Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 6 (8): 1738-1744, 2006
ISSN 1812-5654

Termination Analysis Approach and Implementation

L. Baba-hamed and H. Belbachir
Laboratory: Systems, Networks and Data bases, USTO, B.P. 1505 El Menouar, Oran, Algeria

Abstract: An active database system is a system which provides the same functionalities as a classic database system and is, equally, capable to react, automatically, to state changes by means of rules said active rules. The triggering of these rules can produce an infinite cycle and leads to the no termination problem. In this study, we propose a method of termination analysis of active rules based on Petri nets (PN) and give an object oriented representation to implement it. This approach is better than the previous ones because it takes into a count composite events and the rule priority on the one hand and both rule representation and rule analysis are performed in the same PN on the other hand.

Key words: ECA rule, termination, petri net, priority, path, class

INTRODUCTION

Active database systems (Widom and Ceri, 1996; Paton and Diaz, 1999) aim at the representation of more real-world semantics in the database by supporting event-condition-action rules (ECA-rules). ECA-rules can be interpreted as "when the specified event occurs and the condition holds, execute the action". An event indicates the point in time when some sort of reaction is required from the DBMS. For primitive events, this point in time can be specified by an occurrence in the database, by an occurrence in the DBMS, or by an occurrence in the database environment. For composite events (Chakravarty and Mishra, 1994; Gatziu and Dittrich, 1994; Gehani *et al.*, 1992), the point in time is defined on the basis of other points in time which represent other primitive and/or composite events (called component events). These components are combined by means of event constructors as: negation, conjunction, disjunction, sequence etc. The action describes treatments to achieve when a specific event happens and some condition holds.

The active rule behaviour is difficult to predict and require the development of techniques to analyse properties of a set of rules automatically. One of these important properties is the property of the rule termination. Many works have tried to solve the no termination problem; nevertheless, most of them present insufficiencies.

The aim of this study is to propose a rule termination analysis approach which considers the impact of both the composite events and the rule priority on the termination analysis problem. This method detects cyclic paths in the base of ECA rules, can analyse the relationships among ECA rule components and detects cases of termination which are not detected by the other approaches.

Among all methods which proposed to study rule termination, some are based on models of graphs and others use formal basis as the systems of rewrite or the Petri nets. We present some of them in the following.

Aiken *et al.* (1995) are the first to introduce the notion of Triggering Graph (TG). They show that a triggering graph without cycle determines and guarantees the termination of a set of active rules in the context of relational systems. Baralis *et al.* (1996) grouped the active rules into modules, termination of rule execution, within each module is assumed and inter-module termination is analysed. It is the only method that presents a modular conception of the active rules. Baralis *et al.* (1998) proposed a technique that exploits the information of the graph TG and other graph called Activation Graph (AG) to analyse the termination of a set of ECA rules. This analysis uses an algorithm called algorithm of reduction. This approach presents an inconvenience because it doesn't propose a method of building the AG graph which is not obvious. Baralis and Widom (2000) try to improve these last methods. Their approach is based on a "propagation algorithm", which uses an extended relational algebra to accurately determine when the action of one rule can affect the condition of another. The termination analysis is made by building the graph AG. Their study is considered as a complementary method for the two last one. Lee and Ling (1998) propose a path technique for reducing the graph TG. The method considers together the conditions of long triggering sequences called activation formulas. It is necessary to

Corresponding Author: L. Baba-hamed, Laboratory: Systems, Networks and Data Bases, USTO, B.P. 1505 El Menouar, Oran, Algeria

guarantee that the execution of rules outside the triggering sequence cannot unpredictably change the database state. Hence, only non-updatable predicates can be included in the activation formula. This condition severely limits the applicability of the technique. Bailey *et al.* (1997) use abstract interpretation for the termination analysis of active rules. The idea is to reason about sequences of database states using "approximate semantics" and to use the fix point computation (over a lattice) to handle cycles. This approach is applicable to a simple and restricted rule language. A different approach is taken in (Karadimce and Urban, 1994), where ECA rules are reduced to term rewriting systems and known analysis techniques for termination of term rewriting systems are applied. The analysis is based on an object-oriented data model and instance-oriented rule execution model. This approach is powerful, since it exploits the body of work on Conditional Term Rewriting System, but its implementation appears to be complex even for small rule applications. Kokkinaki (1998) uses Parameterised Petri Nets (PPN) to analyse the active rule termination in the relational model. A PPN is a Petri Net (PN) whose places are parameterised. The firing of the transition corresponds to the rule execution. If there is no cycle in the PPN model then the rules execution in the ADBS must terminate. The inconvenience of this approach is that, the use of PPN in modelling complex systems results in complex graphic representations which are very difficult to be conceptualised and handled. Other PN based method is presented in (Zimmer *et al.*, 1996). To represent the triggering and activation notions in the PN, the authors give for each rule two subnets Ei (for the triggering) and Ci (for the activation). The authors detect a non-terminating behaviour of rules using a coverability graph based on the reachability graph. In this approach, the conception of the PN is too complex. In addition, the presence of a cycle in the PN does not imply that will occur an infinite rule triggering. Li and Marin (2004) present an approach based on coloured Petri nets named CCPN (Conditional Coloured Petri Net) for modelling the active database behaviour. Incidence matrix of PN theory is used to find cyclic paths existing in the CCPN. Cycles which satisfy some theorems given by the authors are deleted. If there is no cycle in the CCPN, the termination of the corresponding set of rules is guaranteed. Nevertheless, this approach does not consider the priority of rules.

EXTENDED COLORED PETRI NET (ECPN)

Our approach is inspired by Li and Marin (2004). We improve this method by adding the notion of rules priority and showing how the termination analysis can be affected by this notion.

PN is a graphical and mathematical tool and may be applied in various areas. Active database is a promising application area of PN. Up to now, few researches have used PN as ECA specification language. In SAMOS (Gatziu and Dittrich, 1994; Geppert *et al.*, 1995), PN is partially used for composite event detection and termination analysis.

In our model named ECPN, the rule event is represented by a place p1, the condition c and the priority pr of the rule are attached to a transition t and the rule action a is represented by a place p2. Relationships between the rules can be viewed in the same graph (Fig. 1).

ECPN is a colored PN (David and Alla, 1989; Jensens, 1992) defined as follows:

ECPN = {E, P, T, A, N, C, Con, Action, D,τ, I} where:

E is a finite set of non-empty type, called colour sets. It determines all the data value, the operations and functions that can be used in the net expressions.

P is a finite set of places. It is divided into four subsets: P_{prim}, P_{comp}, P_{virt} and P_{copy}. P_{prim} represents the set of primitive places and correspond graphically to a single circle. P_{comp} represents the set of composite events. P_{comp} includes the following events: negation, sequence, closure, last, history and simultaneous. They correspond graphically to a double circle. P_{virt} represents the set of composite events which includes the conjunction, disjunction and any. They correspond graphically to a single dashed circle. P_{copy} is the set of places which are used when two or more rules are triggered by the same event. They correspond graphically to a double circle where the interior circle is a dashed one.

T is a finite set of transitions; it is divided into three subsets: T_{rule}, T_{copy} and T_{comp}. T_{rule} corresponds to the set of rules. Each transition of rule type is represented graphically by a rectangle. T_{copy} is the set of transitions of copy type. They are represented graphically by a single

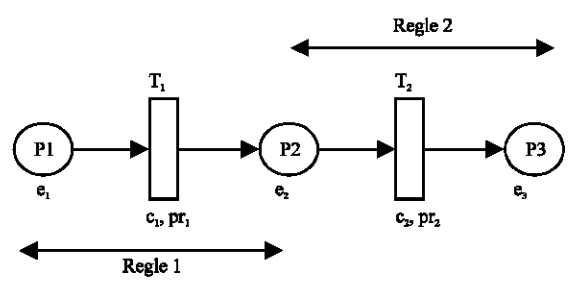

Fig. 1: Relationships between two rules r1 and r2

bar. A copy transition is used when one event e can trigger two or more rules. A copy transition will produce n same events where n is the number of rules which are triggered by the same event e. T_{comp} is the set of transitions of composite type. They are represented graphically by a double bar. A composite transition is used for generating a composite event from a set of primitive or composite events.

A is a finite set of arcs. It is divided into two subsets: The input arcs which are defined from P to T and output arcs which are defined from T to P. Inhibitor arcs are used to represent the Negation composite event.

N is a node function. It maps each arc into a pair where the first element is the source node and the second is the destination node ($N: A \rightarrow P \times T \cup T \times P$).

C is a color function. It maps each place p to a type C(p) (i.e., $C: P \rightarrow E$).

Con is a condition function. It is defined from either T_{rule} or T_{comp} into expression such that:

$$\forall t \in T_{rule}: \text{ Type (con (t))} = B,$$

where Con function evaluates the rule condition.

$$\forall t \in T_{comp}: \text{ Type (con (t))} = B,$$

where Con function evaluates the time interval of t against tokens timestamp. B is used to denote the Boolean type containing the values false and true.

Action is an action function. It maps each rule type transition $t \in T_{rule}$ into a type C(p) which will be deposited into its output place.

D is a time interval function. It is defined from T_{comp} to a time interval $[d_1(t), d_2(t)]$, where $t \in T_{comp}$ and $d_1(t)$, $d_2(t)$ are the initial and the final interval time, respectively. The interval is used by the Con function to evaluate transitions $t \in T_{comp}$.

τ is a timestamp function. It assigns each token in place p a timestamp.

I is an initialization function. It maps each place p into a closed expression which must be of type C(p).

Example of an ECPN: Let's consider the set of rules R1, R2, R3 andR4 expressed according to the following formalism:

Define rule rule-name On event If condition Then action

Define rule R1	
On reduce-salary ()	(e0)
If employee.salary <1500	(T0)
Then raise-salary ()	(e1)
Define rule R2	
On raise-salary ()	(e3 a copy of e1)
If employee.children-nbr >5	(T3)
Then send-bonus ()	(e4)
Define rule R3	
On raise-salary ()	(e2 a copy of e1)
If employee.age >60	(T2)
Then be-retired ()	(e5)
Define rule R4	
On send-bonus ()	(e4)
If employee.salary <10000	(T4)
Then raise-salary ()	(e1)

These rules necessitate the class Employee which has the attributes: id-emp, salary, bonus, children-nbr, age. The ECPN which corresponds to the set rules given above is shown in Fig. 2. Furthermore, we consider that $R1 >_p R3 >_p R2 >_p R4$ where $Ri >_p Rj$ means that Ri has priority than Rj.

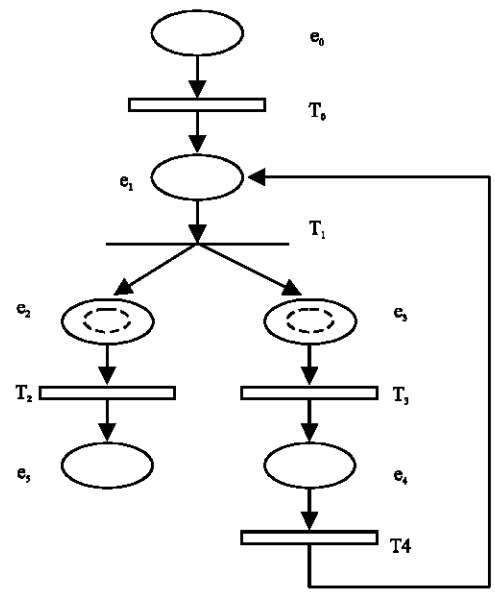

Fig. 2: Example of an ECPN

	e0	e1	e2	e3	e4	e5
T0	-1	1	0	0	0	0
T1	0	-1	1	1	0	0
T2	0	0	-1	0	0	1
T3	0	0	0	-1	1	0
T4	0	1	0	0	-1	0

Fig. 3: Incidence matrix from ECPN of Fig. 2

Termination analysis: Let S = {r1, r2, r3, . . ., rn} be a set of rules, when the action of rule r1 triggers the rule r2, action of rule r2 triggers the rule r3 and so on and finally, action of rule rn, triggers the rule r1, this process performs a cyclic rule triggering and it can produce an inconsistent state of the database when it executes these rules infinitely. To study the problem of rules termination and the impact of rules priority on it, we give an algorithm which uses the incidence matrix, the notions of paths, cyclic paths and acyclic paths of PN theory. In incidence matrix, places are represented by its columns and transitions are represented by its rows, so it is possible to identify both initial and final nodes of an ECPN.

ECPN is a directed graph constituted by a sequence of nodes forming paths, where each node is either a place or a transition in an alternate way. If a cyclic path is found, then it may produce an infinite rule triggering.

A path R is a sequence of pairs (r, c) which are obtained from the incidence matrix of the ECPN, where r and c are incidence matrix indexes.

The sequence of pairs (r, c) describes the connection between places and transitions as follows: (t1, p1), (t1, p2), (t2, p2), ..., (tn-1, pn-1), (tn, pn-1), (tn, pn). The first pair is formed by the transition t1 and its input place p1, following the path, the next pair is formed by the same transition t1 and its output place p2, the third pair is formed by the same place p2 that now is the input place to transition t2 and so on.

Paths search starts from the coordinate of (t1, p1) and then a positive integer is looked for in the row corresponding to t1, finding the coordinate to (t1, p2). After, a negative integer is looked for in the column corresponding to p2, finding the coordinate to (t2, p2). Then another positive integer is looked for in the row corresponding to t2 and so on, until either a terminal node or an existing node in the path is found. A cyclic path CP is a path R where the last pair (r, c) has been already found. An acyclic path AP is a path where the last pair (r, c) is different from each other in AP. If all the paths, in the ECPN, are acyclic then the termination is guaranteed. If there is at least one cyclic path CP in the ECPN, the rule triggering may not finish. But it does not mean that whenever exist a cyclic path CP the rule triggering does not terminate; there are other facts that should be taken into account. For example, in ECPN model, the priority, the condition and the composite events of ECA rules are considered, so they can have an impact on the termination analysis problem. The composite events that affect the termination analysis are: Negation, conjunction, any and sequence. The conditions 1, 2 and 3, given below, correspond to them. The condition 4 concerns the rule condition belonging to a cyclic path.

Condition 1: If a cyclic path CP contains an inhibitor arc i.e., a composite event negation is included in CP, then CP finishes its rule triggering.

Condition 2: For composite events conjunction, sequence and simultaneous, if any of its constituent events is not generated by the action of a rule belonging to CP then the rule triggering finishes.

Condition 3: If composite event any (m, e1, e2, …, en) is a part of a cyclic path CP and if k constituent events of the composite event any are not generated by the action of a rule belonging to CP and n-k < m then the rule triggering finishes.

Condition 4: If the condition of a transition ti $\in$ {t | (t, p) $\in$ CP} is always false according to the event produced by the action of previous rule, then the rule triggering finishes.

Algorithm of the priority: The following algorithm shows the impact of the rules priority on the termination analysis for a set of rules modelled by an ECPN.

Step 1: Convert a base of ECA rules into an ECPN
Step 2: Create the incidence matrix from the ECPN
Step 3: Search all the paths of ECPN
Step 4: Create a set Rset of paths R
Step 5: If Rset contains no cycle.
Then returns "the termination is guaranteed"

Else explore the paths in Rset one by one according to the priority of rules:

```
    termine ← false
    Repeat for each path R
        If R is acyclic
        Then termine ← true
        Else /*R is cyclic*/
            If R satisfies the conditions given above
            Then termine ← true
            Else consider the next path
            End
        End
    Until the end of the paths exploration or termine=true
    If termine = true
    Then returns "the termination is guaranteed"
    Else returns "the termination is not guaranteed"
    End
End
```

Illustrative example: To show the impact of the rules priority on the termination analysis in an ECPN, we consider the example given above. The incidence matrix

generated from the ECPN of Fig. 2 (which corresponds to the example) is presented in Fig. 3. It can be observed that there are two paths constituted by the elements: (0,0), (0,1), (1,1), (1,2), (2,2) and(2,5) for the first one and the elements: (0,0), (0,1), (1,1), (1,3), (3,3), (3,4), (4,4), (4,1) and (1,1) for the second one which includes the cycle. Nevertheless, as R3 has priority than R2, this cycle is not considered and the termination of the set {R1, R2, R3, R4} is guaranteed according to the algorithm. The path to which belongs R3 (represented by the transition T2) is acyclic.

IMPLEMENTATION ISSUES

In order to implement an ADBS, the architecture of a (passive) DBMS has to be augmented by new components like an ECA rule editor, an analyzer of rules, a rule manager, an event detector for primitive and composite events and a rule execution component.

ECA rules are defined by ECA rule editor. After ECA rules are converted into an ECPN, ECPN is saved into ECPN base as places, transitions and arcs. ECPN rule base is used by ECPN rule manager which calls the event detector for detecting events, the rule execution component for evaluating the condition and executing the action of rules. It calls also the termination analyzer component to cheek no-termination problem in ECPN rule base. This last component includes an incidence matrix generator, a paths generator, a cyclic paths detector and an analyzer of paths which take into a count the priority of rules. As the implementation of our passive DBMS is an object-oriented implementation and in accordance to the idea "to stay in the same world and exploit its advantages", we built the rule structure by means of object-oriented features (Atkinson *et al*., 1989). In our model, the ECA rules are represented by transitions of rule type, the conditions and the priorities of rules are attached to these transitions. The events and the actions are modelled by places which are inputs and outputs of a transition respectively. The rule firing corresponds to transition firing. So we need four main classes named: PLACE, TRANSITION ARC and TOKEN.

All places are instances of a class PLACE with attributes index, event-name, event-body, list-rules,

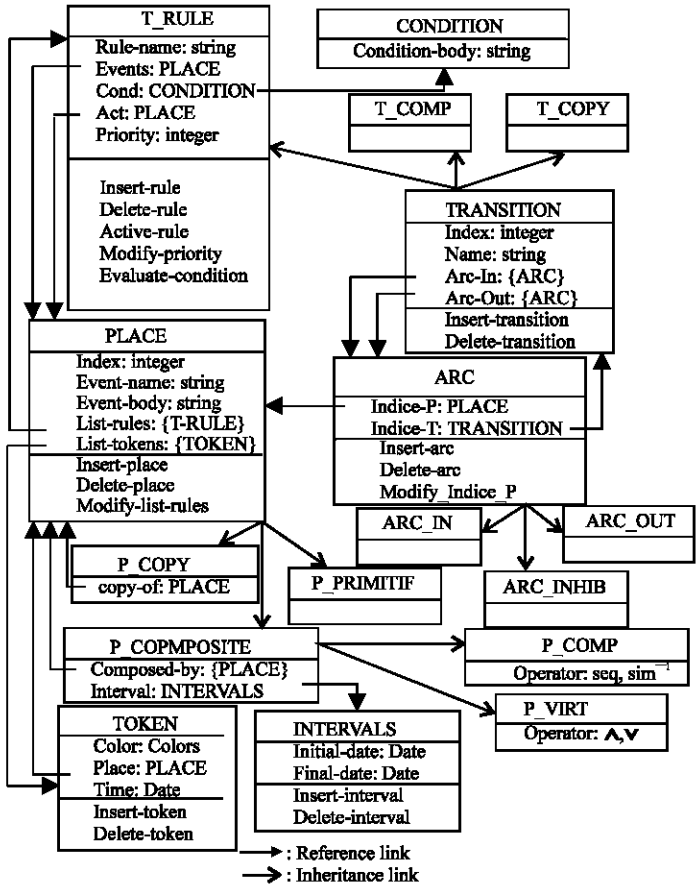

Fig .4: Class hierarchy

list-tokens and methods insert-place, delete-place, modify-list-rules and modify-list-tokens. The index is an internal number assigned by the system. The event-name is the event name. The event-body is the content of the rule event part. The list-rules is the list of references to objects of class T-RULE which are triggered by this event. The list-tokens is a list of references to objects of class TOKEN; these tokens represent the actual marking of the place. Class PLACE is the super-class of the subclasses P-COMPOSITE (which includes the attributes composed-by and interval), P-PRIMITIVE and P-COPY (which includes the attribute copy-of). composed-by holds the events (single or composite) that comprise the event. interval refers to objects of class INTERVALS (i.e., refers to time intervals). copy-of holds the event for which we have made copies.

Each transition is an object of class TRANSITION with attribute index, name, arc-in, arc-out and methods insert-transition and delete-transition. The name is the transition name. The arc-in is the list of references to objects of class ARC (the set of the input arcs). The arc-out is the list of references to objects of class ARC (the set of the output arcs). Class TRANSITION is the super-class of the subclasses T-COMP, T-COPY and T-RULE. The last one regroups all transitions of rule type; it includes attributes rule-name, events, cond, act, priority and methods insert-rule, delete-rule, activate-rule, modify-priority, evaluate-condition. rule-name represents the rule name. events represents the rule event; it refers to an object of class PLACE. *cond* represents the rule condition; it refers to an object of class CONDITION. act represents the rule action; it refers to an object of class PLACE. priority is a number reflecting the importance of the rule.

Also we define a class ARC with three subclasses: ARC-IN (input arcs), ARC-OUT (output arcs) and ARC-INHIB (inhibitor arcs). Class ARC has as attributes indice-p and indice-t, the value of which are the appropriate place and transition indexes (in relation to the attribute index of classes PLACE and TRANSITION).

The attribute *time* in the class TOKEN (the class which regroups all the tokens of an ECPN) indicates the time when we put the token on a place. The Fig. 4 presents the class hierarchy of our metabase with more details.

CONCLUSIONS

The approach presented in this study, is based on the Petri net model to analyse the termination of a set of rules. In this PN named Extended Coloured Petri Net (ECPN), the different components of the rules are presented such as their events, conditions, actions and priorities. ECPN can model both primitive and composite events. Furthermore, not only composite events can affect the active rules termination, but also the rules priority. Our approach is better than those presented in related work section because the ECPN is a good model for modelling, analysing and simulation of active database systems. It does not perform a simple analysis of cyclic paths but analyzes each element of the graph to determine if the rule triggering in a cyclic path finishes or not. This approach is general and can be applied not only in the database area but also in others applications which need event detection.

REFERENCES

Aiken, A., J.M. Hellerstein and J. Widom, 1995. Static analysis techniques for predicting the behaviour of active database rules. In ACM Trans. Database Sys., 20: 3-41.

Atkinson, M., F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier and S. Zdonik, 1989. The object-oriented database system manifesto. In Proc. of the 1st Inlt. Conf. on deductive and object-Oriented databases, Kyoto, Japan. Elsevier Science Publishers B.V (NorthHolland), pp: 223-240.

Bailey, J., L. Crnogorac, K. Ramamohanarao and H. Sndergaard, 1997. Abstract interpretation of active rules and its use in termination analysis. In Proceedings of the 6th International Conference on Database Theory, Jenuary 1997, Delphi, Greece, pp: 188-202.

Baralis, E., S. Ceri and S. Paraboschi, 1996. Modularization Techniques for Active Rules Design. ACM Trans Database Sys. (TODS), 21: 1-29.

Baralis, E., S. Ceri and S. Paraboschi, 1998. Compile-time and runtime analysis of active behaviors. Commun. IEEE TCDE., 10: 353-370.

Baralis, E. and J. Widom, 2000. Better static rule analysis for active database systems. ACM Trans. Database Sys. (TODS), 1 25: 269-332.

Chakravarty, S. and D. Mishra, 1994. Snoop: An expressive event specification language for active databases. In Data and Knowledge Engineering, 14: 1-26.

David, R. and H. Alla, 1989. Du grafcet aux réseaux de petri. Edition Hermès, Paris.

Gatziu, S. and K. R. Dittrich, 1994. Detecting composite events in active database systems using petri nets. In Proceedings of the 4th International Workshop on Research Issues in Data Engineering: Active Database Systems. Houston, Texas, February, pp: 1-8.

Gehani, N., H.V. Jagadish and O. Shumeli, 1992. Composite event specification in active databases: Model and Implementation. In Proc. 18th International Conference on Very Large Data Bases, Vancouver, Canada, August, pp: 1-12.

Geppert, A., S. Gatziu, K.R. Dittrich, H. Fritschi and A. Vaduva, 1995. Architecture and implementation of the active object oriented database management system SAMOS. Technical Report 95.29, CS Department, University of Zurich, pp: 1-39.

Jensens, K., 1992. Coloured Petri Nets: Basic concepts, analysis method and practical use, Vol. 1, EATCS.

Karadimce, A. and S. Urban, 1994. Conditional term rewriting as a formal basis for analysis of active database rules. In 4th Workshop on Research. Issues in Data Engineering (RIDE-ADS94), Texas, February, pp: 156-162.

Kokkinaki, A.I., 1998. On using multiple abstraction models to analyse active databases behaviour. In: Biennial World conference on Integrated Design and Process Technology, Berlin, Germany, June, pp: 1-8.

Lee, S.Y. and T.W. Ling, 1998. A path removing technique for detecting trigger termination. In Proceedings of 6th EDBT, Valencia, pp: 341-355.

Li, X. and J.M. Marín, 2004. Termination analysis in active databases: A petri net approach. International Symposium on Robotics and Automation (ISRA2004), Querétaro, Mexico, July, pp: 677-684.

Paton, N.W. and O. Diaz, 1999. Active database systems. In ACM Computing Surveys, 1: 63-103.

Widom J. and S. Ceri, (Eds.), 1996. Active Database Systems: Triggers and Rules for Advanced Database Processing, Data Management Systems, Morgan Kaufmann Publishers, San Francisco, California.

Zimmer, D., R. Unland and A. Meckenstock, 1996. Rule termination analysis based on petri nets. Cadlab Research Report 3, University of Paderborn, Germany, February, pp: 1-17.