Termination Analysis of Active Rules Modular Sets

Alain Couchot

Université Paris Val de Marne, Créteil, France

Postal address :
Alain Couchot
16, rue Fagon
F-75013 Paris
France

email :
a.couchot@libertysurf.fr

Abstract. This paper presents an algorithm for static analysis of termination of active rules sets in the context of a modular conception. The termination of the active rules is an undecidable problem. Several recent works suggested proving termination by using the concept of triggering graph. We propose here an original approach, based on these works, and that allows to guarantee the termination of a set of rules, conceived by several designers, even when none of the designers knows the set of the active rules. In this optics, we clarify the notions of private event and of public event, and we refine the notion of triggering graph (by enclosing events in graphs). We introduce the notion of stabilization field, which allows to represent by an uniform frame all the situations of deactivation of a rule condition listed by the previous termination algorithms.

Keywords. Active databases. Termination. Modular conception. Rules. Static analysis.

1 Introduction

We are here interested by the termination problem of active rules for databases in a modular context. The active rules are structured according to paradigm Event-Condition-Action [10, 12]. The event is an immediate fact, which can arise inside or outside of the system. The condition is a predicate or a request on the database. The action is generally composed of a sequence of updates of the database, or of a procedure containing updates of the database. Coupling modes allow to specify the moment of evaluation of the part Condition, or the moment of execution of the part Action. The most frequent modes are immediate or deferred modes.

In the section 2, we present the previous works on the subject ; in the section 3, we define the notions of public events and private events ; in the section 4, we introduce the events/rules graphs ; in the section 5, we define the notion of stabilization field ; in the section 6, we present our algorithm for termination static analysis ; the section 7 concludes.

2 Previous works

The termination of the active rules is an undecidable problem, except when languages of rules with very limited possibilities are used [2]. The previous works on the static analysis of the active rules propose criteria supplying sufficient conditions allowing to guarantee termination.

Some methods develop translation tools of the active rules into rewriting terms systems [15], into Condition-Action rules [7], or into deductive rules [11 , 14]. However, the application of these methods is often complex, and reserved for relational databases.

The majority of works on the termination of the active rules exploit the concept of *triggering graph*.

[8] introduced, for the first time, the notion of *triggering graph* ; this notion is clarified by [1] : A such 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 an triggering event of *r2*. The presence of cycles in a such 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. However, the possible deactivation of the condition of the rule is not taken into account by this analysis. A finer analysis is led by [3 , 5, 6, 13, 16, 17], taking the possible deactivation of the condition of a rule into account. [18] proposes a technique to remove a path instead of removing a node.

The behavioral stratification [4] takes place in a context of modular conception of the active rules. The principle of the behavioral stratification [4] is to separate rules into strata,

based on the features of rules, so that actions carried out by the rules of every stratum converge on a given result. For every stratum *S*, convergence is described by a metrics *ms*. Intuitively, *ms* measures the distance which separates the current state from a stable state (that is a state where no rule is triggered), produced by the execution of the rules of one stratum working in insulation (that is without execution of the rules of the other strata). Strata are ordered by priorities. The execution of rules belonging to the strata of weaker priority must not disrupt the metrics of rules belonging to the strata of stronger priority.
The algorithm proposed by [4] is the only algorithm which takes into account a modular conception of the active rules.

Our work is based on the following observation: when several designers define subsets of a global set of active rules, one of the designers must have a global knowledge of all the defined rules, so that one of the termination algorithms can be applied. Even in the case of the algorithm proposed by [4], which leans on a modular conception of the active rules, one of the designers must have a knowledge of the set of priorities established among all strata, and must know metrics associated with every stratum. We propose a termination algorithm, which does not require to establish priorities between rules and strata, and that does not require the definition of a metrics for every stratum.

Let us intuitively expose the solution that we propose. First of all, we distinguish two types of events : *private events* and *public events*. A designer can define rules triggered by events corresponding to public operations (*public events*), and events corresponding to operations which are reserved for him (*private events*). A rule can provoke public events and private events. Every designer can build a graph, containing the rules defined by the designer, the public events and the private events. If every designer succeeds in eliminating, in his own graph, cycles, as well as the paths which connect two public events, the termination of the set of rules is guaranteed.

3 Private events and public events

We introduce in this paragraph the notions of *private event* and of *public event*. We first distinguish between *database events* and *external events* : the *database events* are provoked by a database operation and the *external events* are not provoked by a database operation.

Let us consider a database *DB*. Let us suppose that *n* designer *D1*, *D2*, …, *Dn* work on this database. Each designer is charged to conceive a part of *DB*.

An database event *E* is *private for the designer Di* iff :
• *Di* is the only designer authorized to use the database operation provoking the event *E*.

If the database event *E* is not a private event for any designer, then *E* is a *public event*.

4 Events/rules graphs

Let us consider a designer *Dk*. A *events/rules graph for the designer Dk* is a directed graph, where nodes are :

- the rules defined by *Dk,*
- the public events,
- the events private for the designer *Dk,*
- the external events.

There is an directed edge from the event *E* towards the rule *R* if event *E* can trigger the rule *R*. There is a directed edge from the rule *R* to the event *E* if the rule *R* can provoke the event *E*.

Example. *Designer_a* and *Designer_b* define a set of active rules. *Ra1*, *Ra2* are the two rules defined by *Designer_a* ; *Rb1*, *Rb2* are the two rules defined by *Designer*_b. *E1* is a public event ; *Ea1* is an event private for *Designer_a* ; *Eb1*, *Eb2* are events private for *Designer_b*.

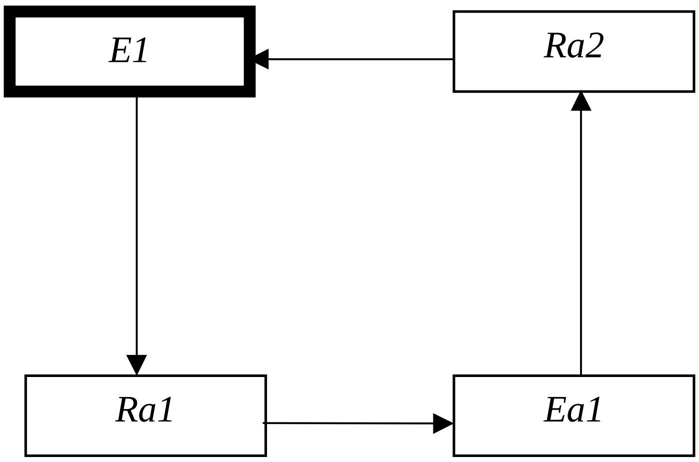

Figure 1. *Events/rules graph for Designer_a.*

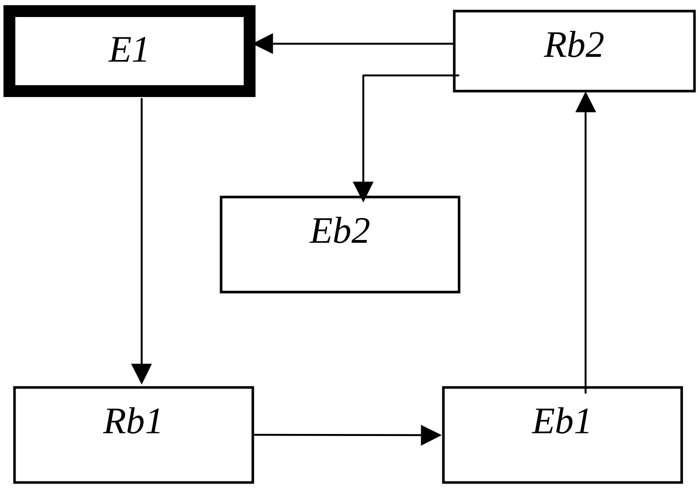

Figure 2. *Events/rules graph for Designer_b.*

Private path of an events/rules graph. Let *Path* be a path of an events/rules graph. We say that *Path* is a private path iff *Path* contains no public event.

5 Stabilization fields

We introduce here the notion of *stabilization field*. The utility of this notion is to propose an uniform presentation of the different causes of deactivation of a condition. A stabilization field contains a private path (the *stabilizer* of the field) and a set of events (the *destabilizing set* of the field). Informally, the conjunction of the conditions of the rules of the stabilizer can only be evaluated to **TRUE** a finite number of times, if there is only a finite number of occurrences of the events of the destabilizing set.

Definition. Let *Path* be a private path of the events/rules graph for the designer *Dk* : *G*(*Dk*). Let *E1* , *E2*, …, *Es* be *s* events of $G(D1) \cup \ldots \cup G(Dn)$.

We say that the pair (*Path*, {*E1*, *E2*, ... , *Es* }) is a *stabilization field of the graph G*(*Dk*) iff the following property holds for each rules process *P* :
(a finite number of occurrences of the events *E1* , *E2*, …, …, *Es* appear during *P*) $\Rightarrow$ (*P* contains a finite number of occurrences of *Path*)

Path is a *stabilizer* ; {*E1*, *E2*, ... *Es* } is a *destabilizing set* associated with the stabilizer *Path*.

The interest of the stabilization fields is to define a common frame of presentation for all the deactivation cases listed by the previous algorithms [3 , 5 , 6, 13 , 16 , 17].

For example, a private path is a stabilizer, if there is a generalized connection formula along this path which can not be satisfied [16 , 18] ; an associated destabilizing set contains all the database operations susceptible to modify the attributes contained in the atoms of the connection formula.

Thanks to this notion, it is possible to represent in an uniform way all the cases of deactivation of a condition listed by the previous termination algorithms :

- The deactivation of the condition of the rule *Rule* because of the incompatibility of the condition of *Rule* with the condition of a rule *Rule1* [16] (the path containing *Rule* and *Rule1* is then a stabilizer) ;
- The deactivation of the condition of *Rule* because of the action of a rule *Rule1* preceding the triggering of *Rule* [17] (the path containing *Rule* and *Rule1* is then a stabilizer) ;
- The deactivation of the condition of *Rule* because of limited monotonous operations executed by the action of *Rule1* [13] (the path containing *Rule* and *Rule1* is then a stabilizer) ;
- The deactivation of the condition of *Rule* when *Rule* is a self-deactivating rule [3 , 5 , 6] (the path (*Rule*) is then a stabilizer).

6 Termination algorithm

The termination algorithm consists of two main parts.

The first part removes from the graph *G*(*Dk*) :
• the rules which are just triggered a finite number of times during any rules process, and
• the private events which can just be provoked a finite number of times during any rules process.

The second part "removes" from the graph *G*(*Dk*) the private path *Path* by means of the "Path Removing Technique" [18], if :
• (*Path*, {*E1*, *E2*, …, *Es*}) is a stabilization field of *G*(*Dk*),
• all the events *E1*, *E2*, … *Es* are private for *Dk*, and
• no event *E1*, *E2*, … *Es* is in the reduced graph *G*(*Dk*).

```
G(Dk) = initial events/rules graph for the designer Dk
Remove the external events from G(Dk)
WHILE nodes of G(Dk) are removed
    WHILE nodes of G(Dk) are removed
        Part (a) : "Deletion of nodes"
        Remove from G(Dk) the nodes without incoming edge, except the public events
        Remove from G(Dk) the edges without origin node
    ENDWHILE

    FOR each private path Path of G(Dk)
        Part (b) : "Deletion of paths"
        IF ((Path, {E1, E2, … , Es}) is a stabilization field of G(Dk)) AND (E1, E2, … Es are private for Dk) AND
        (no event E1, E2, … , Es is in G(Dk))
                "Remove" Path from G(Dk) by means of the "Path Removing Technique" [18]
        ENDIF
    ENDFOR
ENDWHILE
```

Termination Algorithm.

We can now expose our termination theorem :

Termination theorem.
If, for every designer *Dk*, the following property holds :
• the reduced graph *G*(*Dk*), obtained by means of the termination algorithm, is without cycle and without path connecting two public events,
Then : Termination of the set of the rules defined by *D1* , *D2*, … , *Dn* is guaranteed.
(Prove is omitted due to space limitations).

Example. Let us consider the example of section 4. Let us call *Path1* the private path (*Ra1*, *Ea1*, *Ra2*) and *Path2* the private path (*Rb1*, *Eb1*, *Rb2*).Let us suppose that (*Path1*, ∅) is a stabilization field of G(*Designer_a*) and that (*Path2*, ∅) is a stabilization field of

G(*Designer_b*). Then, termination of the set of rules {*Ra1*, *Ra2*, *Rb1*, *Rb2*} is guaranteed by *Designer_a* and *Designer_b,* even if *Designer_a* does not know the rules defined by *Designer_b*, and even if *Designer_b* does not know the rules defined by *Designer_a*. Note that the behavioral stratification [4] can not draw the same conclusion (indeed, no priority between rules has been defined ; furthermore, the local convergence of each stratum, required for application of [4], is not necessarily guaranteed).

7 Conclusion

We have presented a significant improvement of the termination analysis of the active rules in a modular context. We have first introduced the notions of public event and of private event. We then proposed a refinement of the triggering graphs : the events/rules graphs. We have then defined the notion of stabilization field. We finally exposed a termination algorithm, which can prove the termination of a set of rules defined by several designers, even when no designer knows rules defined by the other designers.

Compared with the behavioral stratification [4], which is, in our knowledge, the only work comparable to our approach, our termination algorithm offers the following advantages:

- Local convergence, necessary to prove termination during the use of [4], is a very restrictive condition and is difficult to prove : a "good metrics" must be chosen by every designer, to prove local convergence. Our termination algorithm does not rest on the definition of a metrics, but uses the notion of stabilization field. This notion exploits the previous termination algorithms. So, the determination of the stabilization fields can automatically be computed.
- A total order must be defined on strata for the application of [4]. Our algorithm does not require the definition of priorities between rules.
- The use of [4] supposes that a designer has a knowledge of all the metrics associated with strata, and of the priorities between strata. As regards our termination algorithm, every designer is supposed to have no knowledge of the active rules defined by the other designers.

In the future, we plan to take the semantics of the composite events [9] into account, in order to refine our termination analysis, by enclosing a graphical representation of the composite events in the events/rules graphs. We also plan to investigate new stabilization fields.

References

1. A. Aiken, J. Widom, J.M. Hellerstein. Behavior of Database Production Rules : Termination, Confluence and Observable Determinism. In *Proc. Int'l Conf. on Management of Data* (*SIGMOD*), San Diego, California, 1992.
2. J. Bailey, G. Dong, K. Ramamohanarao. Decidability and Undecidability Results for the Termination Problem of Active Database Rules. In *Proc. ACM Symposium on Principles of Database Systems* (*PODS*), Seattle, Washington, 1998.
3. E. Baralis, S. Ceri, S. Paraboschi. Improved Rule Analysis by Means of Triggering and Activation Graphs. In *Proc. Int'l Workshop Rules in Database Systems* (*RIDS*), Athens, Greece, 1995.
4. E. Baralis, S. Ceri, S. Paraboschi. Modularization Techniques for Active Rules Design. In *ACM Transactions on Database Systems,* (*TODS*), 21(1), 1996.
5. E. Baralis, S. Ceri, S. Paraboschi. Compile-Time and Run-Time Analysis of Active Behaviors. In *IEEE Transactions on Knowledge and Data Engineering*, 10 (3), 1998.
6. E. Baralis, J. Widom. An Algebraic Approach to Rule Analysis in Expert Database Systems. In *Proc. Int'l Conf. on Very Large Databases* (*VLDB*), Santiago, Chile, 1994.
7. E. Baralis, J. Widom. Better Static Rule Analysis for Active Database Systems. In *ACM Transactions on Database Systems* (*TODS*), to appear, 2000.
8. S. Ceri, J. Widom. Deriving Production Rules for Constraint Maintenance. In *Proc. Int'l Conf. on Very Large Databases* (*VLDB*), Brisbane, Queensland, Australia, 1990.
9. S. Chakravarthy, D. Mishra. Snoop : An Expressive Event Specification Language for Active Databases. In *Data and Knowledge Engineering*, 14 , 1994.
10. C. Collet, T. Coupaye, T. Svensen. NAOS : Efficient and Modular Reactive Capabilities in an Object Oriented Database System. In *Proc. Int'l Conf. on Very Large Databases* (*VLDB*), Santiago, Chile, 1994.
11. S. Comai, L. Tanca. Using the Properties of Datalog to prove Termination and Confluence in Active Databases. In *Proc. Int'l Workshop on Rules in Database Systems* (*RIDS*), Skoevde, Sweden, 1997.
12. U. Dayal, A.P. Buchmann, D.R. Mc Carthy. Rules are Objects too : a Knowledge Model for an Active Object Oriented Database System. In *Proc. Int'l Workshop on Object-Oriented Database Systems*, Bad Münster am Stein-Ebernburg, FRG, 1988.
13. S. Debray, T. Hickey. Constraint-Based Termination Analysis for Cyclic Active Database Rules. In *Proc. Int'l Conf. on Deductive Object Oriented Databases* (*DOOD*). London, United Kingdom, 2000.
14. S. Flesca, S. Greco. Declarative Semantics for Active Rules. In *Proc. Int'l Conf. on Database and Expert Systems Applications* (*DEXA*), Vienna, Austria, 1998.
15. A.P. Karadimce, S.D. Urban. Conditional Term Rewriting as a Formal Basis for Analysis of Active Database Rules. In *Proc. Int'l Workshop on Research Issues in Data Engineering* (*RIDE-ADS*), Houston, Texas, 1994.
16. A.P. Karamdice, S.D. Urban. Refined Triggering Graphs : a Logic-Based Approach to Termination Analysis in an Active Object-Oriented Database. In *Proc. Int'l Conf. on Data Engineering* (*ICDE*), New-Orleans, Louisiana, 1996.

17. S.Y. Lee, T.W. Ling. Refined Termination Decision in Active Databases. In *Proc. Int'l Conf. on Database and Expert Systems Applications* (*DEXA*), Toulouse, France, 1997
18. S.Y. Lee, T.W. Ling. A Path Removing Technique for Detecting Trigger Termination. In *Proc. Int'l Conf. on Exttermination Database Technology* (*EDBT*), Valencia, Spain, 1998.