Improving Termination Analysis of Active Rules with Priorities

Alain Couchot

Laboratoire Cedric-Isid,
Conservatoire National des Arts et Métiers, France
couchot-a@wanadoo.fr

Abstract. This paper presents an algorithm for termination static analysis of active rules with priorities. Active rules termination is an undecidable problem. Several recent works have suggested proving termination by using the concept of triggering graph. We propose here a refinement of these works, exploiting the priorities defined between rules.

1 Introduction

We are here interested in the active rules termination problem. The active rules are structured according to paradigm Event-Condition-Action. The active rules termination is an undecidable problem. The majority of works on active rules termination exploit the concept of *triggering graph* [4] ; the nodes of the graph are rules. Two rules *r1* and *r2* are connected by a oriented edge from *r1* to *r2* if the action of *r1* can provoke a triggering event of *r2*. The presence of cycles in a such graph means a risk of non-termination of the rules set. The absence of cycles in the triggering graph guarantees the termination of the rules set. Some works refine the triggering graph analysis, taking into account the influence of rules conditions [1, 3, 7, 8], the influence of composite events [5, 9], the modular rules design [2, 6]. Our work is based on the following observation : priorities between rules can be used to refine termination analysis. We show in this paper that much more termination cases can be detected using the priorities between rules.

2 Path Sets

We introduce here the notions of *path set of a rule* and *path set of a path*. The path set serves for replacing the notion of cycle, used in the previous termination algorithms. We suppose that each rule is defined with a numeric priority and that the rules process is the following:

1. Choose a rule instance with the strongest priority in the set of triggered rules instances.
2. Remove the chosen rule instance from the set of the triggered rule instances.

3. Evaluate the condition.
4. If the condition is false : go to step 1.
5. If the condition is true, execute the action.
6. Update the set of triggered rules instances.
7. Go to step 1.

2.1 Path Set of a Rule

We first recall the notion of triggering graph: the nodes of the triggering graph are the active rules; there is an oriented edge from a rule R_1 to a rule R_2 if the rule R_1 can trigger the rule R_2.

Let G be a triggering graph. We first precise the notion of *path*. Let N_1 , N_2... N_i... N_n be *n node*s (not necessarily all different) of G, such as there is a oriented edge since N_{i+1} towards N_i. The tuple (N_1 , N_2... N_n) constitutes a *path*. We adopt the following notation : $N_n \rightarrow N_{n-1} \rightarrow ... \rightarrow N_i \rightarrow ... \rightarrow N_1$. N_1 is called the *last node* of the path. N_n is called the *first node* of the path.

The *path set* of the rule *R Path_Set*(*R* ; *G*) is the set of the paths *Path* of *G* which satisfy the following properties: (i) the last rule of the path *Path* is *R*; (ii) the path *Path* does not contain twice the same node; (iii) the path *Path* is not included in a path which satisfies the properties (i) and (ii) (except itself).

The path set of *R* is built performing a "depth search" in the opposite direction of the edges.

2.2 Path Set of a Path

We generalize now the notion of path set, defining the *path set of a path*. Let us consider a path *Path* = $N_n \rightarrow N_{n-1} \rightarrow ... \rightarrow N_i \rightarrow ... N_2 \rightarrow N_1$. Let us suppose that *Path_Set*(N_n ; *G*) = {$Path_1$, $Path_2$,… , $Path_p$ }. The path set of the path *Path* is *Path_Set*(*Path* ; *G*) = {($Path_1 \rightarrow N_{n-1} \rightarrow ... \rightarrow N_1$), ($Path_2 \rightarrow N_{n-1} \rightarrow ... \rightarrow N_1$), …, ($Path_p \rightarrow N_{n-1} \rightarrow ... \rightarrow N_1$)}.

2.3 Priority of a Path / Priority of a Path Set

We introduce here the notion of *priority of a path*. The priority of a path is the weakest priority of the rules of the path. We use the notion of priority of a path to define now the *minimal priority* of a path set: this is the weakest priority of the paths of the path set. We adopt the following notation: *m_p*(*Path_Set*(*Entity* ; *G*)). (*Entity* is a path or a rule). We also define the *maximal priority* of a path set: this is the stronger priority of the paths of the paths set. We adopt the following notation: *M_p*(*Path_Set*(*Entity* ; *G*)).

3 Reduction of the Triggering Graph

3.1 Destabilizing Set of a Path

In order to refine the deactivation of a path, we introduce the notion of *destabilizing set*. The utility of this notion is to list the rules which can oppose the deactivation of the path.

Definition. Let *Path* be a path of the triggering graph G. Let R_1 , R_2, …, R_s be s rules of G. We say that the set $\{R_1, R_2, ... , R_s\}$ is a *destabilizing set* of *Path* iff the following property holds for each rules process P:
(A finite number of instances of the rules R_1 , R_2, …, …, R_s occur during P) $\Rightarrow$ (There is only a finite number of occurrences of *Path* during P).

For a destabilizing set, we will use the following notation: *Desta_Set*(*Path* ; G).

3.2 Reduction of the Triggering Graph

We can reduce the triggering graph thanks to the four following considerations:
(1) The rule R can be removed from the triggering graph if R has no incoming edge. Indeed, R will be just triggered a finite number of times.
(2) Let R be a rule. Let $Path_j$ be a path of *Path_Set*(R ; G). We can remove $Path_j$ from *Path_Set*(R ; G) if $\emptyset$ is a destabilizing set of $Path_j$. Indeed, in this case, it is impossible for the rules process to go through the path $Path_j$ an infinite number of times.
(3) The rule R can be removed from the triggering graph if *Path_Set*(R ; G) = $\emptyset$. Indeed, in this case, it is impossible for the rules process to reach the rule R an infinite number of times, since all the paths which lead to R are deactivated after a finite time.
(4) Let us consider now a destabilizing set $\{R_1, R_2, \dots , R_s\}$ of the path *Path*. If we observe one of the following properties : (i) the maximal priority of *Path_Set*(*Path* ; G) is strictly smaller than the minimal priority of *Path_Set*(R_i ; G) or (ii) the maximal priority of *Path_Set*(R_i; G) is strictly smaller than the minimal priority of *Path_Set*(*Path* ; G), then the rule R_i can be removed from the destabilizing set $\{R_1, R_2, \dots , R_s\}$.

3.3 Termination Algorithm

We can now sketch the termination algorithm. The termination algorithm captures the four properties which we have shown above :
(1) We can remove a rule from the current graph if the rule has no incoming edge.
(2) We can remove a path *Path* from a path set if $\emptyset$ is a destabilizing set of *Path*.

(3) We can remove a rule from the current graph if the path set of the rule is empty.
(4) We can remove a rule from a destabilizing set depending on the priorities.

If the final triggering graph is empty, termination is guaranteed. Else, the remaining rules can possibly be triggered an infinite number of times.

4 Conclusion

We have presented a significant improvement of the termination analysis of the active rules defined with priorities. We have developed the notions of path set of a rule, path set of a path, destabilizing set of a path. We can then reduce the destabilizing set of a path thanks to the priorities of the path sets. When the destabilizing set of a path is empty, the path can be removed from the path set of a rule. When the path set of a rule is empty, the rule can be removed from the triggering graph. So, the triggering graph can be reduced thanks to considerations about the priorities of the rules. In the future, we plan to conceive an algorithm which proposes priorities between rules, when the termination can not be guaranteed.

References

1. 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.
2. E. Baralis, S. Ceri, S. Paraboschi. Modularization Techniques for Active Rules Design. In *ACM Transactions on Database Systems, (TODS)*, 21(1), 1996.
3. 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.
4. S. Ceri, J. Widom. Deriving Production Rules for Constraint Maintenance. In *Proc. Int’l Conf. on Very Large Databases (VLDB)*, Brisbane, Queensland, Australia, 1990.
5. A. Couchot. Improving Termination Analysis of Active Rules with Composite Events. In *Proc. Int' l Conf. on Database and Expert Systems Applications (DEXA)*, Munich, Germany, 2001.
6. A. Couchot. Termination Analysis of Active Rules Modular Sets. In *Proc. Int' l Conf. on Information and Knowledge Management (CIKM)*, Atlanta, Georgia, USA, 2001.
7. A. Couchot. Improving the Refined Triggering Graph Method for Active Rules Termination Analysis. In *Proc. British National Conf. on Databases* (*BNCOD*), Sheffield, United Kingdom, 2002.
8. A.P. Karadimce, 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.
9. A. Vaduva, S. Gatziu, K.R. Dittrich. Investigating Termination in Active Database Systems with Expressive Rule Languages. In *Proc. Int’l Workshop on Rules in Database Systems*, Skoevde, Sweden, 1997.