_Contac* were used as goals. First of all, the normalization process was made to the KB (For this case, the KB was already normalized).
Next, the mapping of KB for EPN modeling and its incidence matrix was made. Finally, the verification process was done. The table 4 shows the obtained results from verification process applied to the previous KB.

Rule	**Evaluation**
2	Conflict with R12, dangling conditions
4	Conflict with R17, dangling conditions
12	Conflict with R2
17	Conflict with R4, dangling conditions
1,3,5,6,7,8,9,10, 11,13,14,15,16	Dangling conditions

Table 4. Results from the verification process.

In this process we can see that incomplete knowledge due to dangling conditions was detected in all rules (conditions will not be matched with any conclusion).
The users make elimination of rules according to their requirements, except for duplicated rules, which are eliminated automatically. The new rule base contains less rules than the original one, it is free of errors and is better structured.

8.2 Results of the validation process

For this analysis, each of the test used different initial facts. *Hard_Contact, Soft_Contact* and *No_Contact* were used as goal propositions. Table 5 shows the obtained results from validation process applied to the verified KB.

Tests
Test:1 *Initial facts:* Spectacle (Hypermetrope), Age(Young), Age(Are-presbyotopic), *Visited rules:*1; *Fired rules:*1, *Deduced facts:*Hard_Contact, *Validate:*YES->goal: Hard_Contact
Test:2

Initial facts: Spectacle(Hypermetrope), Age(Young), Age(Are- presbyotopic), Tear(Reduced), *Visited rules:*1,2,3,4; *Fired rules:*4, *Deduced facts:*Hard_Contact, *Validate:*YES->goal: Hard_Contact
Test:3 *Initial facts:* Spectacle(Hypermetrope), Age(Young), Age(Are-presbyotopic), Tear(Reduced), Tear(Tormal), *Visited rules:*1,2,3,4,5,6; *Fired rules*:6, *Deduced facts:*Soft_Contact, *Validate:*YES->goal: Soft_Contact

Table 5. Results from the validation process.

9. Summary

In this chapter an Enhanced Petri Net model is presented for the verification and validation of Neural Symbolic Hybrid Systems. KBs of the NSHS are expressed in production rules based on propositional logic. Such KBs can involve negative information and contain disjunctions in their production rules. These aspects can be expressed by using our EPN method, but not in a traditional Petri net. It is due to in a traditional Petri net some authors create a new place to represent a negative proposition, and use a new place to represent a set of disjunctions. Our method reduces the processing time in validation and verification processes.

The verification module allowed us to formalize the checking concepts of the KB using a formal and conceptual frame to the specification of such checking. This formalization improves the understanding of verification in reason to the group of anomalies that might arise and the algorithms that can be used to detect them.

The considered anomalies have to be with four fundamental properties of verification, which are: redundancy, circularity, inconsistency, incompleteness. Besides, this method shows to the knowledge engineer the rule group potentially in conflict to its previous analysis.

Our verification method is based on incidence matrix of an EPN. This method has the advantage that it is independent from the initial marking of the net.

According to the developed tests, the extracted knowledge in form of production rules and any other KB of a Rule Based System, can contain the errors presented in the anomaly definitions.

The validation method is based on a reachability analysis of the enhanced Petri net. This analysis is executed from test cases and expected goals with such selected inputs. Important aspects of RBS such as facts conservation, refraction and closed world assumption, can be easily modelled from the color scheme here presented.

Cases of test are considered according to: complexity of the KB, relations between evaluated and deduced propositions following an inference process. Besides, cases of test are considered with respect to the expected goals.

As future work we consider to include verification and validation of production rules with uncertainty factors. In order to do this, it would be necessary to redefine other verification

definitions. This adaptation leads us to consider extending the redundancy and inconsistency definitions in order to detect such anomalies in a set of rules. Another promising line of work we are tackling is an extension of the proposed method that permits detecting incompleteness using submarking reachability and simulation.

10. References

Chavarria, L., & Li X. (2006). Structural error verification in active rule based systems using Petri nets. *Proceedings of the Fifth Mexican International Conference on Artificial Intelligence (MICAI'06)*, pp. 12 - 21, ISBN: 0-7695-2722-1, Apizaco, Mexico, Nov 2006, IEEE Computer Society.

Cloete, I. & Zurada, J. (2000). *Knowledge-Based Neurocomputing (Eds.)*. MIT Press, ISBN: 0-262-03274-0, Cambridge, MA.

Cruz, V. (2004). Neural-Symbolic Hybrid System to refine the knowledge of an Artificial Vision System. Master Thesis, Cenidet, Cuernavaca, Morelos, Mexico.

Cruz, V.; Reyes, G.; Vergara, O.; Perez, J. & Montes, A. (2005). Compilation of symbolic knowledge and integration with numeric knowledge using hybrid systems. *Proc. 4th Mexican International Conference on Artificial Intelligence (MICAI'05),* pp. 11 – 20, ISBN: 3-540-29896-7, Monterrey, Nuevo León, México, Nov 2005, Springer Lecture Notes in Computer Science.

Cruz, V.; Reyes, G.; Vergara, O. & Pinto, R. (2006). A Combined Representation to Refine the Knowledge Using a Neuro-Symbolic Hybrid System applied in a Problem of Apple Classification. *Proc. 16th IEEE Int. Conf. on Electronics, Communications and Computers (CONIELECOMP-06),* pp. 30-36, ISBN: 0-7695-2505-9, Puebla, México, Feb 2006, IEEE Computer Society.

David, R. & Alla, H. (2005). *Discrete, continuous and Hybrid Petri nets*, Springer- Verlang, ISBN: 978-3-540-22480-8, Berlin Heidelberg.

He, X.; Chu, W. & Yang, H. (2003). A new Approach to Verify Rule-Based System Using Petri Nets. *Information and Software Technology*, Vol.45, No. 10, 663-669, ISSN 0950-5849.

Knauf, R.; Gonzalez, J. & Abel, T. (2002). A Framework for Validation of Rule-Based Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, part b: cybernetics, vol. 32, no. 3, (Jun 2002), pp. 281-295, ISSN: 1083-4419.

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE,* pp. 541-580, ISSN: 0018-9219, Illinois, Chicago, USA, April 1989.

Nazareth, D. (1993). Investigating the Applicability of Petri Nets for Rule-Based Systems Verification. *IEEE Transactions on Knowledge and Data Engineering*, vol. 4, no. 3, (June 1993), pp. 402-415, ISSN: 1041-4347.

Nazareth, D. & Kennedy M. (1991). Verification of Rule-Based Knowledge Using Directed Graphs. *Knowledge Acquisition*, vol. 3, 339–360.

Negnevitsky, M. (2005). *Artificial Intelligence. A guide to Intelligent Systems*, Second Edition. ADDISON WESLEY, ISBN: 0-321-20466-2, University of Tasmania.

Nikolopoulos, C. (1997). *Expert Systems*. MARCEL DEKKER INC, ISBN: 0-8247-9927-5, Bradley University, Peoria Illinois.

Ramaswamy, M.; Sarkar, S. & Chen, Y. (1997). Using Directed Hypergraphs to Verify Rule-Based Expert Systems. *IEEE Transactions on Knowledge and Data Engineering,* vol. 9, no. 2, 221-237.

Ramirez, J. & De Antonio, A. (2001). Checking Integrity Constraints in Reasoning Systems based on Propositions and Relationships. *Proceedings of the 13th International Conference on Software Engineering and Knowledge Engineering, SEKE.01*, pp. 188-196, Buenos Aires, Argentina, June 2001.

Ramirez, J. & De Antonio, A. (2007). Checking the Consistency of a Hybrid Knowledge Base System. *Elsevier Science. Knowledge-Based Systems*, Volume 20, Issue 3, April 2007, 225-237, ISSN: 0950-7051.

Santos, F. (1998). INSS - Un Système Hybride Neuro-Symbolique pour l'Apprentissage Automatique Constructif, PhD Thesis, LEIBNIZ-IMAG, Grenoble – France.

Tsai, W.; Vishnuvajjala, R. & Zhang, D. (1999). Verification and Validation of Knowledge-Based Systems. *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, January/February 1999, 202-212, ISSN: 1041-4347.

Vermesan, A. (1998). Foundation and Application of Expert System Verification and Validation, In: *The Handbook of Applied Expert Systems*, Jay Liebowitz, Ed. CRC Press LLC, pp. 5.1–5.32, CRC Press, Inc. Boca Raton, ISBN:0849331064, FL, USA.

Villanueva, J.; Cruz, V.; Reyes, G. & Benítez, A. (2006). Extracting Refined Rules from Hybrid Neuro-Symbolic Systems. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'06)*, pp. 3021-3025, ISBN: 0-7803-9490-9, Vancouver, Canada, July 2006.

Wu, C. & Lee S. (1997). Enhanced High Level Petri Nets with Multiple Colors for Knowledge Verification/Validation of Rule-Based Expert Systems. *IEEE Trans. on Systems, Man, and Cybernetics*, Part B. vol 27, no. 5, October 1997, 760-773, ISSN: 1083-4419/97.

Wu, C., & Lee, S. (2000). A Token-Flow Paradigm for Verification of Rule-Based Expert Systems. *IEEE Trans. on Systems, Man, and Cybernetics*, Part B. vol. 30, no. 4, August 2000, 616-624, ISSN: 1083-4419/00.

Wu, Q.; Zhou, C.; Wu, J. & Wang, C. (2005). Study on Knowledge Base Verification Based on Petri Nets. *International Conference on Control and Automatization (ICCA2005)*, ISBN: 0-7803-9137-3/05, Budapest, Hungry.

Yang, S.; Lee, A.; Chu, W. & Yang, H. (1998). Rule Base Verification Using Petri Nets. in *Proccedings 22-nd Annual Int. Computer Software and Applications Conf. (COMPSAC-98)*, ISBN: 0-8186-8585-9, pp. 476-481, Vienna, Austria, august.

Yang, S.; & Tsai, J. & Chen, C. (2003). Fuzzy Rule Base System Verification Using High-Level Petri Nets. *IEEE transactions on knowledge and data engineering*, vol. 15, no. 2, March/April 2003, 457-473, ISSN: 1041-4347.

Petri Nets Applications

Edited by Pawel Pawlewski

ISBN 978-953-307-047-6

Hard cover, 752 pages

Publisher InTech

Published online 01, February, 2010

Published in print edition February, 2010

Petri Nets are graphical and mathematical tool used in many different science domains. Their characteristic features are the intuitive graphical modeling language and advanced formal analysis method. The concurrence of performed actions is the natural phenomenon due to which Petri Nets are perceived as mathematical tool for modeling concurrent systems. The nets whose model was extended with the time model can be applied in modeling real-time systems. Petri Nets were introduced in the doctoral dissertation by K.A. Petri, titled "„Kommunikation mit Automaten" and published in 1962 by University of Bonn. During more than 40 years of development of this theory, many different classes were formed and the scope of applications was extended. Depending on particular needs, the net definition was changed and adjusted to the considered problem. The unusual "flexibility" of this theory makes it possible to introduce all these modifications. Owing to varied currently known net classes, it is relatively easy to find a proper class for the specific application. The present monograph shows the whole spectrum of Petri Nets applications, from classic applications (to which the theory is specially dedicated) like computer science and control systems, through fault diagnosis, manufacturing, power systems, traffic systems, transport and down to Web applications. At the same time, the publication describes the diversity of investigations performed with use of Petri Nets in science centers all over the world.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen