Results on Reasoning about Updates in Transaction Logic

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. Transaction Logic was designed as a general logic of state change for deductive databases and logic programs. It has a model theory, a proof theory, and its Horn subset can be given a procedural interpretation. Previous work has demonstrated that the combination of declarative semantics and procedural interpretation turns the Horn subset of Transaction Logic into a powerful language for logic programming with updates [BK98,BK94,BK93,BK95]. In this paper, we focus not on the Horn subset, but on the full logic, and we explore its potential as a formalism for *reasoning* about logic programs with updates. We first develop a methodology for specifying properties of such programs, and then provide a sound inference system for reasoning about them, and conjecture a completeness result. Finally, we illustrate the power of the inference system through a series of examples of increasing difficulty.

1 Introduction

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 with ad hoc operators without a logical semantics. In previous work, we addressed this problem by developing a general logic of state change called *Transaction Logic* (abbreviated $\mathcal{TR}$). Like classical logic, $\mathcal{TR}$ has a Horn fragment that supports logic programming; but unlike classical logic programs, programs in $\mathcal{TR}$ can *update* the database as well as evaluate queries. Previous research on $\mathcal{TR}$ has focussed on the specification and execution of logic programs in this Horn fragment [BK98,BK94,BK93,BK95]. In contrast, this paper focuses not on the Horn fragment, but on the full logic. In particular, we show that the full logic can be used to express properties of $\mathcal{TR}$ logic programs and to reason about them.

Because it is a general logic of state change, $\mathcal{TR}$ can be applied in many areas, including databases, logic programming, workflow management, and artificial intelligence. These applications, are discussed in detail