Int. J. Advanced Networking and Applications
Volume: 03, Issue: 02, Pages:1098-1104 (2011)

A New Algorithm to Detect the Non-Termination of Triggers in Active Databases

Dr. R.Manicka chezian
Department of Computer Science, N G M College (Autonomous), Pollachi, Coimbatore - 642001, India
Email: chezian_r@yahoo.co.in

Dr.T.Devi
Department of Computer Science & Engineering., Bharathiar University, Coimbatore – 641 046, India
Email: tdevi5@gmail.com

----------ABSTRACT----------
Active Databases are a combination of traditional static databases and active rules, meant to be automated mechanisms to maintain integrity and facilitate in providing database functionalities. Active database systems can react to the occurrence of some predefined events automatically. In many applications, active rules or triggers may interact in complex and sometimes unpredictable ways, thus possibly yielding infinite rule executions by triggering each other indefinitely causing non-termination. The termination of active rules is an unpredictable problem, except when rule languages with very limited number of rules are used. This paper presents new algorithms for detecting termination / non-termination of rule execution using triggering graph and complex triggering graph, and these algorithms do not pose any limitation on the number of rules.

Key words: - Active Rules, Active Databases, Non-Termination, Termination, Triggers

Date of Submission: February 24, 2011 Date Revised: July 19, 2011 Date of Acceptance: August 2, 2011

1. Introduction

The processing of active rules (in commercial databases such as Oracle and Sybase called as triggers) is characterized by two important properties: termination and confluence. Confluence property of rules decides whether the execution order of non-prioritized rules make any difference in the final database state. Confluence for active database rules is a particularly difficult problem because, in addition to the standard problems associated with confluence, the interaction between rule triggering and rule priorities should also be counted [14],[15]. Rule activations in active databases can "cascade", i.e. the execution of an active rule can cause a change in the database state that causes another rule to be executed; the resulting change can then cause the activation of a third rule and so on. Ensuring that such cascaded rule activations do not go on forever therefore becomes of fundamental importance. Analyses that examine a set of active rules to determine whether rule activations will terminate are called termination analysis [17],[10],[4],[6]. The processing of a set of active rules terminates if, given any initial active database state, the execution of the rules does not continue indefinitely. Researchers in the past have used three ways to analyze this non-termination problem.. First, using static analysis, by giving *a priority*, the non termination is impossible for a particular rule set [16]. This task is made difficult, due to the complex interactions which can occur among rules. The second approach, is to impose some fixed limit upon the number of rules or triggers which can be executed in a triggering sequence - such a method is adopted by commercial database systems such as Oracle and Sybase. While easy to implement, it has the defect that valid rule execution sequences may exceed this limit and be prematurely halted and aborted, an approach unsuitable for applications where correctness and performance is paramount, such as mission critical systems and even banking systems. A third approach involves the imposition of syntactic restrictions on the rule set to ensure that rule execution always terminates. The difficulties of defining such criteria are recognized by the current SQL3 standard for triggers which does not attempt to prescribe methods for ensuring termination.

2. Related Work

The introduction of the active rules into database management systems produced new problems. Among these problems, non-termination is the one of the main problems. A rule set is guaranteed to terminate if, for any database state and initial modification, rule processing cannot continue forever. Thus, it is necessary to take measures to prevent against an infinite execution of the system. Several researches have started to try to give a solution to this problem. Aiken et al., (1995) are the first to introduce the notion of Triggering Graph (TG). They showed that a triggering graph without cycles determines and guarantees the termination of a set of active rules in an active database system [1]. 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 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 [13]. This condition severely limits the applicability of the technique. Baralis and Widom (2000) try to improve the previous 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 called Activation Graph (AG) [4]. Belbachir H and Ougouti N.S. (2006) present a new static approach for termination analysis of the active rules. It consists of the detection of cycles in a graph says dependences graph. This is built by taking into account the triggering of rules by other rules, influence of rule's actions on triggered rule's conditions and satisfiability of rule's conditions [5]. The principal work on dynamic analysis is done by E.Baralis et al., (1998). They performed the checking of active rules at run time to see whether a repeating database state has occurred in a history of previous states [7]. James Bailey et al.,(2000) described the new approach based on a dynamic upper limit to the number of rule firings. This limit reflects knowledge about past rule behavior on the database and provides a more accurate measure for when the data base management system (DBMS) should terminate rule execution [11]. Baba-hamed .L and Belbachir .H (2005) propose a method of termination analysis of active rules based on Petri Nets (PN) called as Extended Coloured Petri Net (ECPN) and give an object oriented representation to implement it [3]. Latifa Baba-Hamed (2008) has done a comparative study of the above method with the most known methods available in the literature for detecting non-termination. He claimed that their approach is better than the previous methods because ECPN is a good model for modeling, analyzing and simulation of active database systems and it does not perform a simple analysis of cyclic paths but analyzed each element of the graph to determine if the rule triggering in a cyclic path finishes or not [12].

3. Triggering Graph

Many of the works to date on termination analysis for active databases, triggering graph is used to check the set of rules is acyclic. One of the first works in this field is that of Aiken et al. (1995) who are the first to introduce the concept of triggering graph [1]. According to them, if the triggering graph is acyclic, the termination of the system of rules is guaranteed. Otherwise, the termination of the rules is not guaranteed. Non-termination occurs frequently when a set of rules or triggers available in a database system. Normally, rules are explained in the form of Event-Condition-Action (ECA) rules. For example, consider the following two rules.

Rule R1

Event:	X1 → decrease_overdraft(Y1)
Condition:	5000 < X1.capacity
Action:	X1 → decrease_capacity(500)

Rule R2

Event:	X2 → decrease_capacity(Y2)
Condition:	200 < X2.overdraft
Action:	X2 → decrease_overdraft(40)

The above two rules can be illustrated as in the Fig. 1 as a triggering graph. This shows there is a possibility of non-termination.

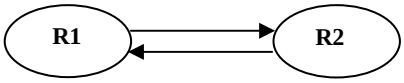

Figure 1 Triggering Graph

4. New Algorithm for Checking Non- Termination

Termination of rules in active databases is an important research issue for which a number of papers have been published [1],[2],[3],[5],[9],[10],[11],[17],[13],[8]. The processing of a set of active rules terminates if, given any initial active database state, the execution of the rules does not continue indefinitely. Termination of triggers in an active database is an undecidable problem. In most of the previous works, for finding the non termination, only triggering graph is considered [1],[2],[18],[13],[5]. In a triggering graph, whenever a cyclic graph occurs, it shows that the possibility of non-termination. Active rules may interact in complex and sometimes unpredictable ways, thus possibly yielding infinite rule executions by triggering each other indefinitely. In all the previous works, the researchers have analyzed the problem by taking a simple graph or by considering the single cyclic graph [1],[2],[18],[5], In the proposed approach, a complex graph having many cyclic graphs is considered. Consider six rules R1, R2, R3, R4, R5 and R6. A graph is built by means of a syntactic analysis of rules. The nodes of the graph are rules. Two rules R1 and R2 are connected by a directed edge from R1 towards R2 if the action of R1 contains a triggering event of R2. The presence of cycles in such a graph means a risk of non-termination of the set of rules. The absence of cycles in the triggering graph guarantees the termination of the set of rules. This is shown in the following triggering graph as Fig. 2.

Volume: 03, Issue: 02, Pages:1098-1104 (2011)

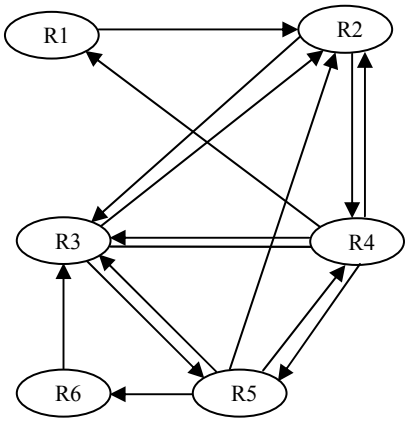

Figure 2 Triggering Graph having Six Rules

This implies that these rules may yield termination. The above two situations i.e., triggering graph with and without cycles are explained in the following new approach as an algorithm. The algorithm implemented for Fig. 2 for checking non-termination when two nodes are involved in forming a cyclic graph in Java's JDK1.3 compiler is shown in Appendix-1.

Step 1: Check whether all the rules belong to the same domain
Step 2: Draw a triggering graph showing the Activities of rules in a rule based system.
Step 3: Find out all the in-links and out-links for all the rules (or) nodes.
Step 4: If there is a same rule come as an in-link and out-links for a node, there is a possibility of non-termination (or) a cyclic graph and then go to step 5. Otherwise, the rules are terminated and then go to step 7.
Step 5: If there is a non-termination, assign a highest priority number for a rule that has to be fired first. If the rule execution generates events triggering higher priority rules, the rule should be suspended and resumed only when there will be no more higher priority triggered rules. Then assign next highest priority number to the next possible rule to get fired.
Step 6: After giving the priority numbers, if there is a cyclic graph, go to step 4. Otherwise go to next step.
Step 7: Exit rule processing and resume the transaction.

Algorithm1: Algorithm for checking non-termination when two nodes involved in forming a cyclic graph

5. Triggering Graph having More Than Two Rules forming a Cyclic Graph

Consider the triggering rules R1, R2, R3, R4, R5, R6 and R7. The interaction among the rules is shown in Fig. 3. When any two nodes are taken, they are not forming a cyclic graph. By using the algorithm 1, if the checking of termination is done, it gives there is a termination. But by seeing the triggering graph shown in Fig. 3, it is forming a cyclic graph. So, the algorithm 1 is not valid if more than two rules are forming a cyclic graph. The algorithm has to be changed in the case of more than two rules give a non-termination.

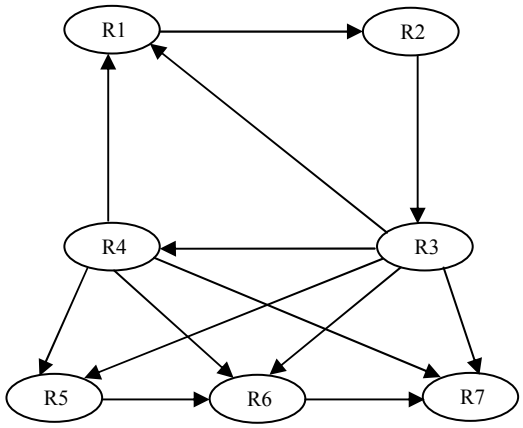

Figure 3 Triggering Graph having cyclic graphs

The cyclic graphs that are considered in the Fig. 2 are all involved with only two nodes. The algorithm 1 is using the in-links and out-links of a node. When in-link and out-link of a node is same, then there is a possibility of a non-termination. So, the algorithm1 may be suitable for a triggering graph forming a cyclic graph using two nodes. When a triggering graph is having a cyclic graph which is attained by three nodes or above, the algorithm 1 is not suitable since it will not reveal the possibility of a cyclic graph. So, the above proposed algorithm has to be modified to find a cyclic graph having three or more nodes. This approach is explained in algorithm 2.

Step 1: Check whether all the rules R_1, R_2, R_3… .. R_n Є D where D is a domain

Step 2: Draw a triggering graph showing the Activities of rules R_1, R_2, R_3 .. R_n.

Step 3: Find out all the in-links and out-links of R_1,R_2, R_3 … R_n

Step 4: Let in-link (R_i) denote the in-links of rule R_i and out-link(R_i) denote the out- links of rule R_i.

For rule R_i, let S_j denote the successor nodes, j = 1,2,3, … m.
begin
For Rule R_i, i=1, 2, 3,… n
For every S_j, j=1, 2, 3, … m
Form an ordered pair
(in-link (S_j), out-link (S_j));
Let this be (In_j, Out_j);
End;
End;
Termination = true;
For j = 1,2,3, ….m
For k = 1,2,3, … m
If In_j = out_k then termination= false;
End;
End;
If termination = true then
print "Termination occurs"
else print "Termination is not guaranteed";
End;

Step 5: Assign a highest priority number for a rule that has to be fired first. Then assign next highest priority number to the next possible rule to get fired and so on.

Step 6: After giving the priority numbers, if there is a cyclic graph, go to step 4. Otherwise go to next step.

Step 7: Exit rule processing and resume the transaction.

Algorithm 2: Algorithm for checking non-termination when more than two nodes are involved in forming a cyclic graph

The algorithm 2 is very much useful when a triggering graph is forming a cyclic graph in the case of more than two nodes are forming cyclic graphs. This can work for any number of nodes forming a cyclic graph in a triggering graph. The algorithm implemented for Fig. 3 in Java's JDK1.3 compiler is shown in Appendix-2.

6. Conclusions

Almost all of the work to date on termination analysis for active databases uses a simple triggering graph having two or three nodes / rules. In this work, a complex graph is considered to analyze termination of a set of active rules. Two different algorithms are given in this paper for checking non-termination. The first algorithm is considered for the triggering graph when two nodes are forming a cyclic graph. The second algorithm deals with the triggering graph when more than two nodes are involved in forming a cyclic graph. This work is implemented using Java's JDK1.3 compiler. So, these new algorithms are used to determine the termination/non-termination of rules for **any number of nodes in a complex triggering graph**. This work can be easily extended to check the non-termination of active rules or triggers using Petri nets approach.

References

[1] Aiken. A, Joseph M. Hellerstein and Jennifer Widon "Static Analysis Techniques for predicting the Behavior of Active Database Rules", *ACM Transactions on Database Systems, Vol. 20,* No.1, 1995, 3-41.

[2] Alain Couchot "Termination Analysis of Active Rules with Priorities", *Lecture Notes in Computer Science, Database and Expert Systems Applications, Springer Berlin / Heidelberg, Vol. 2736/2003,* 846 – 855.

[3] Baba-Hameed. L and Belbachir.H "Priority of Active Rules and Termination Analysis" *Journal of Theoretical and Applied Information Technology*, 2005, 11 – 19.

[4] Baralis E., and Widom J., "Better Static Rules Analysis for Active Database Systems", *ACM Transactions on Database Systems (TODS) Vol. 25,* No.3, 2000, 269-332.

[5] Belbachir. H and Ougouti. N.S., "A New Approach for Termination Analysis of Active Rules" *Journal of Applied Sciences Vol. 6(3),* 2006, 657 – 661.

[6] Danilo Montesi, Elisa Bertino, Maria Bagnato and Peter Dearnley "Rules Termination Analysis Investigating the Interaction between Transactions and Triggers", *Proceedings of the International Database Engineering and Applications Symposium (IDEAS'02),* 2002, 285 – 294.

[7] Elena Baralis, Stefano Ceri and Stefano Paraboschi "Compile-Time and Runtime Analysis of Active

Behaviors" *IEEE Transactions on Knowledge and Data Engineering, Vol. 10,* No.3, 1998, 353 – 370.

[8] Eun-Hye CHOI, Tatsuhiro TSUCHIYA, Tohru KIKUNO "Model Checking Active Database Rules*", Programming Science Technical Report, AIST CVS, Osaka University, Japan,* 2006, 1 – 16.

[9] Indrakshi Ray and Indrajit Ray "Detecting Termination of Active Database Rules Using Symbolic Model Checking", *Proceedings of the 5th Ease European Conference on Advances in Databases and Information Systems,* 2001, 266-279.

[10] James Alexander Bailey "*On the Foundations of Termination Analysis of Active Database Rules*", Ph.D., Thesis, University of Melbourne, Australia, 1997.

[11] James Bailey, Alexandra Poulovassilis, Peter Newson "A Dynamic Approach to Termination Analysis for Active Database Rules", *Proceedings of 1st International Conference on Computational Logic (CL2000),* LNCS 1861, 2000, 1106 – 1120.

[12] Latifa Baba-Hameed, "Termination Analysis Approach: A Comparative Study" *3rd International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA 2008)* Damascus, Syria, Vol.7, Issue 11, 2008, 1-6.

[13] Lee S.Y. and Ling T.W. "A Path Removing Technique for detecting Trigger Termination" *Proceedings of 6th EDBT*, Valencia, 1998, 341 – 355.

[14] Manicka chezian. R and Devi. T, "Confluence Property Determination using Associative Law", *International Journal of Engineering Research and Industrial Applications, Vol. 2,* No. II, 2009, 349-358.

[15] Manicka chezian. R and Devi. T, "Confluence in Active Data Base System", *Proceedings of the International Conference on Digital Factory – ICDF 2008,* Coimbatore Institute of Technology, Coimbatore, India, 2008, 2010-2015 **(Best Paper Award).**

[16] Sara Comai and Letizia Tanca "Termination and Confluence by Rule Prioritization" *IEEE Transactions on Knowledge and Data Engineering, Vol. 15*, No.2, 2003, 257 – 270.

[17] Saumya Debray and Timothy Hicky "Constraint-Based Termination Analysis for Cyclic Active Database Rules", *Lecture Notes in Computer Science, Computational Logic – CL 2000, Springer Berlin / Heidelberg, Vol.1861/2000,* 1121 – 1136.

[18] Susan D.Urban, Michael K.Tschudi, Suzanne W.Dietrich and Anton P.Karadmice "Active Rule Termination Analysis: An implementation and Evaluation of the Refined Triggering Graph Method"

[19] *Journal of Intelligent Information Systems, Vol. 12,* No. 1, 1999, 27 -60.

Authors Biography

Dr. R.Manicka chezian received his M.Sc., degree in Applied Science from P.S.G College of Technology, Coimbatore, India in 1987. He completed his M.S. degree in Software Systems from Birla Institute of Technology and Science, Pilani, Rajasthan, India and Ph D degree in Computer Science from School of Computer Science and Engineering, Bharathiar University, Coimbatore, India. He served as a Faculty of Maths and Computer Applications at P.S.G College of Technology, Coimbatore from 1987 to 1989. Presently, he has been working as an Associate Professor in N G M College (Autonomous), Pollachi under Bharathiar University, Coimbatore, India since 1989. His research focuses on Network Databases, Data Mining, Distributed Computing, Mobile Computing, Real Time Systems and Bio-Informatics.

Dr. T.Devi received the Master of Computer Applications from P.S.G. College of Technology, Coimbatore in 1987 and PhD from the University of Warwick, United Kingdom in 1998. She is presently Heading Department of Computer Applications, School of Computer Science and Engineering, Bharathiar University, Coimbatore. She has also served as an Associate Professor in Indian Institute of Foreign Trade, New Delhi from 2004 to 2008. Her current research centered on the Software Engineering, Product Introduction, Technical Process Management, Concurrent Engineering, Distributed Computing and Data Mining. She has published more than twelve research papers in International / National Journals. She has contributed more than 75 papers in various National / International / Conference / Seminars / Symposia.

Volume: 03, Issue: 02, Pages:1098-1104 (2011)

Appendix-1

LOOP BETWEEN TWO NODES

```
Enter the no of nodes:
6
Enter the node: 1
R1
Enter the no of in-links for R1:
1
Enter the in-links for R1:
R4
Enter the no of out-links for R1:
1
Enter the out-links for R1:
R2
Starting time: 1.310162317453e12
End time =1.310162317453e12
Elapsed time =0.0
Enter the node: 2
R2
Enter the no of in-links for R2:
4
Enter the in-links for R2:
R1, R3, R4, R5
Enter the no of out-links for R2:
2
Enter the out–links for R2:
R3, R4
THERE IS A LOOP BETWEEN R3 AND R2
A NON TERMINATION OCCURS.
Starting time: 1.310162909625e12
End time =1.310162909625e12
Elapsed time =0.0
THERE IS A LOOP BETWEEN R4 AND R2
A NON TERMINATION OCCURS.
Starting time: 1.310162909625e12
End time =1.310162909625e12
Elapsed time =0.0
Enter the node: 3
R3
Enter the no of in-links for R3:
4
Enter the in-links for R3:
R2, R4, R5, R6
Enter the no of out-links for R3:
3
Enter the out-links for R3:
R2, R4, R5
THERE IS A LOOP BETWEEN R2 AND R3
A NON TERMINATION OCCURS.
Starting time: 1.310163118718e12
End time =1.310163118718e12
Elapsed time =0.0
THERE IS A LOOP BETWEEN R5 AND R3
A NON TERMINATION OCCURS.
Starting time: 1.310163118718e12
End time =1.310163118718e12
Elapsed time =0.0
Enter the node: 4
R4
Enter the no of in-links for R4:
3
Enter the in-links for R4:
R2, R3, R5
Enter the no of out-links for R4:
4
Enter the out-links for R4:
R1, R2, R3, R5
THERE IS A LOOP BETWEEN R3 AND R4
A NON TERMINATION OCCURS
Starting time: 1.310163563765e12
End time =1.310163563765e12
Elapsed time =0.0
THERE IS A LOOP BETWEEN R5 AND R4
A NON TERMINATION OCCURS.
Starting time: 1.310163563765e12
End time =1.310163563765e12
Elapsed time =0.0
Enter the node: 5
R5
Enter the no of in-links for R5:
2
Enter the in-links for R5:
R3, R4
Enter the no of out-links for R5:
4
Enter the out-links for R5:
R2, R3, R4, R6
THERE IS A LOOP BETWEEN R3 AND R5
A NON TERMINATION OCCURS
Starting time: 1.310163563765e12
End time =1.310163563765e12
Elapsed time =0.0
THERE IS A LOOP BETWEEN R4 AND R5
A NON TERMINATION OCCURS.
Starting time: 1.310163563765e12
End time =1.310163563765e12
Elapsed time =0.0
Enter the node: 6
R6
Enter the no of in-links for R6:
1
Enter the in-links for R6:
R5
Enter the no of out-links for R6:
1
Enter the out-links for R6:
R3
THERE IS A LOOP BETWEEN R3 AND R6
A NON TERMINATION OCCURS.
Starting time: 1.310163563765e12
End time =1.310163563765e12
Elapsed time =0.0
```

Volume: 03, Issue: 02, Pages:1098-1104 (2011)

Appendix-2

MULTIPLE LOOP DETECTION

```
Enter the no of nodes:
7
Enter the nodes one by one:
R1, R2, R3, R4, R5, R6, R7
Enter the no.of.in links for   R1:
2
Enter the in-links for  R1:
R3, R4
Enter the no. of out - links for R1:
1
Enter the out –links for R1:
R2
 ***************************
Enter the no.of.in links for  R2:
1
Enter the in-links for R2:
R1
Enter the no of out - links for  R2:
1
Enter the out - links  for R2:
R3
***************************
Enter the no.of.in links for  R3:
1
Enter the in-links for R3:
R2
Enter the no of out -links for R3:
5
Enter the out - links for  R3:
R1, R4, R5, R6, R7
***************************
Enter the no.of.in links for  R4:
1
Enter the IN-Links for  R4:
R3
Enter the no of out links for  R4:
4
Enter the out - links for R4:
R1, R5, R6, R7
***************************
Enter the no.of.in - links for  R5:
2
Enter the in-links for  R5:
R3, R4
Enter the no. of out - links for  R5:
1
Enter the out - links  for  R5:
R6
***************************
Enter the no.of.in links for  R6:
3
Enter the in-links for  R6:

R3, R4, R5
Enter the no of out links for  R6:
1
Enter the out - links  for  R6:
R7
***************************
Enter the no.of.in links for  R7:
3
Enter the in-links for  R7:
R3, R4, R6
Enter the no. of. out Links for  R7:
0
***************************
Start Time =1.310250671156E12

THERE IS A LOOP AMONG NODES
R1      R2      R3
A  NON TERMINATION OCCURS.

End time =1.310250671171E12
Elapsed Time =15.0
THERE IS A LOOP AMONG NODES
R1      R2      R3      R4
A  NON TERMINATION OCCURS.

End Time =1.310250671171E12
Elapsed Time =15.0
```