The Theory of Transactional Memory: What's There and What's Not

Faith Ellen
University of Toronto
Goal of Transactional Memory:

make it easy to develop correct, efficient concurrent programs
Many papers about:

- ease of correct implementations using TM
- design of a TM + experimental performance results
Semantics:
precise, unambiguous definitions
What does it mean for an implementation of TM to be correct?

it obeys the sequential semantics
What does it mean for an implementation of TM to be correct?

• data items can only be accessed transactionally:

serializability (aka sequential consistency) [L]
strict serializability (aka linearizability) [HW]

for every execution $E$, there is a total ordering of all its committed transactions so that, in the sequential execution obtained by performing these transactions in order, every transaction behaves the same as in $E$ and, if $T$ finishes before $T'$ begins in $E$, then $T$ is ordered before $T'$
What does it mean for an implementation of TM to be correct? What about the uncommitted transactions?

opacity [GK]

for every execution E, there is a total ordering of its transactions so that, in the sequential execution obtained by performing the committed transactions in order, every committed transaction behaves the same as in E, if T finishes before T' begins in E, then T is ordered before T', and each uncommitted transaction in E behaves as if it was performed immediately after the committed transactions ordered before it.
What does it mean for an implementation of TM to be correct?

What about the uncommitted transactions?

- virtual world consistency [IMR]
- snapshot isolation [RFF]
- causal consistency [RTA]
- causal serializability [RTA]
- Z-linearizability [RFSF]
- locally serializable linearizability [GKR]
What does it mean for an implementation of TM to be correct?

- data items can also be accessed nontransactionally:
  - strong atomicity/isolation [BLM]
  - weak atomicity/isolation [BLM]
  - single lock atomicity [HLR]
  - disjoint lock atomicity [MBSAHSW]
  - asymmetric flow ordering [MBSAHSW]
  - encounter time lock atomicity [MBSAHSW]
  - selective strict serializability [SDMS]
  - internal consistency [L']
  - race-free atomicity [SAGNW]
What does it mean for an implementation of TM to be correct?

- data items can also be accessed nontransactionally:
  - privitization [MBSAHSW]
  - publication [HLR, SMDS, DSS]
What is a transaction?

A piece of code that must be executed so that it appears to have been performed atomically.

- transactions that can release variables early [FGG]
- view transactions [AMT]
- elastic transactions [FGG]
- polymorphic transactions [GG]
- boosting [H]
- multiversion semantics [PFK,AH]
- nested transactions [ALS]
- locks inside transactions [DSS]
- checkpointing and continuation inside transactions [KH]
- irrevocable transactions [WSA]
How should the semantics of transactional memory be described?

• operationally [HLR]
• as sets of legal executions [GD,SDMS]
• as automata [DGLM]

- proving that a given implementation satisfies certain correctness conditions
- comparing different correctness conditions [SAGNW,DSS]
Efficiency of TM implementations

Limit when aborts can occur

permissive: a transaction must commit
unless doing so violates correctness [GHK]

progressive: a transaction must commit
if it doesn't conflict with any other transaction [GK']

c-progressive: a transaction must commit
unless it has more than c conflicts

read-only transactions never abort [AH]

It is easy to ensure that no transaction ever aborts
by allowing waiting or helping.
If not, worst case abort rate is large.
Efficiency of TM implementations

**progress/liveness conditions**

**wait-freedom**: every transaction will commit

**non-blocking**: infinitely often, some transaction will commit

**obstruction-freedom**: each transaction will commit if it runs solo for sufficiently long

**local progress**: transactions of every process will commit infinitely often [BGK]
Efficiency of TM implementations

parallelizability

disjoint-access parallelism: two transactions cannot interfere with one another unless they conflict [IR,AHM,EFKMT]
possible/impossible combinations

strict dap + variant of obstruction-freedom   impossible [GK"]
dap + nonblocking                                       possible [HLMS]
dap + wait-freedom                                      impossible [EFKMT]
dap + wait-freedom for operations with bounded data sets possible [EFKMT]
dap + nonblocking + read-only transactions never abort   impossible [PFK]
dap + read-only transactions never abort
assume strict serializability
possible/impossible combinations

opacity + progressiveness

opacity + permissiveness
  + invisible reads

opacity + permissiveness
virtual world consistency
  + probabilistic permissiveness
  + invisible reads

possible [IR']

impossible [CIR]

NP-hard [KP]

possible [CIR]
complexity

opacity + permissiveness [KR]:
$\Omega(r)$ synchronization primitives required for a transaction that reads $r$ data items

opacity + progressiveness [KR]:
$O(1)$ synchronization primitives suffice

opacity + progressiveness + dap:
$\Omega(w)$ synchronization primitives required for a transaction that writes $w$ data items [KR]
complexity

strict serializability + dap + wait-free read only transactions: reading $r$ data items requires updating $\geq r - 1$ base objects [AHM]

strong atomicity + dap + invisible reads + progressiveness/obstruction freedom: the data set of a privitizing transaction must contain all privitized data items [AH'']

strong atomicity + dap + progressiveness/obstruction freedom: in the worst case, a transaction privitizing $k$ data items must access at least $k$ base objects [AH'']
complexity

universal constructions [FK,CER,FK'] and some simple TMs [HTA] have:

• correctness proofs
• bounds on step complexity
• bounds on space
complexity

competitive analysis of contention managers

optimize length of schedule for completing a collection of transactions, as compared to an optimal offline scheduler that knows the conflict graph

- greedy [GHP,SB,SEB]
- Randomized [AEST]
- lower bounds for some restricted classes [AEST,AM]
studying selfish behaviour

If a contention manager gives priority to long transactions, then a transaction can be written to improve its performance at the expense of the system. [EW]
What's there

- semantics: lots of definitions, some relationships between them
- impossibility proofs or lower bounds for certain combinations of properties
- transactional memory designs with certain combinations of properties
- complexity analyses of a few relevant problems
- one game theory result
What's not there

- good models of computation: which aspects are important and which are not
- good complexity measures
- agreement on the right definitions
- decomposition of transactional memory systems into modules
- identification of fundamental and useful algorithmic questions
- innovative algorithm design, complexity analyses, and lower bounds on problems
PEANUT GALLERY
REFERENCES


[AH'] Attiya and Hillel, Single-Version STMs can be Multi-Version Permissive, ICDCN 2011, pages 83-94.


[CIR] Crain, Imbs, and Raynal, Brief Announcement: Read Invisibility, Virtual World Consistency, and Permissiveness are Compatible, SPAA 2011, pages 315-316.


[DSS] Dalessandro, Scott, Spear, Transactions as the Foundation of a Memory Consistency Model, DISC 2010, pages 20-34.


[H] Herlihy, Modular Integration of Type-Specific Concurrency Control with Transactional Memory, WTTM 2012.


The workshop has being supported by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement number 238639, ITN project TransForm and by the European Cooperation in Science and Technology (COST) Action IC 1001 Euro-TM.