2009 International Conference on Computer Technology and Development

ECA rule analysis in a Distributed Active Database

Joselito Medina-Marín*, Gilberto Pérez-Lechuga*,
Juan Carlos Seck-Tuoh-Mora*
*Universidad Autónoma del Estado de Hidalgo
Centro de Investigación Avanzada en Ingeniería Industrial
Pachuca, Hidalgo, México
jmedina@uaeh.edu.mx

Xiaoou Li†
†Instituto Politéctnico Nacional
Centro de Investigación y de Estudios Avanzados
México, D.F., México
lixo@cs.cinvestav.mx

***Abstract*—Active database systems integrate event-based rule processing with traditional database functionality. The model most widely used to represent event-based rules is the Event-Condition-Action rule (ECA rule) model. However, the relationships among rules in the development of a base of ECA rules can fall in an infinite rule triggering: the No termination problem. In this article, an approach based in Petri Net theory is proposed. This approach detects cyclic paths in the base of ECA rules. Furthermore, it can analyze the relationships among ECA rule components.**

***Keywords*-active database; ECA rule; rule analysis;**

I. INTRODUCTION

Traditional databases (DB) were developed to store a huge amount of information. In this DB type the information only was accessed by insert, delete, update and query algorithms, which were previously programmed in a Data Manipulation Language (DML) by the DB administrator. The set of all this data manipulation programs is the Database Management System (DBMS). However, the execution of those programs is performed only by the request of either a DB user or the DB administrator.

Nevertheless, there are systems that cannot be implemented by using a traditional DB approach. Such systems are those where is well known that if certain events occur in the DB and if the DB state satisfies certain conditions, then an action or procedure is performed in the DB. Therefore, it is necessary to use an approach where a DB could have the ability to react automatically when an event occurs either inside or outside DB environment, after this, it can verify the DB state to evaluate conditions, and if condition is evaluated to true it can execute procedures that modify the DB state. In order to provide of active behavior to traditional DB, Active Databases (ADBs) were introduced. If a human being takes charge to detect the event occurrences, verify conditions, and execute procedures instead an ADB system, then the system may not work well. Thus, it is very important to add enough information to DB about the active behavior and convert a traditional DB into an Active one.

Active behavior of a DB can be defined through a base of active rules, which has the specification of events that will be detected, conditions that will be evaluated, and actions or procedures that will be performed in the DB. The model most widely used is the *event-condition-action rule* (ECA rule) model, whose general form is as follows:

On event
If condition
Then action

ECA rule model works in the following way: when an event e_1 that modifies the current DB state occurs, if condition c_1 is evaluated to true against DB information, then either an action a_1 is executed inside DB or a message is sent outside DB.

An event e_1, which can trigger to an ECA rule, can be of two types: *primitive event* or *composite event*. A *primitive event* is generated by the execution of an operation over the DB information (insert, delete, update, or select), a transaction, a clock event (which can be absolute, relative, or periodic), or the occurrence of a DB external event. On the other hand, *composite events* (disjunction, conjunction, sequence, closure, times, negation, last, simultaneous, and any) are formed by the occurrence of a combination of primitive and/or composite events.

Composite events increase the complexity of a base of active rules because composite events are represented by complex structures, which need to be evaluated when a composite event is raised. In the same way that a composite event increases the complexity of a base of active rules, relationships between ECA rules increase the complexity of a base of active rules. In other words, there is a relationship between two ECA rules when the action of one rule r_1 triggers to a rule r_2, at this moment the relationship does not represent a problem, however, when there is a rule r_3 which is triggered by the event generated by the action of rule r_2 and the action of rule r_3 generates the event that triggers to rule r_1, then a rule triggering cyclic is achieved and likely it could be an infinite rule triggering among rules r_1, r_2, and r_3. Infinite rule triggering is known as the problem of No Termination, and it can produce an inconsistent state of DB because consume a lot of compute

978-0-7695-3892-1/09 $26.00 © 2009 IEEE
DOI 10.1109/ICCTD.2009.282

time when it executes infinitely the same instructions.
No Termination problem has been tackled using two types of analysis, by one side, *static analysis* of rules is used (at compile time), which perform the analysis of a base of active rules before its implementation in a DBMS. Static analysis verifies the existence of cyclic paths inside the rule base. On the other side, *dynamic analysis* of rules is used (at runtime), which monitors the cyclic paths that can fall in an infinite rule triggering.
In this work, a termination analysis approach based in Petri Net (PN) theory is proposed. This approach is an extended model of PN, named Conditional Colored Petri Net (CCPN). A CCPN is generated from a base of active rules, and information about rule events, conditions, and actions is stored in the CCPN. An analysis method of PN theory, Incidence matrix, is used to find cyclic paths existing in the CCPN. Those cyclic paths are used to create a set of potential infinite rule triggering. Cyclic path set found in CCPN is analyzed and if can be inferred if any cyclic path never will produce an infinite rule triggering, then those cyclic paths are deleted from the set. Finally, in dynamic analysis a monitoring of cyclic paths set is performed.

II. Conditional Colored Petri Net

Conditional Colored Petri Net (CCPN) is an extended PN model, which was adequate to support the features of an ECA rule model.
Petri Nets are a graphical and mathematical tool for modeling concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic systems. Petri net may be extended widely and applied in every area with logic relations. Their mathematical representation of the modeled system can be used to reveal important information about the system structure and dynamic behavior. Active database is a novel and promising application area of Petri nets. Up to now, few researches have adopted Petri nets as ECA rule specification language [1], [2], [3],[4]. SAMOS is a successful ADB system, which partially uses Petri nets for composite event detection and termination analysis. But, the framework is not Petri-net-based[5].
In our previous work, the CCPN was introduced for modeling and simulation of active database behavior and its corresponding implementation was realized too. In this article an enhanced CCPN model is presented, which currently support both composite and primitive events.

A. CCPN description

In a CCPN, ECA rule event e is stored as a place p_1, conditional part c is stored inside a transition t, and the action rule a, because of its similarity to an event, is stored in a place p_2. Therefore, if t is the transition where the

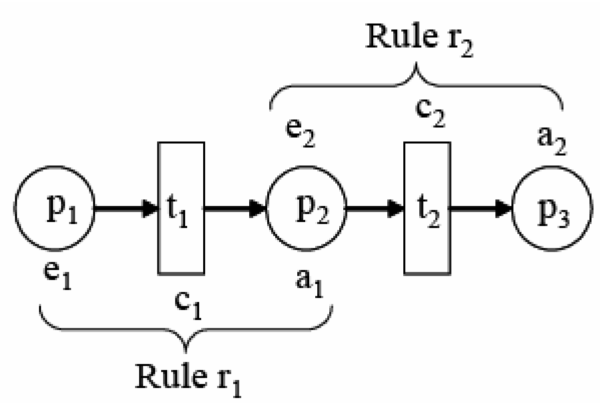

Figure 1. Relationship between the action of rule r_1 and the event of rule r_2.

condition of rule r is stored, then ${}^\bullet t = \{p_1\}$, and $t^\bullet = \{p_2\}$, where ${}^\bullet t$ is the set of the input places of t, and $t^\bullet$ is the set of the output places of t.

In a CCPN it is very easy to detect the existence of both relationships and dependencies between two or more rules according to its graphics representation. Some of the ECA rule models presented in the related work does not consider directly these relationships, they use both the triggering graph and activation graph to view them. Moreover, relationships viewed by using triggering and activation graph are only viewed in a rule level. On the other hand, in the approach presented by this paper, relationships can be viewed in the same model where ECA rules are represented. Furthermore, existing relationships among rules can be viewed as relationships among ECA rule elements (event, condition, action). Figure 1.

Moreover, both primitive and composite events can be modeled with the CCPN model.

During CCPN execution, the events that occur in the DB can be detected by the CCPN, and if there is a CCPN place p_1 which represents to the detected event e then a token is generated with information about the event (e.g. the record of an employee) and with a timestamp according to the time when the event was raised. By CCPN execution, the new token is sent to transition t_1, ${}^\bullet t_1 = \{p_1\}$, and the condition c stored in t_1 is evaluated against token information. If token information is not enough to evaluate c then a query to BD is executed to know the DB state and perform the evaluation to c. If c is evaluated to true then one other token with information about the rule action a is generated and it is sent to place p_2, $t_1^\bullet = \{p_2\}$, which represents the ECA rule action a.

Composite events that deal with time interval evaluate the timestamp of tokens, and if the timestamp belongs to the composite event interval, then the token is sent to its

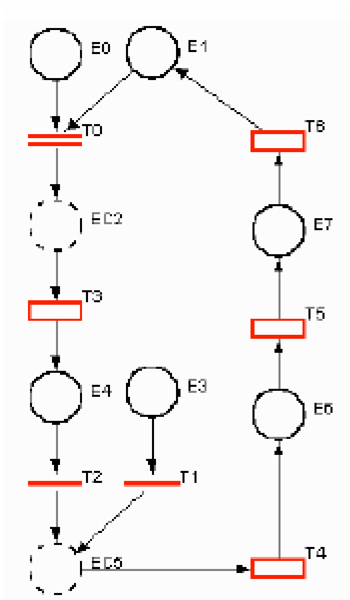

Figure 2. CCPN obtained from the base of four ECA rules.

corresponding transition.

B. *Modeling ECA rules with CCPN*

In order to show the modeling of a base of active rules as a CCPN, four ECA rules are converted into a CCPN, whose description is as follows:

Rule 01: When an employee is inserted in the office DB and the production of employee's department is modified, if the production is greater than $900.00, then the employee's bonus is updated to $100.00.

Rule 02: When either salary or bonus of an employee is modified, if the salary is increased by more than $200.00 or the bonus is increased by more than $50.00, then the employee's rank is increased too.

Rule 03: When the employee's rank is updated, if rank value is greater than 15, then the employee's department budget is added with $1000.00

Rule 04: When a department budget is modified, if the budget is greater than $20,000.00, then the department production is increased 3%.

Definition of tables needed to this rules are as follows:

`DEP(TheDep,Production,Budget).`

`EMP(ItsDep,TheEmp,Salary,Bonus,Rank).`

CCPN obtained from the rules listed above is showed in figure 2.

III. Termination analysis

An important topic in active database design is the No termination analysis, which appears from the relationship between a set of active rules and each element of the set fires to another, i.e., from an active rule set $S = \{r_1, r_2, r_3, \ldots, r_n\}$, when action of rule r_1 enables the fire of rule r_2, action of rule r_2 enables the fire of rule r_3, an so on, finally, action of rule r_n, enables the fire of rule r_1. This process performs an infinite rule triggering that uses a huge amount of compute time, and the database system becomes instable.

Therefore, it is necessary to determine if the infinite rule triggering in a ADB finishes or not, in order to avoid inconsistent states in the DB. Thus, a new approach to detect termination problem is presented in this paper. This approach uses the incidence matrix of PN theory, which offers enough information about the system that is being modeled via PNs. Even though, in this case, an active rule base is being modeled by a CCPN, the incidence Matrix obtained is similar to those which are obtained from a pure PN.

In incidence matrix, places are represented by its columns and transitions are represented by its rows, so it is possible identify both the initial and the final nodes of CCPN.

IV. Algorithm

The algorithm that perform the termination analysis by using the CCPN model is as follows:

Step 1.- Convert a base of ECA rules into a CCPN graph.

Step 2.- Create the incidence matrix from the CCPN.

Step 3.- Search all the paths of CCPN

Step 4.- Create a set CP_{set} of cyclic paths CP.

Step 5.- Delete from CP_{set} those cyclic paths that satisfy the theorems conditions.

Step 6.- Cyclic paths that even stay in CP_{set} are analyzed in a deeper level, i.e., each place $p \in \{p \mid p \in^{\bullet} t_2,\ p \in t_1^{\bullet},\ (t_{1,2}, p) \in CP, CP \in CP_{set}\}$ is checked to verify if it always will trigger to t_2. If there is, at least, one place p that, according to information sent from transition t_1, t_2 does not trigger then rule firing finishes and CP is deleted from CP_{set}.

At runtime, CP_{set} is monitored to avoid an infinite rule triggering, and in consequence an instable database system.

V. Example

To show the feasibility of CCPN model in the detection of No termination problem in rule triggering, the example listed above is used to perform the analysis.

The incidence matrix generated from CCPN of figure 2 is presented in figure 3.

It can be observed that there is a cyclic path constituted by the elements (3,2), (3,4), (2,4), (2,5), (4,5), (4,6), (5,6), (5,7), (6,7), (6,1), (0,1), (0,2), (3,2). Nevertheless by theorems definitions presented in this work the presence of a cyclic path does not mean an infinite rule triggering, so it can be eliminated according to theorem 3, because of the presence of composite event *"conjunction"*. Therefore, in this case there is not an infinite rule triggering for this base of ECA rules.

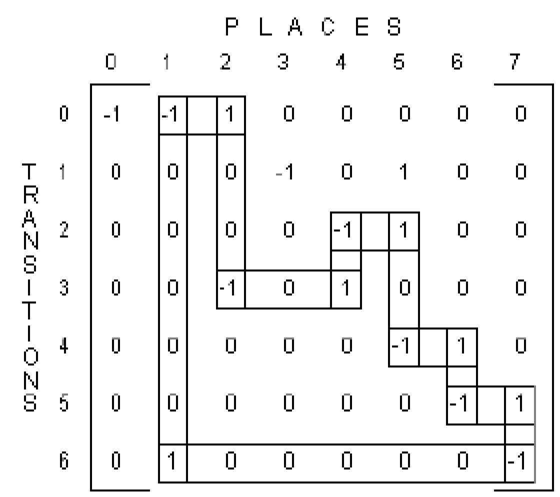

Figure 3. Incidence matrix obtained from CCPN of figure 2.

VI. Conclusion

An approach to detect the No termination problem was developed in this article, which is based on an PN extension named Conditional Colored Petri Net (CCPN). CCPN stores enough information about a base of ECA rules, such as its events, conditions, and actions. Furthermore, CCPN can model both primitive and composite events, which are useful in the detection of false infinite rule triggering inside a cyclic path. Cyclic paths are found by using the incidence matrix of PN theory, which are analyzed taking into account the set of theorems presented.

Unlike approaches presented in related work section, this approach is better than those in the following aspects:

- Both ECA rule representation and ECA rule analysis are performed in the same CCPN model.
- CCPN supports composite events.
- This approach goes beyond a simple analysis of cyclic paths in a graph because it analyzes each element of the CCPN graph to determine if the rule triggering in a cyclic path will finish.
- CCPN model can be used for dynamic analysis at runtime.

As future work, ECA rule analysis based in CCPN model will be implemented in the ECAPNSim interface, which was developed by the authors of this article to give an active behavior to a passive DB.

References

[1] Zimmer, D.: ”Rule Termination Analysis based on a Rule Meta Model”. In: Cadlab Report 2, Cadlab, Bahnhofstr. 32, 33102 Paderborn, Germany, April 1995

[2] Kokkinaki, A.I.: ”On using Multiple Abstraction Models to Analyze Active Databases Behavior”. In: Biennial World Conference on Integrated Design and Process Technology, Berlin, Germany, June, 1998.

[3] Schlesinger, M. and Lorincze, G. : ”Rule modeling and simulation in ALFRED”. In: The 3rd International workshop on Rules in Database (RIDS’97) (or LNCS 1312), Skovde, Sweden, June, pp. 83-99, 1997

[4] Guisheng, Y., Qun, L., Jianpei, Z., Jie, L., Daxin, L.: ”Petri Based Analysis Method For Active Database Rules”. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 2, 1996, pp. 858-863

[5] Gatziu S., Dittrich K.R., ”Detecting Composite Events in Active Database Systems Using Petri Nets”, Proceedings of the 4th International Workshop on Research Issues in Data Engineering: Active Database Systems, Houston, Texas, February 1994

[6] Li X., Medina Marn J., ”Composite Event Specification in Active Database Systems: A Petri Net Approach”, IEEE International Conference on System, Man, and Cybernetics, The Hague, The Netherlands, Oct, 2004.