Zombies Are Our Friends

or: Why You Don't Need (or Want) Opacity

Michael L. Scott

TRANSACT 2015 "lightning talk"

Zombie Transactions
(a.k.a. doomed or orphan)

- Have seen inconsistent state, but haven't noticed yet
  - will not be able to commit
- Useful in real-world systems
  - Casper et al. [ASPLOS 2011] (1.7×); Kestor et al. [PACT 2011] (1.8–5.2×); Dalessandro & Scott [PACT 2012]; IBM Blue Gene/Q; lazy subscription
- Forbidden by opacity (Guerraoui & Kapałka [PPoPP 2007])

Semantics & Levels of Abstraction

- Different formalisms appropriate at different levels
  - language level: TSC for DRF programs
    - no notion of aborted txns
  - run-time level: TM API (start, read, write, try-commit, ...)
  - implementation level: conventional memory model (e.g., C'11) (assuming building STM)
- Run-time level is where zombies matter
  - not an issue above or below

Sequential Semantics for TM

Specify the meaning of sequential histories (interleavings of the TM-relevant ops of your threads)

- *API* (start, read, write, try-commit, validate, abort, ...)
- *memory model:* what can reads see?
- *conflict function:* which concurrent transactions cannot both commit? (which give me permission to become a zombie?)

Safety and Liveness

A TM implementation is correct if

1. we can prove the *fundamental theorem of TM:*

   every sequential history is equivalent to a serial history

1. try-commit fails only in the presence of a conflicting txn
2. read *r* in unsuccessful txn *T* is inconsistent w/ previous reads only given a txn *S* whose prefix prior to *r* conflicts w/ *T*
3. zombie execution is bounded (given any history prefix, if T can run arbitrarily long w/out completing, it can do so consistently)
4. exceptions never escape an unsuccessful txn

Opacity v. SSTM

- Serializability & consistency
  - fundamental in opacity — but behavior of individual API methods is unspecified
  - *flow from* the memory model & conflict function in SSTM — but every choice for these induces different semantics, and requires a new proof of the fundamental theorem
- Zombies
  - forbidden in opacity
  - allowed in SSTM: compiler's responsibility to validate when necessary — i.e., to sandbox

Please read the position paper!

www.cs.rochester.edu/u/scott/papers/2014_WTTM_zombies.pdf

UNIVERSITY of ROCHESTER