

Memory Models for Open-Nested Transactions

Jim Sukha

MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, USA

In this document, we summarize our results from [1], describing the transactional computation framework and the definition of various memory models for transactional memory with open nesting.

The transactional computation framework permits an *a posteriori* analysis of a program’s execution on a TM system. The framework assumes a program execution generates a *trace* which is abstractly represented as a pair (C, Φ) , where C is a “computation tree” that summarizes both the information about the control structure of the program and the nesting structure of transactions, and Φ is an “observer function” that describes the behavior of read and write operations.

The computation tree C is an ordered tree with two types of nodes: *memory-operation nodes* as leaves and *control nodes* as internal nodes. Each memory operation node represents either a read from or a write to a single memory location ℓ . Control nodes capture the computation’s parallel structure. If X is labeled as an S -node, its children are executed in series from left to right. Otherwise, X is a P -node, and its children can be executed in parallel. As we describe in [1], from C we can construct a *computation dag* $G(C)$. Intuitively, every internal node X corresponds to a subdag $G(X)$ represented by the subtree of C rooted at X . In the framework, we mark some subset of internal nodes as transactions. If X is so marked, then $G(X)$ is a transaction. Transactions are further categorized as being committed or aborted, and as being closed or open. Figure 1 illustrates an example computation tree C and its corresponding dag $G(C)$.

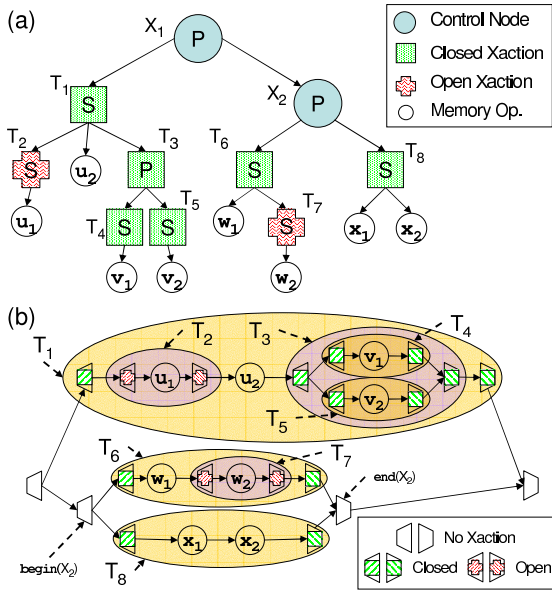


Figure 1. A sample (a) computation tree C and (b) the corresponding dag $G(C)$.

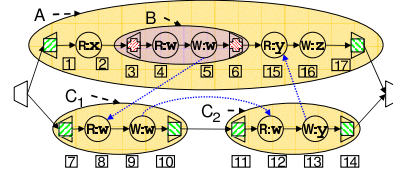


Figure 2. A program trace (C, Φ) which, assuming all transactions are committed, is not serializable or race-free, but is prefix race-free. Nodes are numbered in a legal execution order, 1–17. Blue dotted edges represent the observer function Φ . For example, node 5 denotes a write to location i , and 8 denotes a read from w which observes the value written by 5.

Instead of specifying the value that an operation reads or writes to a memory location ℓ , we follow the computation-centric framework in [2, 3] and abstract away the values entirely using an *observer function* Φ . In the computation dag, consider a memory operation v that either reads from or writes to a memory location ℓ . The node $\Phi(v)$ is defined to be the operation that wrote the value of ℓ that v sees.

In the framework, it turns out that a memory model is defined as a subset of traces “allowed” by that model. In [1], we define two transactional memory models, *race-freedom* and *prefix race-freedom*, and prove the following theorem.

THEOREM 1. *The memory models of serializability, race-freedom, and prefix race-freedom are all equivalent for computations with only closed-nested, committed transactions, but distinct when computations have open-nested transactions.*

Informally, the memory model of *prefix race-freedom* is the set of all traces (C, Φ) such that there exists a topological sort S of the computation dag $G(C)$ that satisfies two conditions. First, for all memory operations v that read from or write to a location ℓ , the node $\Phi(v)$ is the last node in the sort order S which precedes v , writes to location ℓ , and is not “hidden from” v because of an aborted transaction. Second, for every memory operation u “belonging to” a transaction T , in the order S there should be no operations w belonging to an independent transaction T' that appear between $x_{\text{begin}}(T)$ and u which “conflict” with u .

Race-freedom strengthens the second guarantee of prefix race-freedom, to require that there are no operations w between $x_{\text{begin}}(T)$ and $x_{\text{end}}(T)$ which “conflict” with u . Serializability offers the strongest requirement, that no operations w from an independent transaction T' can appear between $x_{\text{begin}}(T)$ and $x_{\text{end}}(T)$. Figure 2 illustrates a trace (C, Φ) which is prefix race-free, but is not race-free or serializable.

We prove that the open-nesting semantics described in [4] guarantees prefix race-freedom, but not race-freedom. Proposed implementations of TM with open nesting in [4, 5] both admit the trace illustrated in Figure 2.

REFERENCES

- [1] K. Agrawal, C. E. Leiserson, and J. Sukha. Memory models for open-nested transactions. In *MSPC*, Oct. 2006.
- [2] M. Frigo. The weakest reasonable memory model. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Jan. 1998.
- [3] M. Frigo and V. Luchangco. Computation-centric memory models. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 240–249, 1998.
- [4] A. McDonald, J. Chung, B. D. Carlstrom, C. Cao Minh, H. Chafi, C. Kozyrakis, and K. Olukotun. Architectural semantics for practical transactional memory. In *ISCA*, June 2006.
- [5] J. E. B. Moss and A. L. Hosking. Nested transactional memory: Model and architecture sketches. In *Science of Computer Programming*, volume 63, pages 186–201. Elsevier, Dec 2006.