Conference Paper

A Security Technique for Censor Production Rules-based Systems

Nabil M. Hewahi

Department of Computer Science, College of Information Technology, University of Bahrain

Abstract

Censor production rules systems are those systems that should respond within the time period. If the system does not respond within the time limits, a catastrophe might occur. Censor production rules are designed in a way that if more time is given to the system, the system can check more conditions called censor conditions to take the most proper action, if not, the system has to take the proper action without checking further censor conditions. Such a kind of systems can be used in defence, fires fights, patient emergency rescue, hurricanes, disasters and many other applications. One very important issue is how to protect the censor conditions from any intruder change to maintain the right response of the system. In this article, the author presents a technique that ensures that the censor conditions are more secure and cannot be altered easily.

Keywords: Security, Censor Production Rules, Real Time Systems

Corresponding Author:
Nabil M. Hewahi
nhewah@uob.edu.bh

Received: 18 September 2018
Accepted: 10 October 2018
Published: 15 October 2018

Publishing services provided by Knowledge E

Selection and Peer-review under the responsibility of the Sustainability and Resilience Conference Committee.

1. Introduction

Censor Production Rules (CPRs) are rules that are used in decision making in real time systems. CPR is an extension of commonly used rule or what so called the standard rule structure of the form IF<condition> THEN <action>. CPR has been introduced by Michalski and Winston[14] to make rule based systems to deal with real time systems. The CPR structure is as the standard rule structure but with UNLESS part. The UNLESS part contains censors, these censors are as conditions but can only be checked if there is still time to get the system respond. CPR has the following structure

IF <condition> THEN <action> UNLESS <censors>

The <condition> can have one condition or more, <action> is having one action and <censors> is having one or more censor. As an example, let us assume the following rule

IF c_1,c_2,..c_n THEN a UNLESS cc_1,cc_2,..cc_m A:B

$\mathbf{c}_i$ is a condition, a is the action and $\mathbf{cc}_j$ is a censor condition. **A** is the certainty value for obtaining the action **a** if all the conditions are true. **B** is the certainty value for also considering some censors based on the given time. The more time is given, the more censors are checked and the more certain we are from out conclusions. Censors usually rarely occur. For **a** to be achieved all the conditions need to be true, and if we have more time, we can check the censors one by one as far as the time permits, if any censor is true, the action will not be taken. For example

IF It is holiday THEN Sam will go to mounting UNLESS Sam is sick:0.02, Sam has a guest:0.05: 0.8

The above rule means if it is holiday, Sam will go to mounting with certainty value 0.8. If we have time, the system can check whether Sam is sick or not, if Sam is not sick, the overall certainty becomes 0.8+0.02 = 0.82. value of 0.82 is value of **B** which is changeable based on the checked censors. If Sam is sick, it means the conclusion of Sam will go to mounting will not be achieved. This means the more time we have, the more certain we are by having an ability to check more censors. A more realistic example that can be applied on real time systems is

IF Enemy jet is detected THEN launch a rocket UNLESS admit communication 0.25 : 0.9

This means if an enemy jet is detected then launch a rocket. If we still have time we can check if there a communication between the pilot and our forces.

CPRs are very important and has many extension and applications. Baharadwaj and Jain [1] extended CPR to handle general as well as specific conclusions and proposed what so called Hierarchical Censored Production Rules (HCPRs) tree. As an extension of HCPR, Hewahi [6] proposed a rule structure called General Rule Structure (GRS) that can direct the system where to go in case rule fails. Jain and others [10] proposed a new extension to their work and proposed Extended Hierarchical Censored Production Rules (EHCPR) to allow it to be easily represented using semantic networks and frames. Hewahi [9] suggested some modifications on CPRs and proposed a rule called Concept Based Censor Production Rule (CBCPR) in which in each rule there is something called concept that indicates what is the main concept of this rule. The idea is that in various cases many rules might have the same condition, but produce different action depends on the rule context and its main concept. If the user specifies his/her concept, rules with other concepts will not fire even if their conditions match, this will reduce the response time because the system does not need to fire unnecessary rules. Various extensions of HCPRs to show its importance and applications other than those mentioned above are published, some of these works are presented in [1-5,7-8,10-13].

CPRs and its variations can be used in many applications related to real time system. In most of real time systems the input is coming through hardware sensors or other devices connected directly or indirectly with the system that uses the CPRs. The main problem here is that if any modification happens to the input value related to the condition or a censor condition, a catastrophe might occur. For example in the case of the enemy jet rule, if the system is still having more time to respond, assume "there is no communication with the pilot" but through intrusion this input has changed to "there is a communication with the pilot", this will stop launching the rocket, which might lead to target destruction aimed by the enemy.

In this paper we propose a method that might help in increasing the security of CPRs and mitigate the intrusion that could cause the collapse or misbehavior of systems designed based on CPRs.

2. The Problem and the Proposed Approach

The main security problem of real time systems based on CPRs is the capability of changing the values of censor values which will yield to changing the action of CPRs which might lead to a drastic or destructive results/actions. changing the input value of a main condition or a censor condition is crucial in diverting the rule or the system decision. In our case, we shall consider changing the value of the censor condition rather than talking about main condition, this is due to two reasons, the first is that what is applicable on the censor conditions can be applied on the main condition and the second is that the concept of real time systems is based on checking more censor conditions if time permits.

As usual, the inputs of the censor conditions are obtained though hardware sensors. The change of the censor values can be performed through two ways, firstly, changing the result of the hardware sensor and instead of providing a value of "on" to the censor condition, it is given as "off" for example. This problem is a software problem. The second problem is a hardware problem which will cause the sensor to work improperly for a reason or another. In this research we consider only the software issue. To discuss this issue, we state first the method that might cause the change of the values of the censor condition. To change the censor condition values, the intruder can get in to the system and change the value of the censor condition obtained from the hardware sensor as shown in Figure 1.

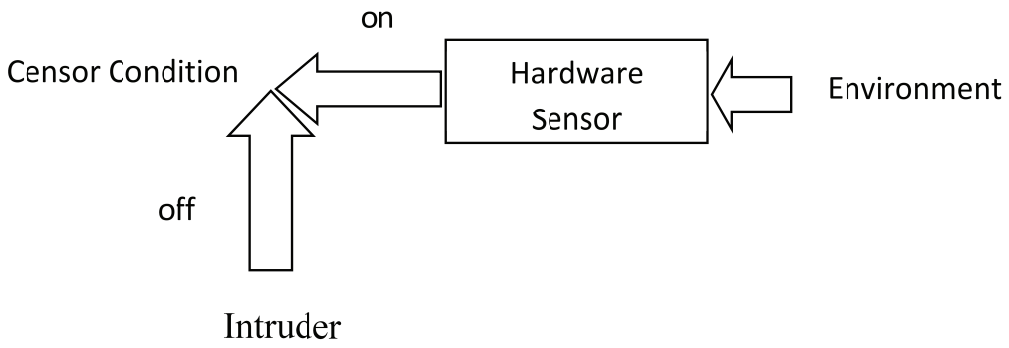

Figure 1: How the intruder might get in to the system.

2.1. The proposed approach

In this section we present the proposed approach for securing systems based on CPRs.

2.1.1. Proposed approach overview

The main idea is to ensure that the value of the output of is the hardware sensor which will be the value of the input of the censor condition has not been changed. To do so we need to include with the developed system a piece of software that can detect whether the value of the censor input has been changed or not as shown in Figure 2. We shall call this software **SD.** This software needs to check whether the output of the hardware sensor is as the value submitted to the censor as a value. The idea is that the intruder does not change the output of the hardware sensor but changes the input of the censor condition. Another issue which is of main concern is that prevention of the **SD** itself from the changes that might occur through the intruder. The overall process of security would involve steps presented in Figure 3.

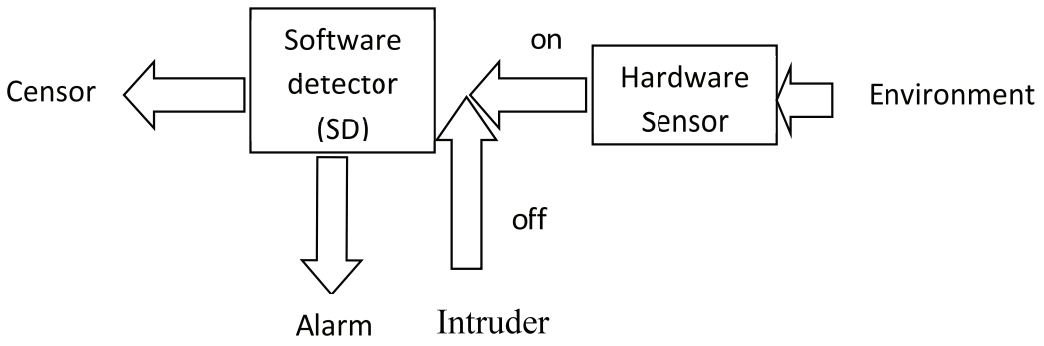

Figure 2: How detection is performed.

The security process follows the steps below:

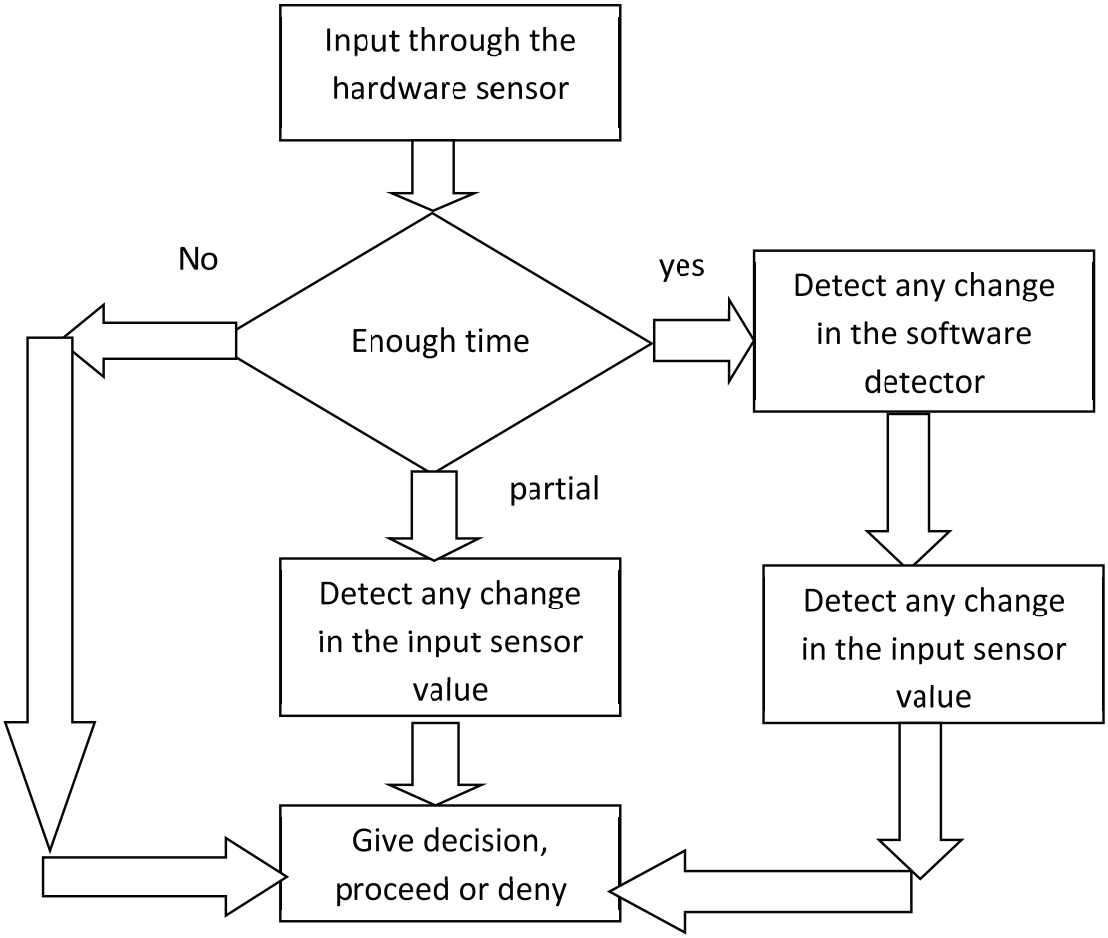

Figure 3: The process of intruder detection.

1. Get the input value through the hardware input sensor

2. Check whether there is enough time to proceed or time limits prevents the process of security check.

3. If time limit allows complete security check (enough time value is "yes"), check first to ensure that the software detector SD hasn't been changed, and then check that the hardware sensor input hasn't been changed. If everything is perfect, it means no intrusion and a decision based on the value of the hardware sensor passed as input value to the censor condition is considered to take a rule/system decision. If there is intrusion, the system will alarm the user if there is enough time to do so, otherwise, the system will take a decision based on the type of detection.

4. If time limit does not allow any of security checks (enough time value is "no"), the system/rule decision will be taken based on the sensor input. Because this is a

crucial decision, it is regularly needed to check that the intruder has not changed the security checking software itself.

5. If time limit allows only a partial check (enough time value is "partial"), a check is performed only on the value of the input hardware senor whether it has been changed or not. If it has been changed, the input of the hardware sensor will be taken directly and given to the system to take the proper decision (alarm or take the decision based on time limits). If not, allow the system to take the decision based on the sensor input.

2.1.2. Hacking software detector

One of the very important steps is to ensure that the intruder hasn't changed the SD used to detect whether the hardware sensor input has been changed or not. To resolve this problem, one solution is to have several copies of this software on various servers even in those that do not use this software. In this case, the more time we have, the software copy on the problem working server can be compared with other copies in other servers. This process is depicted in Figure 4.

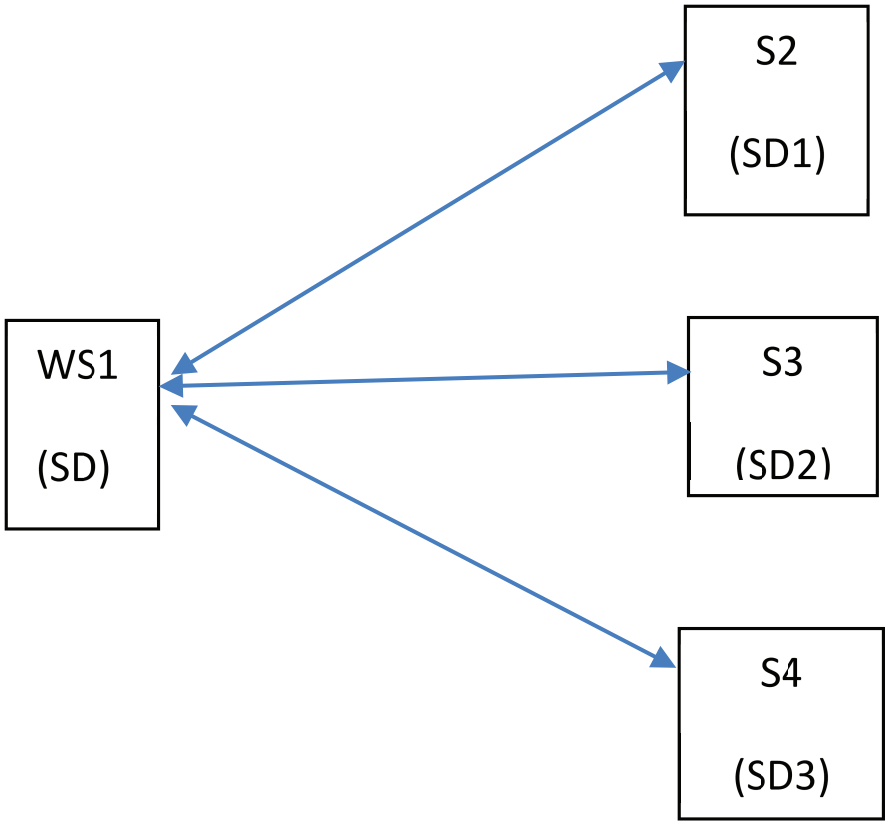

Figure 4: The Process of ensuring no change in the software detector. WS1 is the working server. S_i is the i^{th} server number having the same copy of SD of the working server.

2.1.3. Change detection in hardware sensor input

Since the hardware sensor is input device for the censor conditions, our assumption is that the obtained and produced output given by this hardware device is correct. The trouble might come from changing the sensor input through hacking by injecting a software that changes the value of the sensor input and therefore making the real time system to take improper decision. To solve this problem when the real time system receives the input from the sensor as input for its censor condition, the system checks directly the value of the input sensor value, if it is as received, the system will proceed with its normal steps, if they are not the same and there is enough time, the system will give alarm to the user and if not, it will take a decision based on the values obtained directly from the hardware sensor. Another solution to this problem is to send and save the input obtained from the sensor to other irrelated servers. This will allow us to check if the values in all servers are as the value in the working server if time permits. Also, if time permits, a hybrid technique using the two previous methods can be used to ensure that no change has happened. A simple procedure used to detect the change in the hardware sensor input is shown in Figure 5. Another way for solving this problem is having multi hardware sensors for the same task and then compare their obtained results. This could be more cost expensive but still might be faster than other methods using multiprocessor systems. In Figure 5(c) HS1, HS2 and HS3 are similar and used to detect the same input. This last solution might also solve any hardware deficiency and detect the any hardware problem that might cause incorrect input.

3. Conclusion

In this paper we presented an approach to increase the security of systems based on censor production rules that are used in real time systems. In these systems, the decisions are confined with time limits and the more time is available, the more censor conditions are checked. In such kind of systems, hardware sensors are used as input for the censor conditions in the rules. Two main issue are considered, the possibility of changing the input value of the hardware sensor and the possibility of hacking the software that is responsible for checking the change of the input sensor input. Two procedures to solve both the problems are discussed. The problem of checking the change of the hardware input sensor is performed through further getting the input from the sensor directly or sending and copying the sensor input value to several servers/computers not related directly to the working server of the CPRs system.

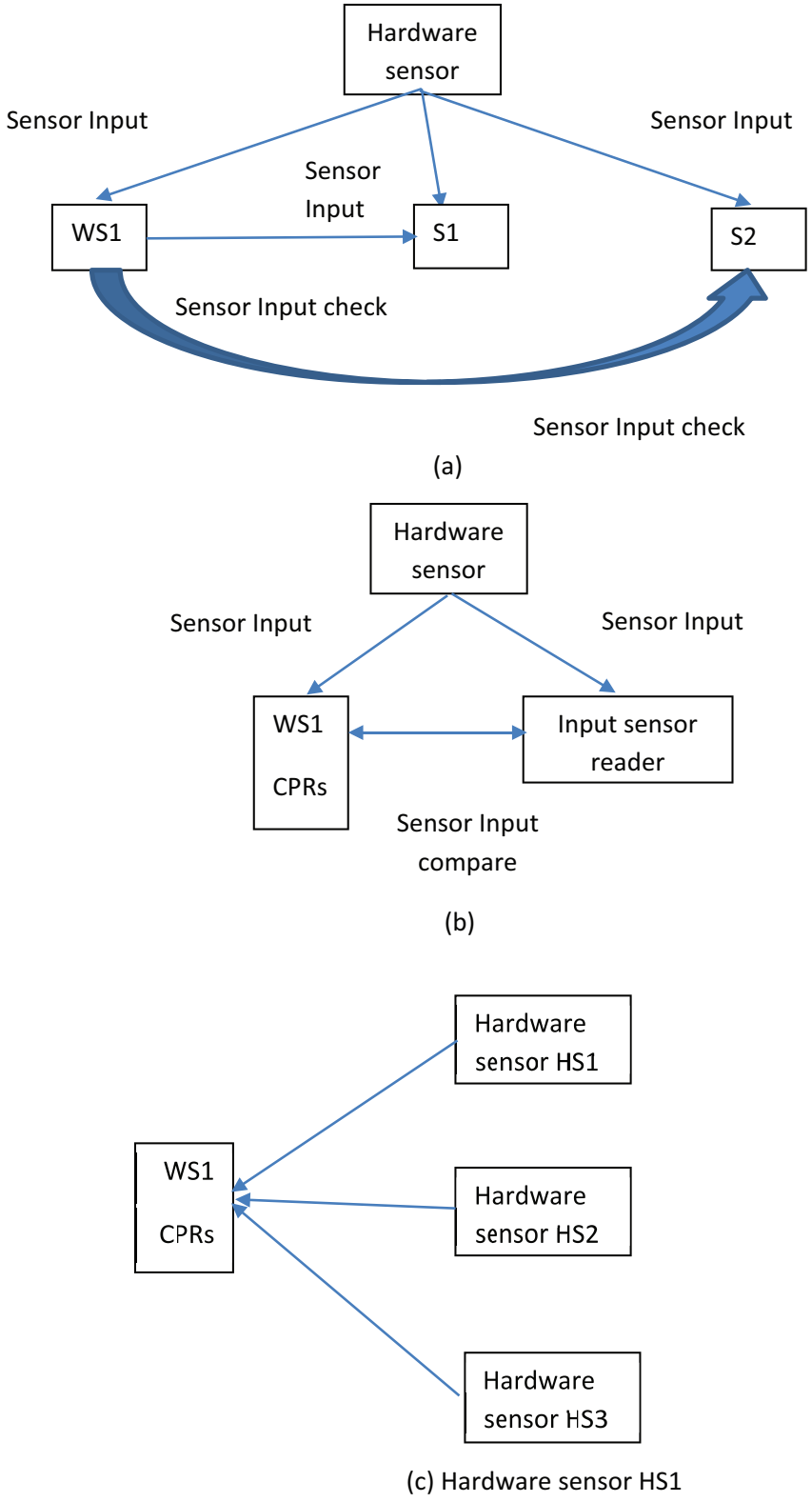

Figure 5: Possible ways to detect any change in the input of the hardware.

The solution for the possibility of hacking the software that checks the possibility of changing the sensor input value is performed by copying the original software to

several irrelated servers other than the working server, another approach which could be faster is to have several copies in various drives on the same working computer server but without using these copies. This last approach could be faster than the previous one, but the first is more secure. Some of the future directions would be focusing on preventing hacking and intruding real time systems based on CPRs rather than focusing on detection.

References

[1] K. Bahradwaj and N. Jain, Hierarchical censored production rules(HCPR) system, Data and Knowledge Engineering, Vol.8,(1992),19-34, 1992.

[2] K. Bharadwaj, N. Hewahi and M. Brando, Adaptive hierarchical censored production rule-based system: A genetic algorithm approach, Advances in Artificial Intelligence, SBIA '96, Lecture Notes in Artificial Intelligence, Springer-Verlag, No. 1159, 81-90. 1996.

[3] K. Bharadwaj and J.Silva, Towards integrating hierarchical censored production rule(HCPR) based system and neural networks", F. Oliveira (Ed.), Lecture notes in Artificial Intelligence, No. 1515, 121-130, 1998.

[4] K. Bharadwaj and R. Varshneya, Parallelization of hierarchical censored production rules(HCPRs) system, Information and Software Technology, Vol.27,No. 8,453-460, 1995.

[5] D. Chuandry and N. Jain, Live EHCPRs system, Conference on Advances in Communication and Control Systems (CAC2S 2013), 2013.

[6] N. Hewahi, A general rule structure, Journal of Information and Software Technology,44, 451-457, 2002.

[7] N. Hewahi, Credit assignment scheme for general rule structure, Proceedings of the 1st Indian Inter. Conf. On Artificial Intelligence, Hyderabad, India,833-842,2003.

[8] N. Hewahi, Principles on integrating General Rule Structure(GRS) based systems and neural networks, Proceedings of the 2004 International Conference on Artificial Intelligence (IC-AI'04), Las Vegas, Nevada, USA, June 21-24,2004.

[9] N. Hewahi, Concept Based Censor Production Rules, International Journal of Decision Support System Technology, vol. 10, issue 1, 59-67, 2018.

[10] N. Jain, K. Bharadwaj and N. Marranghello, Extended of hierarchical censored production rules (EHCPRs) system: An approach toward generalized knowledge representation, Journal of Intelligent System, 9(3, 4), 259-295, 1999.

[11] S. Jain, N. Jain and S. Mishra, Localization of EHCPRs System in the Multilingual Domain: An Implementation, Singh et al. (Eds.), Springer-Verlag Berlin, 314-316, 2011.

[12] S. Jain and N. Jain, Learning techniques in Extended Hierarchical Censored Production Rule (HCPRs) System, Artificial intelligence Review, vol.38, no.2, 97-117,2012.

[13] S. Jain, N. Jain and S. Mishra (2015), ECHCPRs system as an ontology learning systems, 2nd International Conference on Computing for Sustainable Global Development, 11th - 13th March, 2015.

[14] R. Michalski and P. Winston, Variable precision logic, Artificial Intelligence,29, 121-145, 1986.