Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing Letters 94 (2005) 71–77
Information Processing Letters
www.elsevier.com/locate/ipl

Event Trace Independence of active behavior ☆

Angela Bonifati [a,*], Stefano Ceri [b,1], Stefano Paraboschi [c]

[a] *Icar CNR, Via P. Bucci 41C, I-87036 Rende, Italy*
[b] *DEI, Politecnico di Milano, P.za L. da Vinci 32, I-20133 Milano, Italy*
[c] *DIGI, Università di Bergamo, Viale Marconi 5, I-24044 Dalmine, Italy*

Received 12 April 2004; received in revised form 10 December 2004

Available online 16 January 2005

Communicated by S.E. Hambrusch

Abstract

We present the *Event Trace Independence* (*ETI*), a novel property of active rules exhibiting a behavior independent of the specific event sequence that had caused a state transition. When employed in a distributed setting, this property supersedes the classical property of confluence, which is not sufficient herein. We show that ETI is in general undecidable and provide a sufficient condition, called *invertibility*, which offers a practical way to demonstrate ETI.

Keywords: Rule analysis; Active databases; Rule termination; Events; Databases

1. Introduction

Event Trace Independence (ETI) is a novel property that characterizes active databases. The property holds for a given rule system and rule sets when, for any arbitrary state change from an initial to a final state, the behavior of the system is independent of the particular state sequence that produced the final state after rules activation. This property is particularly relevant in the case of asynchronously distributed active systems whenever the changes on a site are gathered in a trace, which is forwarded to another site after the occurrence of the changes. Changes occurred on a site are collected in an *event trace* and exported to another site, on which they may fire a set of actions. The action is a generic *stored procedure*, i.e., code that is located on a site and executed as response to a special event. These systems include (but are not limited to) classical distributed databases, data warehouses, replicated systems, mediators, caching systems, recent P2P architectures and the Web. This problem is also relevant in the context of recent technology developed for XML. Sig-

☆ Research presented in this paper is supported by Esprit project WebSI.
* Corresponding author.
E-mail addresses: bonifati@icar.cnr.it (A. Bonifati), ceri@elet.polimi.it (S. Ceri), parabosc@unibg.it (S. Paraboschi).
URL: http://www.icar.cnr.it/angela.
[1] Work is partially supported by CESTIA-CNR (Milano).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2004.12.014

nificant examples, such as Apache Jelly, Macromedia DreamWeaver, JSP.Net, basically embed method calls inside the documents and make these calls activated upon the click of a mouse or the occurrence of a particular event.

Event traces as described above can be captured at run-time, packed and then shipped to another machine. Or, alternatively, they can be generated by comparing two different versions of the same data [1]. Indeed, besides containing the sequence of occurred events, each event trace also describes the transition between two states of the system. There are usually different event traces describing the same state transition. In a distributed asynchronous context, such as that of the systems described above, event traces may be used liberally, e.g., after merging with other traces. Thus, one would like to guarantee that the active behavior triggered *elsewhere* by those event traces is independent of the trace, but only dependent on the occurred state transition. This is different from what was studied in centralized active databases [2–6], where the event trace has usually been fixed and known in advance. Two properties of *termination* and *confluence* have been defined for sets of database rules. Termination guarantees that any transaction execution associated with rule execution is completed, whereas confluence guarantees that such completion produces the same final state.

However, in a distributed context working asynchronously, such as that of the Web or in any other of the above settings, termination and confluence are no longer sufficient to explain the active behavior. When events may have occurred outside the scope of the system or may not have been traced with the same tool, the behavior of rules in the current site still needs to be guaranteed as safe.

In the remainder, we assume reactive behaviors produced by active rules and we adopt the relational model as the underlying data model. Nevertheless, the results can be applied to any different event-based reactive processing among those discussed.

Statement of the problem. The property of *event trace independence* complements the notion of confluence in all the situations when transaction execution is not known in terms of the sequence of the atomic events, but only in terms of the initial state and the final state, produced by the sequence. To better appreciate the difference between confluence and event trace independence, consider Fig. 1. The (bold) segments (labeled with *ET*) between the database state DB_0 and the state DB_i represent the event traces, while those (labeled with *RE*) between the state DB_i and each final database state DB_{fi} represent the rule executions triggered by the events of the traces. Note that each point in the figure represents a database state, that, in case of the event traces, is due to a user update and, in case of rule executions, is caused by a rule activation.

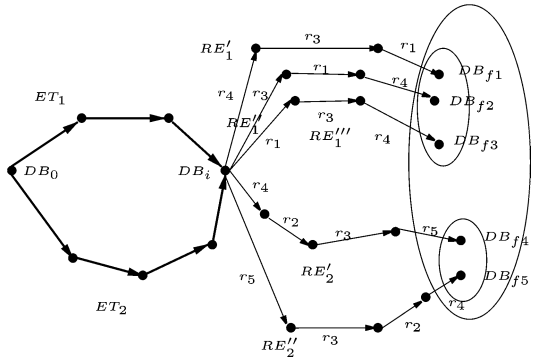

Fig. 1. An overview of confluent and ETI rule sets.

Observe that in classical confluence, for a given event trace, e.g., ET_1, what is required by the property is that the rule executions raised by this event trace ET_1 (in the figure, RE'_1, RE''_1 and RE'''_1, respectively) should converge to the same final state $DB_{f1} = DB_{f2} = DB_{f3}$ (enclosed in the small oval). Instead, in case of the ETI property, for each event trace between DB_0 and DB_i (in the figure only two of them are illustrated, ET_1 and ET_2), different rule executions can be raised (the previous rule executions for ET_1, and RE'_2, RE''_2 for ET_2). What is demanded by ETI is a stronger convergence, i.e., the convergence of all the final states into a unique one ($DB_{f1} = DB_{f2} = DB_{f3} \equiv DB_{f4} = DB_{f5}$, enclosed in the big oval).

Contributions. Event trace independence is a difficult problem, because in addition to confluence, we must guarantee the uniqueness of the final state while enumerating all the possible event traces occurring between two states. We show that ETI is undecidable. Therefore, we devise a sufficient condition, called *rule invertibility*, which guarantees ETI. This is similar to the use of *rule commutativity* to guarantee confluence [2]. However, commutativity holds for a fixed event trace and is no longer applicable here. More-