The State of Change: A Survey

Anthony J. Bonner[1] and Michael Kifer[2]

[1] Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada, bonner@db.toronto.edu

[2] Department of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794, U.S.A., kifer@cs.sunysb.edu

Abstract. Updates are a crucial component of any database programming language. Even the simplest database transactions, such as withdrawal from a bank account, require updates. Unfortunately, updates are not accounted for by the classical Horn semantics of logic programs and deductive databases, which limits their usefulness in real-world applications. As a short-term practical solution, logic programming languages have resorted to handling updates using ad hoc operators without a logical semantics. A great many works have been dedicated to developing logical theories in which the state of the underlying database can evolve with time. Many of these theories were developed with specific applications in mind, such as reasoning about actions, database transactions, program verification, etc. As a result, the different approaches have different strengths and weaknesses. In this survey, we review a number of these works, discuss their application domains, and highlight their strong and weak points.

1 Introduction

Logic-based approaches to updates can be broadly classified into two categories: those designed for database or logic programming and those designed for reasoning about programs and actions. Prolog [CM81] is an early (and the best-known) example of the languages in the first category. The situation calculus [McC63], Dynamic Logic [Har79], and Temporal Logic [Pnu77] are, in our view, the forefathers of the modern approaches to reasoning about actions. As we shall see, both the situation calculus and Dynamic Logic influenced the design of several logic-based languages for programming database transactions. Temporal Logic had a much lesser influence on the languages for database transactions, and we shall not discuss it here.[1]

In the next section, we briefly review these two approaches, and in subsequent sections, we discuss other, more-recent proposals for dealing with state changes in logic-based systems.

[1] Although the concept of time plays an important role in databases, and especially in temporal databases, these areas have adapted little from the vast knowledge that exists on Temporal Logic.