ON LOGICAL FOUNDATIONS OF
ACTIVE DATABASES
Georg Lausen,
Bertram Ludäscher,
and Wolfgang May

Abstract: In this chapter, we present work on logical foundations of active databases. After introducing the basic notions and terminology, we give a short overview of research on foundations of active rules. Subsequently, we present a specific state-oriented logical approach to active rules which aims at combining the declarative semantics of deductive rules with the possibility to define updates in the style of production rules. The resulting language Statelog models (flat) transactions as a sequence of intermediate transitions, where each transition is defined using deductive rules. Since Statelog programs correspond to a specific class of locally stratified logic programs, they have a unique intended model. Finally, after studying further fundamental properties like expressive power and termination behavior, a Statelog framework for active rules is presented. Although the framework is surprisingly simple, it allows to model many essential features of active rules, including immediate and deferred rule execution, and composite events. Different alternatives for enforcing termination are proposed leading to tractable subclasses of the language. Finally, we show that certain classes of Statelog programs correspond to Datalog programs with production rule semantics (i.e., with inflationary or noninflationary fixpoint semantics).
12.1 INTRODUCTION

Traditional database systems are passive, i.e., they only execute queries or transactions explicitly submitted by the user or an application program. In contrast, active databases (ADBs) allow to define (re)active behavior by means of active rules resulting in a quite flexible and powerful formalism. ADBs support automatic triggering of data manipulating operations or more general actions as a response to external or internal events, and are attractive for several reasons. A number of advanced database features like checking and enforcing integrity constraints, incremental view maintenance, (view) updates, access and version control, and many others, can be expressed naturally and uniformly by active rules. From a software engineering perspective, it is desirable to move functionality from application programs to the ADB. Thus, more code can be shared by different application programs, thereby facilitating software development and maintenance. Moreover, an ADB can monitor events more efficiently than a passive system which requires frequent polling by the application program. Finally, active rules can be used in advanced applications like workflow management and data warehousing applications, the latter requiring maintenance of huge materialized views.

Although the area of active databases has been quite active indeed, and produced a huge amount of literature as well as a number of prototypes, theoretical work on foundations is still rare and there is no generally accepted framework as is the case for deductive databases.

In Section 12.2 we present the basic notions of active rules, in Section 12.3 we briefly review the state of the art with a focus on theoretical aspects and foundations of active rules. Sections 12.4 and 12.5 elaborate on a logical framework based on a state-oriented extension of Datalog. In this framework, active rules are viewed from a more abstract and logical point of view, which allows to study fundamental properties of active rules independent from specific implementation aspects.

12.2 BASICS OF ACTIVE RULES

12.2.1 Terminology

Active rules are typically expressed as Event-Condition-Action (ECA) rules of the form

\[
\text{on } \langle \text{event} \rangle \text{ if } \langle \text{condition} \rangle \text{ then } \langle \text{action} \rangle .
\]

Whenever the specified \textit{event} occurs, the rule is triggered and the corresponding \textit{action} is executed if the \textit{condition} is satisfied in the current database.