Concurrency and Communication in Transaction Logic

Anthony J. Bonner*
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4, Canada
bonner@db.toronto.edu

Michael Kifer†
Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794-4400, U.S.A.
kifer@cs.sunysb.edu

Abstract

In previous work, we developed Transaction Logic (or $\mathcal{TR}$), which deals with state changes in deductive databases. $\mathcal{TR}$ provides a logical framework in which elementary database updates and queries can be combined into complex database transactions. $\mathcal{TR}$ accounts not only for the updates themselves, but also for important related problems, such as the order of update operations, non-determinism, and transaction failure and rollback. In the present paper, we propose *Concurrent Transaction Logic* (or $\mathcal{CTR}$), which extends Transaction Logic with connectives for modeling the concurrent execution of complex processes. Concurrent processes in $\mathcal{CTR}$ execute in an interleaved fashion and can communicate and synchronize themselves. Like classical logic, $\mathcal{CTR}$ has a "Horn" fragment that has both a procedural and a declarative semantics, in which users can program and execute database transactions. $\mathcal{CTR}$ is thus a deductive database language that integrates concurrency, communication, *and* updates. All this is accomplished in a completely logical framework, including a natural model theory and a proof theory. Moreover, this framework is flexible enough to accommodate many different semantics for updates and deductive databases. For instance, not only can updates insert and delete tuples, they can also insert and delete null values, rules, or arbitrary logical formulas. Likewise, not only can databases have a classical semantics, they can also have the well-founded semantics, the stable-model semantics, etc. Finally, the proof theory for $\mathcal{CTR}$ has an efficient SLD-style proof procedure. As in the sequential version of the logic, this proof procedure not only finds proofs, it also *executes* concurrent transactions, finds their execution schedules, and updates the database. A main result is that the proof theory is sound and complete for the model theory.

Appears in *Proceedings of the Joint International Conference and Symposium on Logic Programming (JICSLP)*, September 2–6 1996 Bonn, Germany, pages 142–156. MIT Press.

*Work supported in part by an Operating Grant from the Natural Sciences and Engineering Research Council of Canada.

†Supported in part by the NSF Grant IRI-9404629.