Semantics for Transactional Languages

Michael L. Scott

UNIVERSITY *of* ROCHESTER

Workshop on the Theory of Transactional Memory

Rome, Italy

September 2011

A Fleeting Opportunity

- HTM is coming
  - Azul, Sun Rock, AMD ASF
  - IBM has announced for Blue Gene/Q
  - ... ?
- STM for backward compatibility, fallback on HW overflow
- Language support essential
- Narrow window in time to “get the semantics right”

Outline

- Assertions
  - » atomicity is central
  - » speculation is an implementation issue (only)
  - » small transactions are what matter
  - » privatization is essential
    - – necessary for correctness
    - – solves the problem of legacy synchronization
- Open Questions
  - » non-transactional reads and writes
  - » big transactions, integration with system transactions
  - » relationship to “deterministic parallel programming”

Memory Models

- Transactional sequential consistency (TSC)
  - » ideal but expensive: global total order on accesses
    - – consistent w/ program order $<_p$
    - – w/ each transaction contiguous
- Strict serializability (SS)
  - » txns globally totally ordered wrt one another
  - » also ordered wrt preceding & following accesses of same thread (though those accesses aren't necessarily globally ordered wrt one another)
- Read r is permitted to see the value of write w if
  - » r and w access the same location l
  - » $w <_p r \vee w <_{ss} r$, and there is no intervening write of l between w and r

Transactional Data Race Freedom (TDRF)

- An execution E has an (SS) data race if $\nexists <_t$ that induces a $<_{ss}$ that orders all conflicting accesses and explains E's reads
  - » A program has a data race if it has an execution that has a data race
- In analogy to nontransactional models,
  - » if implementation guarantees that
    - – transactions are SS
    - – non-transactional accesses in thread t happen
      - after the commit of the previous transaction in t
      - before the commit of the next transaction in t
  - » and if program P is TDRF
  - » then all of P's executions will be TSC

Strong Isolation Is a Non-Issue

- Hard to explain to the programmer
  - what is a memory access?
- Heavy performance penalty in STM
- Only matters in racy programs
  - constrains the behavior of buggy code
  - less than you want (TSC); more than you need to build what you want (TSC given TDRF)
  - may be useful for debugging, but a good race detector is better

Opacity Is a Semantic Non-Issue

- Aborted transactions do not appear in (language-level) histories
- Opacity is simply one end of the implementation spectrum: validate at every read
- Sandboxing is the other end: validate before every “dangerous” operation (and periodically)
- Some very promising implementations in the middle: delayed/out-of-band validation
  - ask me later!

Privatization Is Essential

- Definition: transaction T with history prefix P privatizes datum D if
  - $\exists$ extensions of P in which a first access to D after P occurs in different threads
  - $\forall$ extensions of P+T, the first access to D after P+T occurs in the same one thread
- Crucial for performance with STM
- Solves the problem of legacy synchronization
  - locking is privatization — `acquire` and `release` are small atomic blocks
- (Publication is a non-issue: implementation chalenges arise only in racy programs)

Transactions ≠ Critical Sections

~~L.acquire()~~ ~~atomic {~~
~~...~~ ~~≡~~ ~~...~~
~~L.release()~~ ~~}~~

```
L.acquire()              atomic { ... }
  ...            ≡         ...
L.release()              atomic { ... }
```

Open Questions

Non-transactional Accesses

- Want reads for, e.g., ordered speculation, high-performance hybrid TM
  - » clearly important at the HTM ISA level
  - » not clear whether needed/wanted at language API level
- Want writes out of aborted txns for debugging
  - » again, clearly important at the HTM ISA level
    - – and probably more useful if immediate
  - » probably important at the language level too
    - – not as clear that these need to be immediate
- Immediate writes, and writes in aborted txns, a challenge for the memory model
- Other compelling uses? (esp. in small txns)

Atomicity and Determinism

- Recall Li's talk this afternoon
  - languages/idioms that guarantee all abstract executions will be “equivalent” in some well-defined sense
- Independent split-merge an obvious foundation for language-level determinism
- Atomic commutative [+associative] ops the obvious extension
- Is there anything else?

And of Course...

- Abort, orElse? (conjecture: no)
- Bigger transactions? Integration with system transactions? (again, conjecture: no)

The Bottom Line: Keep It Simple!

- Atomicity is central
- Speculation is an implementation issue (only)
- Small transactions are what matter
- Privatization is essential (and solves the problem of legacy synchronization)

1850

MELIORA

UNIVERSITY of ROCHESTER

www.cs.rochester.edu/research/synchronization/

Thanks to Luke Dalessandro, Li Lu