

Recovery for Extended Transaction Models

Shu-Wie F Chen

Department of Computer Science
Columbia University
New York NY 10027-7003 USA
swfc@cs.columbia.edu

Calton Pu

Department of Computer Science and Engineering
Oregon Graduate Institute
Portland OR 97291-1000 USA

March 24, 1997

1 Introduction

The transaction has emerged as an important paradigm for building reliable distributed information systems [5, 1]. Its success stems from its ability to hide the problems of concurrency and failure from both the application programmer and the end user. The atomic transaction model defines simple, flat transactions that guarantee the execution properties of atomicity, consistency, isolation, and durability. Transaction processing support for atomic transactions has been well-studied and has been implemented in all commercial database systems.

In this abstract, we summarize our work on recovery for extended transaction models [2]. Our goal is to build flexible and efficient recovery systems to support extended transactions by extending existing atomic TP systems. We have developed a design methodology that takes as input a formal description of an extended transaction model and produces as output a recovery system design that can be readily implemented through modular extensions to atomic recovery systems. Our goal and approach are illustrated in Figure 1.

In the next section, we present the MARS architecture which is the foundation of our work. We then present recovery algorithms which implement the MARS architecture and recovery micro-protocols which implement the recovery algorithms in sections 3 and 4, respectively. In section 5, we describe our design methodology and then summarize the application of our methodology to several transaction models in section 6.

2 The MARS Architecture

Our work is based on the observation that any recovery algorithm that implements transaction-oriented recovery must implement a database-to-transaction mapping that determines which transactions should be aborted and committed to achieve database recovery and a transaction-to-operation mapping that determines which operations must be recovered through UNDO, REDO, or DO to achieve transaction recovery. We have proposed a generic recovery algorithm which is based solely on the definition of transaction-oriented recovery and is thus independent of any transaction model. To support the general recovery algorithm, we have introduced the Modular Architecture for Recovery Systems (MARS) (Figure 2). MARS is a decomposition of the recovery process into a small set of modular composable components and is a generalization of existing atomic TP system architectures.

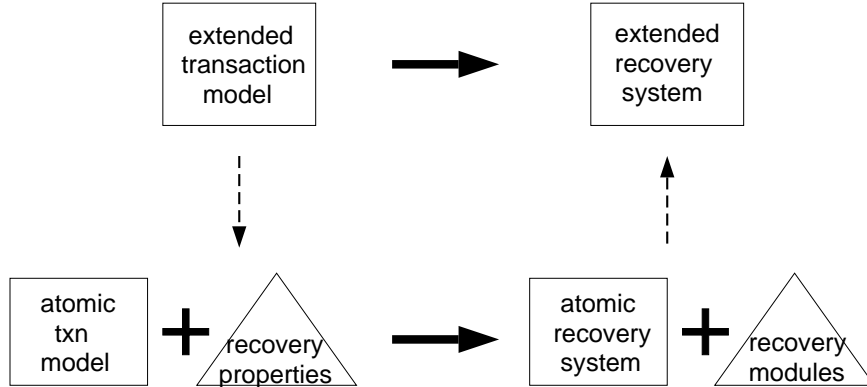


Figure 1: Recovery for Extended Transaction Models

Like atomic recovery systems, extended recovery consists of an analysis phase which generates a recovery plan and a restoration phase which executes the recovery plan. We summarize the extended analysis phase below.

The analysis phase for extended recovery is implemented by a transaction states analysis module and a transaction operations analysis module:

- The transaction states analysis module implements the database-to-transaction mapping of transaction-oriented recovery. Its output consists of transaction recovery specifications (TRS) of the form $(t_i, TS_{\text{crash}}, TS_{\text{recovery}})$ which specify that t_i should be recovered from the TS_{crash} state to the TS_{recovery} state. TRSs subsume traditional abort/commit sets.
- The transaction operations analysis module implements the transaction-to-operation mapping of transaction-oriented recovery. Its output consists of transaction operation specifications (TOS) of the form $(t_j, \text{opn}_i[\text{obj}])$ which specify that the permanence of the operation $\text{opn}_i[\text{obj}]$ should be determined by the outcome of the transaction t_j even though it was invoked by transaction t_i .

3 Recovery Algorithms

We have developed crash-aware algorithms for implementing the transaction states analysis, transaction operations analysis, and transaction recovery modules that constitute the MARS architecture. We summarize our results for the transaction states analysis module.

In designing the transaction states analysis algorithms, we considered those transaction models which allow for the establishment of constraints on transaction outcomes and thus restrict transactions from being unilaterally aborted or committed. We express such constraints using transaction dependencies as defined by ACTA [3]. Our algorithms consider the effects of transaction dependencies on recovery:

- During normal processing, a transaction scheduler ensures that transaction events (e.g. abort, commit) and transaction dependency establishments do not lead to non-recoverable executions. For instance, the scheduler prevents two contradictory dependencies that expect a

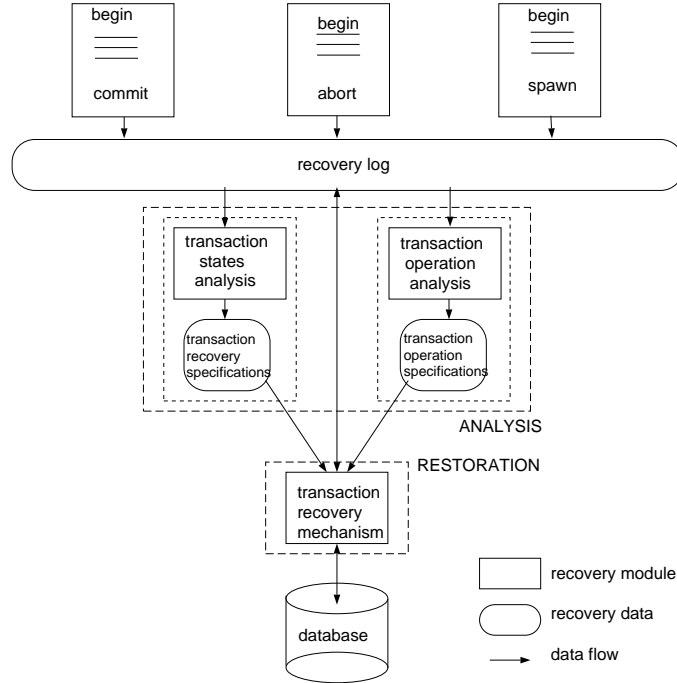


Figure 2: The MARS Recovery Architecture

transaction to both abort and commit from both being established. Such a scheduler can be implemented using a conflict table and a lock protocol in the same manner as that for database schedulers.

- The scheduler writes log records which explicitly specify the expected outcome of active transactions. These log records can be used to compute the TS_{recovery} fields of TRSs in a single log scan. Such an approach avoids the potentially non-linear performance of computing solutions to transaction dependencies during restart processing.

4 Recovery Microprotocols

Based on the descriptions of the previous section, it is not apparent how the MARS recovery algorithms can be implemented through modular composition or extensions to atomic recovery. To this end, we have applied the concept of microprotocols introduced in the *x*-kernel project [7] to define a set of recovery microprotocols that are decompositions of the recovery algorithms which can be combined to implement various recovery functionalities.

As an example, transaction states analysis functionality can be implemented through combinations of the following four simple microprotocols:

- *crash states analysis*- determine the states of all transactions at the time of the system crash by examining transaction state log records.
- *constrained outcome analysis*- determine those transactions which are required to abort or to commit by examining constrained outcome log records.

- *outcome set computation*- compute abort/commit sets using the output of crash states analysis by placing committed transactions in the commit set and active and aborted transactions in the abort set.
- *TRS computation*- compute TRSs using the outputs of crash states analysis and constrained outcome analysis.

The transaction states analysis module for atomic recovery can be implemented with the crash states analysis and outcome set computation microprotocols whereas a sagas recovery system will require the crash state analysis, constrained outcome analysis, and TRS computation microprotocols.

5 A Methodology for Designing Recovery Systems

We have developed a design methodology that takes as input a formal description of an extended transaction model and produces as output a recovery system design expressed as a set of recovery microprotocols to be composed following the MARS architecture. The methodology consists of four steps corresponding to the components in Figure 1:

1. provide a formal definition of the transaction model using ACTA.
2. analyze the ACTA definition to determine the recovery characteristics of the transaction model.
3. determine the required recovery microprotocols based on the recovery characteristics of the transaction model.
4. compose the recovery microprotocols following the MARS architecture to produce a recovery system supporting the transaction model.

6 Applying the Methodology

We have applied our methodology to construct recovery systems for a variety of transaction models. As base cases, our methodology trivially supports the atomic transaction model and produces recovery systems for the distributed and nested models consistent with commercial implementations. We have applied the methodology to design and analyze recovery systems for more exotic transaction models, including the split/join [9], cooperative group [6], and sagas [4] transaction models.

As an example, our methodology produces the recovery system design for the split/join model shown in Figure 3. The solid boxes denote microprotocols required for atomic recovery as well as split/join recovery. The dotted boxes denote microprotocols required solely for split/join. For normal processing, a split/join recovery system must include microprotocols for performing write-ahead logging (WAL), execution of transaction requests, and commit scheduling. For restart analysis, it requires the crash state determination, outcome set computation, and delegation analysis microprotocols. For restart recovery, it requires the recovery log filter, transaction UNDO, and transaction REDO microprotocols.

By comparing the split/join model design with the atomic model design, we can see that a split/join recovery system can be implemented by extending an atomic recovery system with the commit scheduling, delegation analysis, and recovery log filter microprotocols. Alternatively, we can

start with a general recovery system that consists of all possible recovery microprotocols and tailor a recovery system for the split/join model using the technique of specialization previously applied to operating systems [8] and programming languages [8]. Such an approach has the potential of producing modular implementations of extended recovery systems that do not sacrifice the linear-time performance characteristics of atomic recovery systems.

7 Conclusion

Building effective support for extended transactions has been identified as one of the key database research areas leading into the next century [10]. In this abstract, we have summarized results in architecture, algorithms, implementation, design methodology, and system designs for recovery for extended transaction models. These contributions support the modular construction of flexible, efficient extended recovery systems through systematic extensions to atomic recovery systems.

While maintaining backward-compatibility with existing recovery systems, our results are useful for constructing both centralized and distributed extended recovery systems. Our work is also applicable for introducing the transaction concept into non-traditional environments such as the Internet and World-Wide Web (WWW). Such environments will require light-weight, flexible TP support to guarantee reliability. We believe that the MARS architecture implemented via our recovery microprotocols satisfies this property and will be useful in implementing recovery for WWW applications.

References

- [1] Philip A. Bernstein and Eric Newcomer. *Principles of Transaction Processing*. Morgan Kaufmann, 1997.
- [2] Shu-Wie F Chen. *Recovery for Extended Transaction Models*. PhD thesis, Columbia University, 1997.
- [3] Panos K. Chrysanthis and Krithi Ramamritham. Synthesis of extended transaction models using ACTA. *ACM Transactions on Database Systems*, 19(3):450–491, September 1994.
- [4] Hector Garcia-Molina and Kenneth Salem. Sagas. In *ACM SIGMOD Proceedings*, pages 249–259, 1987.
- [5] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.
- [6] Bruce Martin and Claus H. Pederson. Long-lived concurrent activities. In Ozsu, Dayal, and Valduriez, editors, *Distributed Object Management*. Morgan Kaufmann, 1994.
- [7] Sean W. O'Malley and Larry L. Peterson. A dynamic network architecture. *ACM Transaction on Computer Systems*, 10(2):110–143, May 1992.
- [8] Calton Pu, Tito Autrey, and Andrew Black et al. Optimistic incremental specialization: Streamlining a commercial operating system. In *Proceedings of the Fifteenth Symposium on Operating Systems Principles*, 1995.

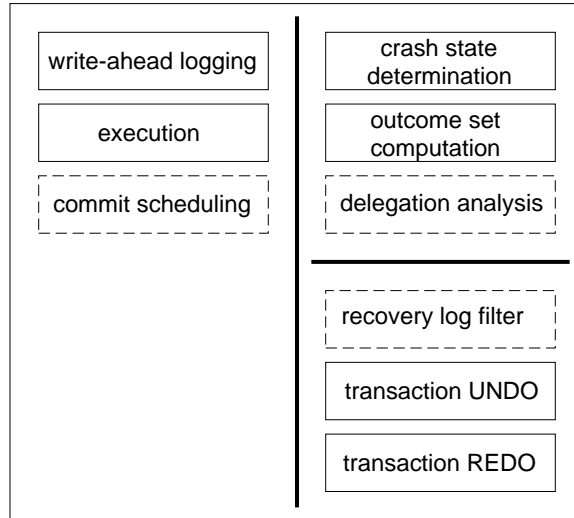


Figure 3: Recovery Microprotocols for the Split/Join Transaction Model

- [9] Calton Pu, Gail E. Kaiser, and Norman Hutchinson. Split-transactions for open-ended activities. In *Proceedings of the 14th VLDB Conference*, pages 26–37, 1988.
- [10] Avi Silberschatz, Mike Stonebraker, and Jeff Ullman. Database research: Achievements and opportunities into the 21st century. Report of an NSF Workshop on the Future of Database Systems Research, May 26-27, 1995, 1995.