The Priority of Rules and the Termination Analysis Using Petri Nets

Latifa Baba-hamed[1] and Hafida Belbachir[2]
[1]Computer Science Department, University of Oran Es-Sénia, Algeria
[2]Computer Science Department, University of USTO, Algeria

Abstract: *An active database system is a conventional database system extended with a facility for managing active rules (or triggers). Rules in active database systems can be very difficult to program, due to the unstructured and unpredictable nature of rule processing. In this paper, we propose a method of termination analysis of rules in an active database system based on Petri nets. We consider here the model structure and the model execution of the approach.*

Keywords: *Active database, ECA rules, termination rules, Petri nets, path, priority.*

Received December 4, 2005; accepted June 192006

1. Introduction

An active database system is a traditional database system and in addition, is capable to react, automatically, to state changes without the user's intervention. The active rules are, in general, of the form Event Condition Action (ECA). The action describes treatments to achieve when a specific event happens and some condition holds.

The system of active rule is the most important layer in an active database system. It is composed of the three main models that are: The model of rule specification, the model of representation and the model of execution. The model of rule specification permits the specification of the three components (event, condition and action) of the rule and its attributes. The model of active rule representation permits to specify the way in which rules are integrated in the (relational or object oriented) database taking into account the features of the data model, the associated language of data manipulation, and the model of transaction used. The model of execution permits to describe the manner in which the set of triggered rules is treated to be executed. This model establishes a link between the set of rules defined by the programmer and the set of transactions of a database application. It must take into account a certain number of factors such as: The modes of coupling, the net effect, the mode of events consumption, the instant of events consumption, and the multiple rules [8].

The active rule behaviour in such a system is, however, 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. The termination of rules is the property which determines if for all user-generated operations and initial database states, rule processing always reaches a point at which there are no triggered rule to consider. Many works concerning the active rule analysis developed techniques that permit, to the programmer of rules, to predict in advance (to the compilation) the termination of the execution of a set of rules [1-6, 9, 11-14, 16]. These methods are called static analysis methods. Other methods called dynamic analysis methods are pared, they permit to study the active rule termination at the runtime [4].

We are interested to the rule termination analysis approach given in [14] which we improve by considering the impact of the rule priority on the termination analysis problem. We have chosen this method because it 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.

The remainder of this paper is organised as follows. Section 2 presents some methods of analysis of the active rule termination, while section 3 exposes the chosen method briefly and shows how the priority of rules can affect their termination. Section 4 gives the implementation of the termination analyzer. Finally, section 5 concludes the paper.

2. Termination Methods

Among methods 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. Aiken *et al.* [1] are the first to introduce the notion of Triggering Graph (TG). In TG, nodes represent rules. Two rules r1 and r2 are joined by a directed edge r1 toward r2 if the action of r1 contains an event which

triggers r2. The authors showed that a triggering graph without cycle determines and guarantees the termination of rules. However, the authors don't take into account the rule condition or the effect of the execution of the rule action on the condition of another rule. This approach has been proposed in the specific case of the relational system.

In [5], an approach for a static analysis of the termination of condition action rule has been presented. This method 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 an Activation Graph (AG). In AG, nodes represent rules, and directed edges indicate that one rule may activate the other. If there is no cycle in the graph AG then rule processing is guaranteed to terminate. The propagation algorithm is used to decide when an edge (r_i, r_j) belongs to AG. To determine if ri may activate rj, the authors apply the propagation algorithm to rj's condition C and ri's action A. If the propagation algorithm yields insert or update operations, then the execution of ri may result in new data satisfying rj's condition. Thus, ri may activate rj, and the edge (ri, rj) belongs to the graph. If only delete operations or no operations are produced by the propagation algorithm, then the execution of ri cannot result in new data for rj's condition, and the edge is not included in the graph.

The work in [6] refined and improved the previous method and considered both condition action and ECA rules, making their approach widely applicable to relational active database rule languages. The inconvenience of this method is that its application is complex and reserved to the relational databases.

The approach in [4] proposed a technique that exploits the information of the two graphs TG and AG to analyse the termination of a set of ECA rules. This analysis uses an algorithm called algorithm of reduction. The inputs of the algorithm are a set of rules R without priority, and the two correspondent graphs TG and AG. If R is reduced to the empty set at the end of the algorithm execution, then the termination is guaranteed. The termination is not guaranteed otherwise. This approach presented an inconvenience because it didn't propose method of building the AG graph which is not obvious. Also, it doesn't detect all cases of termination.

Lee and Ling [13] proposed 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. Since this condition severely limits the applicability of the technique, the selection of predicates that can be updated only a finite number of times by trigger processing, is proposed.

Baralis *et al.* [3] 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. Its inconvenience is that it requires a complementary method to analyse rules within a module.

Bailey *et al.* [2] used 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 to active rule analysis is taken in [11], 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. An instance database is a collection of objects and events. A stable instance of the database is an instance where there is no event in waiting of treatment. An intermediate instance is an instance where there is at least one event in waiting of treatment. The termination means that a stable instance is reached. 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 [12] used Parameterised Petri Nets (PPNs) to analyse the active rule termination in the relational model. A PPN is a Petri Net (PN) whose places are parameterised. Parameters associated to places correspond to events, to names of relations, to names of attributes of these relations and expressions over the names of attributes. Each transition in the PPN corresponds to an active rule. The firing of the transition corresponds to the rule execution. If there are no cycles in the PPN model of an active Data Base System (aDBS) then the rules 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 [16]. To represent the triggering and activation notions in the PN, the authors gave for each rule two subnets Ei (for the triggering) and Ci (for the activation). The authors detected a non-terminating behaviour of rules using a Coverability Graph (CG) based on the Reachability Graph (RG) which contains all the markings (as nodes) which may reached from the start marking by firing all the possible transition sequences. To use the CG for the termination analysis, the authors extended the PN by introducing for every rule (ri) a counting place (P (i, cnt)), which counts the number of executions of ri. The

rule set exhibits a non-terminating behaviour if there is a rule ri for which the CG of the PN contains a node which contains an ∞ in the component (p (i, cnt)). 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 [14] presented an approach based on coloured Petri nets named Conditional Coloured Petri Net (CCPN) 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 did not consider the priority of rules.

3. Proposed Method

Our approach is inspired of [14] based on coloured Petri nets to describe and analyse ECA rules. 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 [12, 14, 16]. In SAMOS [10], PN is partially used for composite event detection and termination analysis.

3.1. Rule Modelling

In our model, named Extended Colored Petri Net (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 as shown in Figure 1.

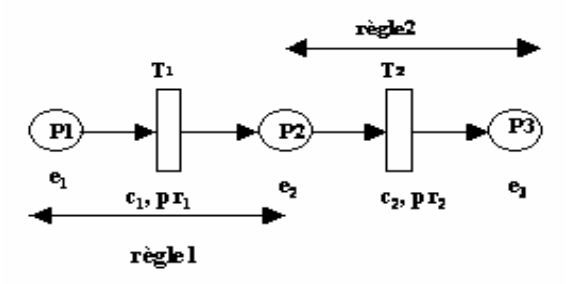

Figure 1. Relationships between two rules r1 and r2.

3.2. Formal Definition of an ECPN

An extended coloured Petri net ECPN is defined as follows:

$$ECPN = \{E, P, T, A, N, C, Con, Action, D, \tau, I\}$$

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. The notion of composite event is aborted in [7, 8, 10, 15] with more details. 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 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 the 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:

  $\forall t \in T_{rule}$: Type (con (t)) = B, where Con function evaluates the rule condition.

  $\forall t \in T_{comp}$: 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).

3.3. ECPN Execution

In ECPN, a token is a triple (*p, c, time*) where $p \in P$, $c \in C$ (p), and *time* specifies the natural time when the token is deposited into the place *p*. The set of all token is denoted by TS. A marking is a multi-set over TS. The initial marking M_0 is the marking which is obtained by evaluating the initialization expressions:

$$\forall (p, c, 0) \in TS: M_0 (p, c, 0) = (I (p))(c, \tau)$$

The set of all markings is denoted by M. We note by $\bullet t$ the set of the input arcs of the transition t, and by $t \bullet$ the set of the output arcs of the transition t. Transition $t \in T$ is enabled at a marking M if and only if:

$$\forall p \in \bullet t: | M (p) | = 0, type (t) = Negation$$
$$\forall p \in \bullet t: | M (p) | \geq 1, else$$

When a transition t is enabled, an enabled function is needed to specify what tokens transition t is enabled about. In ECPN, composite transitions and rule transitions fire conditionally. A composite transition fires once it is enabled (by $C_{composite}$ function) and the temporal condition is satisfied. A rule transition fires once it is enabled (by $C_{enabled}$ function) and the rule condition is satisfied.

A transition $t \in T$ fires if and only if:

$$\forall t \in T_{rule}, t \text{ is enabled and Type (Con(t)) = true}$$
$$\forall t \in T_{copy}, t \text{ is enabled}$$
$$\forall t \in T_{comp}, t \text{ is enabled, and } \forall p \in \bullet t, D (t) = [d_1 (t), d_2 (t)]: [d_1 (t) \leq \tau (M (p)) \leq d_2 (t)]$$

When a transition t is enabled in a marking M_1, and it fires, marking M_1 changes to marking M_2, defined by:

If $t \in T_{rule}$, $\forall p \in P$ *then*
$M_2 (p) = M_1 (p) - C_{enabled} (p, t) + Action (t, p)$
If $t \in T_{copy}$, $\forall p_1, p_2 \in P$, $p_1 \in \bullet t$, $p_2 \in t \bullet$ then
$M_2 (p_1) = M_1 (p_1) - C_{enabled} (p_1, t)$
$M_2 (p_2) = M_1 (p_2) + C_{enabled} (p_1, t)$
If $t \in T_{comp}$ then
If Type (t) = Negation, $\bullet t = \{p_1\}$, $t \bullet = \{p_2\}$ *then*
$M_2 (p_1) = M_1 (p_1)$
$M_2 (p_2) = M_1 (p_2) + C_{composite} (t, p_2)$
Else $\forall p^1_i \in \bullet t$, $t \bullet = \{p_2\}$
$M_2 (p^1_i) = M_1 (p^1_i) - C_{enabled} (p^1_i, t)$
$M_2 (p_2) = M_1 (p_2) + C_{composite} (t, p_2)$

When a transition $t \in T_{rule} \cup T_{comp}$ is enabled at a marking M_1, but not fired because Type (Con (t)) = false, marking change still exists, new marking M_2 is defined as following:

$$\forall p \in P: M_2 (p) = M_1 (p) - C_{enabled} (p, t)$$

3.4. Example of an ECPN

Let's consider the set of rules R1, R2, R3, and R4 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 rules set given above is shown in Figure 2. Furthermore, we consider that $R1 >_p R3 >_p R2 >_p R4$ where $Ri >_p Rj$ means that Ri has priority than Rj.

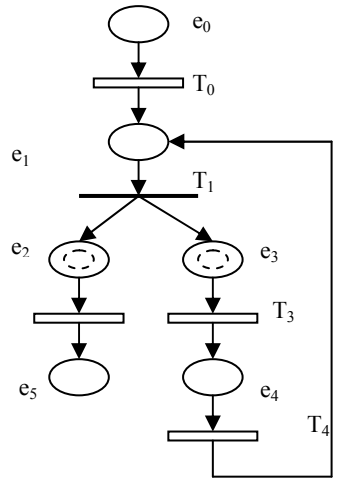

Figure 2. The ECPN corresponding to the set of rules given above.

3.5. Termination Analysis

Let E = {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 of PN theory, the notions of paths, cyclic paths and acyclic paths.

3.5.1. Incidence Matrix

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. A place $p \in P_{prim}$ is an initial node of the ECPN if its column in incidence matrix has zeros values and only one negative value. A terminal node is a place $p \in P_{prim}$ which represents an action rule. The corresponding column to the final node, in incidence matrix, has only values that belong to $\{0\} \cup N+$, where N is the set of natural numbers.

As ECA rule components are represented in the ECPN model, relationships between rules can be found easily. An 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.

In order to form the paths of the ECPN, first, columns of incidence matrix that represent initial nodes should be located, i. e., p1, and by definition of initial node there is only a negative value in the column, then the position index of the negative value is the index of the transition t, i. e., t1.

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 there exists 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.

We give in the following paragraph, the theorems (corresponding to the composite events and the conditions of rules) which affect the termination analysis.

	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

Figure 3. Incidence matrix obtained from ECPN of Figure 2.

3.5.2. Theorems

The composite events that affect the termination analysis are: *Negation, conjunction, any, sequence*, and *simultaneous*. The theorems 1, 2 and 3 correspond to them. The theorem 4 concerns the condition of a rule belonging to a cyclic path.

Theorem 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.

Theorem 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.

Theorem 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.

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

3.5.3. 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: Explore the paths in the ECPN one by one according to the priority of rules and using incidence matrix:

```
termine ← false
Repeat for each path R
    If R is acyclic
       Then termine ← true
    Else /*R is cyclic*/
      If R satisfies the theorems conditions
        Then termine ← true
    End
Until the end of the paths exploration or termine =
  true
If termine = true
  Then the algorithm is stopped and returns
      "The termination is guaranteed"
Else the algorithm returns
      "The termination is not guaranteed"
End
```

3.5.4. Illustrative Example

To show the impact of the rules priority on the termination analysis in an ECPN, we consider the set of rules given in the example of section 3.4 and follow the algorithm steps:

Step 1: Convert the ECA rules set (of section 3.4) into the ECPN given in Figure 2, where:

e0: Reduce-salary ()
e1: Raise-salary ()
e2: A copy of e1 (e1 triggers two rules R2 and R3)
e3: A copy of e1
e4: Send-bonus ()
e5: Be-retired ()
T0: Corresponds to the rule R1
T1: Is a transition of copy type. It produces two same events (e2 and e3)
T2: Corresponds to the rule R3
T3: Corresponds to the rule R2
T4: Corresponds to the rule R4

Step 2: Create the incidence matrix corresponding to this ECPN; it is given in Figure 3.

Step 3: Explore the paths in the ECPN one by one according to the priority of rules and using incidence matrix. So we begin with the path constituted by: (T0, e0), (T0, e1), (T1, e1), (T1, e2), (T2, e2) and (T2, e5) because R3 (represented by the transition T2) has priority than R2 (represented by the transition T3) as it is mentioned in section 3.4. Then we conclude that the set of rules {R1, R2, R3, R4} terminates because the explored path is acyclic. It should be noted that the second path in this ECPN constituted by: (T0, e0), (T0, e1), (T1, e1), (T1, e3), (T3, e3), (T3, e4), (T4, e4), (T4, e1), and (T1, e1) is cyclic.

4. 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. The termination analyzer includes three modules. The first one is used to generate the incidence matrix corresponding to a given ECPN, the second searches the paths in the matrix, and the third one analyzes and treats these paths according to the algorithm given in section 3.5.3.

5. Conclusion

The approach presented in this paper, is based on the Petri nets 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 termination methods section because the ECPN is a good model for modelling, analysing and simulation of Active Database systems (ADB). It is independent of the model used in the ADB (relational or object). 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

[1] Aiken A., Hellerstein J. M., and Widom J., "Static Analysis Techniques for Predicting the Behaviour of Active Database Rules," *ACM Transactions on Database Systems*, vol. 20, no. 1, pp. 3-41, 1995.

[2] Bailey J., Crnogorac L., Ramamohanarao K., and Sondergaard H., "Abstract Interpretation of Active Rules and its Use in Termination Analysis," *in Proceedings of 6th ICDT, LNCS 1186*, Greece, pp. 188-202, 1997.

[3] Baralis E., Ceri S., and Paraboschi S., "Modularisation Techniques for Active Rules Design," *ACM Transactions on Database Systems (TODS)*, vol. 21, no. 1, pp. 1-29, 1996.

[4] Baralis E., Ceri S., and Paraboschi S., "Compile-Time and Runtime Analysis of Active Behaviours," *IEEE TCDE*, vol. 10, no. 3, pp. 353-370, 1998.

[5] Baralis E. and Widom J., "An Algebraic Approach to Rule Analysis in Expert Database Systems," *in Proceedings of the 20th Conference on VLDB,* Santiago, Chile, pp. 475-486, 1994.

[6] Baralis E. and Widom J., "Better Static Rule Analysis for Active Database Systems," *ACM TODS*, vol. 25, no. 3, pp. 269-332, 2000.

[7] Cakravarty S. and Mishra D., "Snoop: An Expressive Event Specification Language for Active Databases," *Data & Knowledge Engineering,* vol. 14, no. 10, pp. 1-26, 1994.

[8] Collet C., "Active Databases: From Relational Systems to Object-Oriented Systems," *Research Report*, 965-I-LSR-4, IMAG Laboratory, Grenoble, France, October 1996.

[9] Debray S. and Hickey T., "Constraint-Based Termination Analysis for Cyclic Active Database Rule," *in Proceedings of DOOD Conference*, London, pp. 1-15, 2000.

[10] Gatziu S. and Dittrich K. R., "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 System (RIDE-ADS'94*), Houston, Texas, pp. 1-8, Febuary 1994.

[11] Karadimce A. and Urban S., "Conditional Term Rewriting as a Formal Basis for Analysis of Active Database Rules," *in Proceedings of the 4th Workshop on Research, Issues in Data Engineering: Active Database System (RIDE-ADS'94*), Texas, pp. 156-162, February 1994.

[12] Kokkinaki A. I., "On Using Multiple Abstraction Models to Analyse Active Databases Behaviour," *in Proceedings of Biennial World Conference on Integrated Design and Process Technology*, Berlin, Germany, pp. 1-8, June 1998.

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

[14] Li X. and Medina Marín J., "Termination Analysis in Active Databases: A Petri Nets Approach," *in Proceedings of the International Symposium on Robotics and Automation (ISRA'2004)*, Querétaro, Mexico, pp. 677-684, July, 2004.

[15] Li X. and Medina Marín J., "Composite Event Specification in Active Database Systems: A Petri Nets Approach," *in Proceedings of the IEEE International Conference on System, Man, and Cybernetics*, The Hague, Netherlands, pp. 97-116, October 2004.

[16] Zimmer D., Unland R., and Meckenstock A., "Rule Termination Analysis Based on Petri Nets," *Technical Report*, University of Paderborn, Germany, pp. 1-17, 1996.

Latifa Baba-hamed received her Master degree in computer science from the University of Oran, Algeria, in 1994. She is a member of Database Systems Group at USTO. Her research interests include object-oriented, temporal, active databases, and integration of data.

Hafida Belbachir received her PhD in computer science from the University of Oran, Algeria, in 1990. Currently, she is a professor in the Department of Computer Science at the University of USTO, where she heads the Database Systems Group. Her research interests include object-oriented, active, and multimedia databases, and data mining.