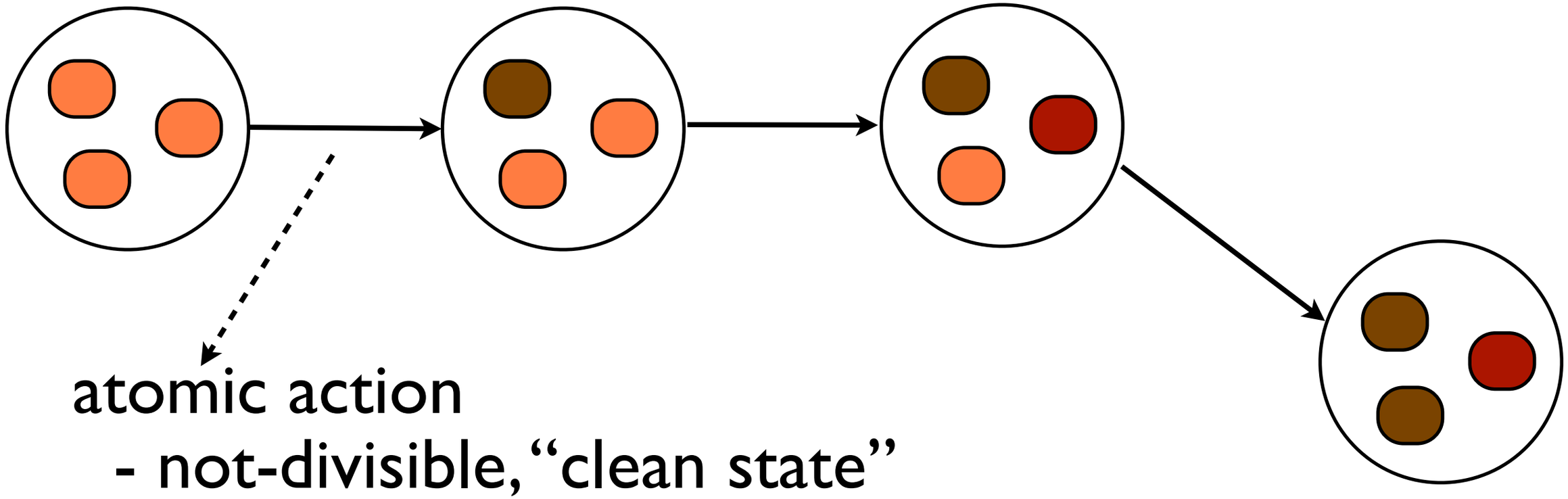
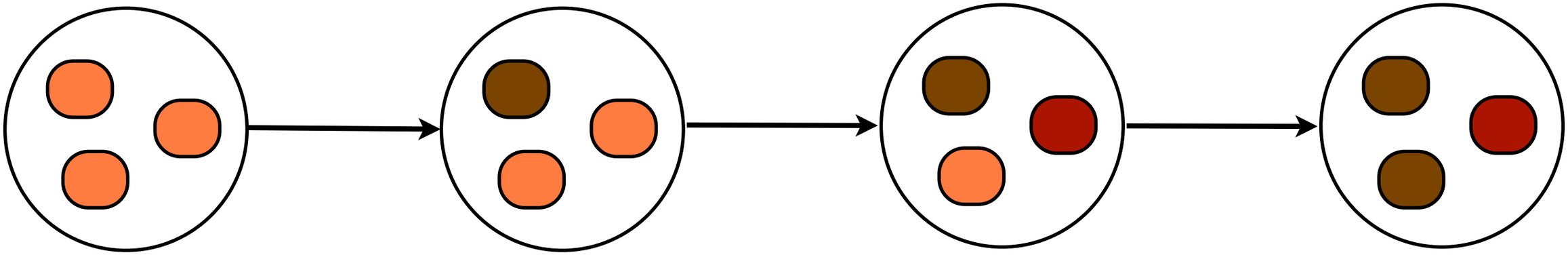
Dealing with Concurrency and Failures

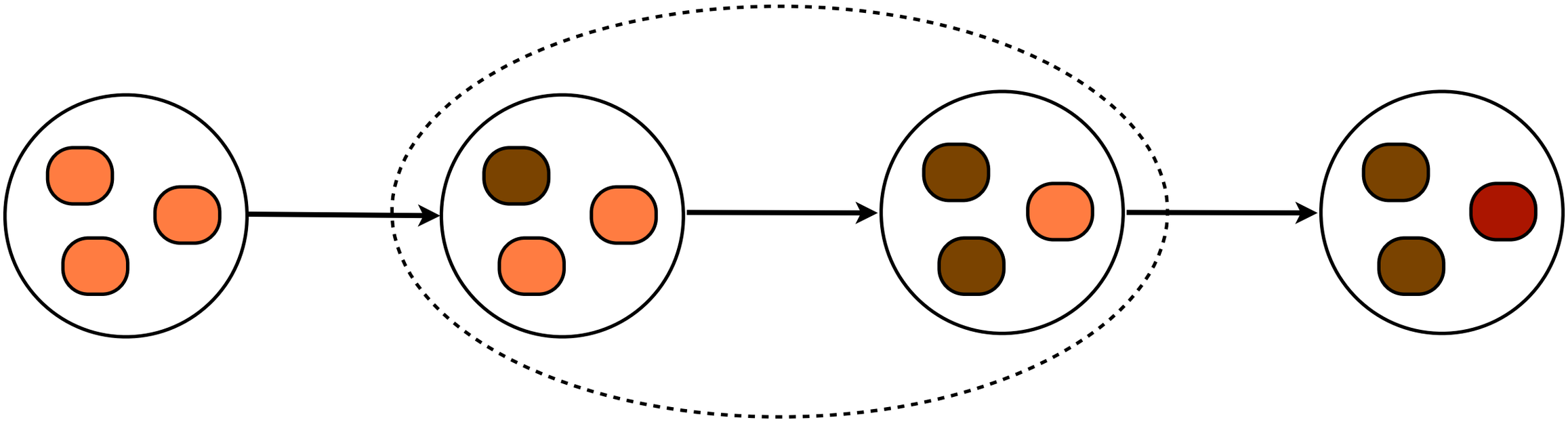


atomic action

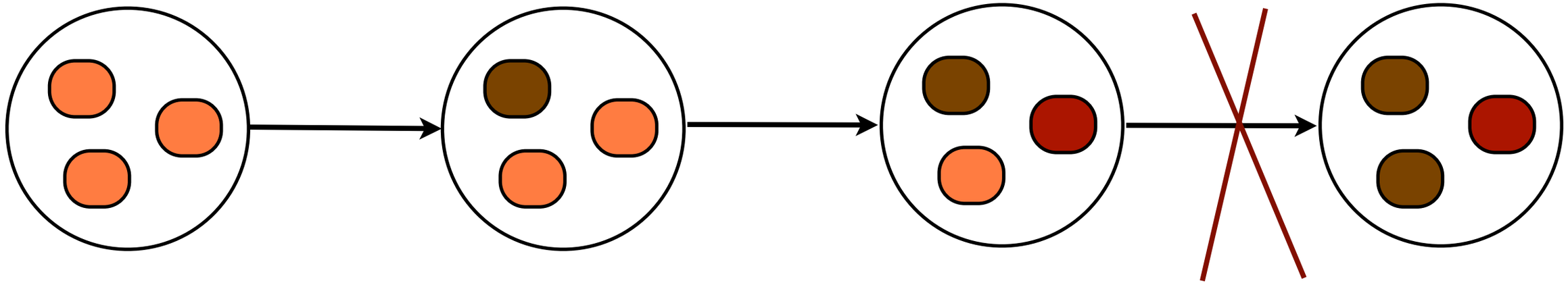
- not-divisible, “clean state”
- (processor instructions, syscalls)

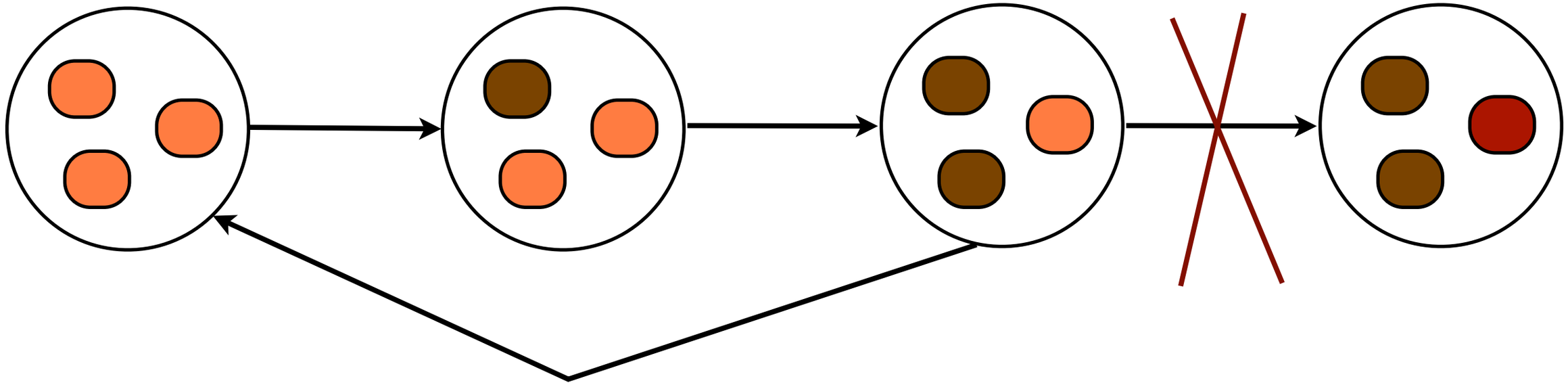
Concurrency: might introduce unwanted traces





Failures: might introduce unwanted intermediary state





High level atomic actions

Do it yourself: locks, spin locks, atomic set, log

Hard: tons of research on identifying concurrency bugs:
MUVI (SOSP'07), ConSeq(ASPLOS'11)

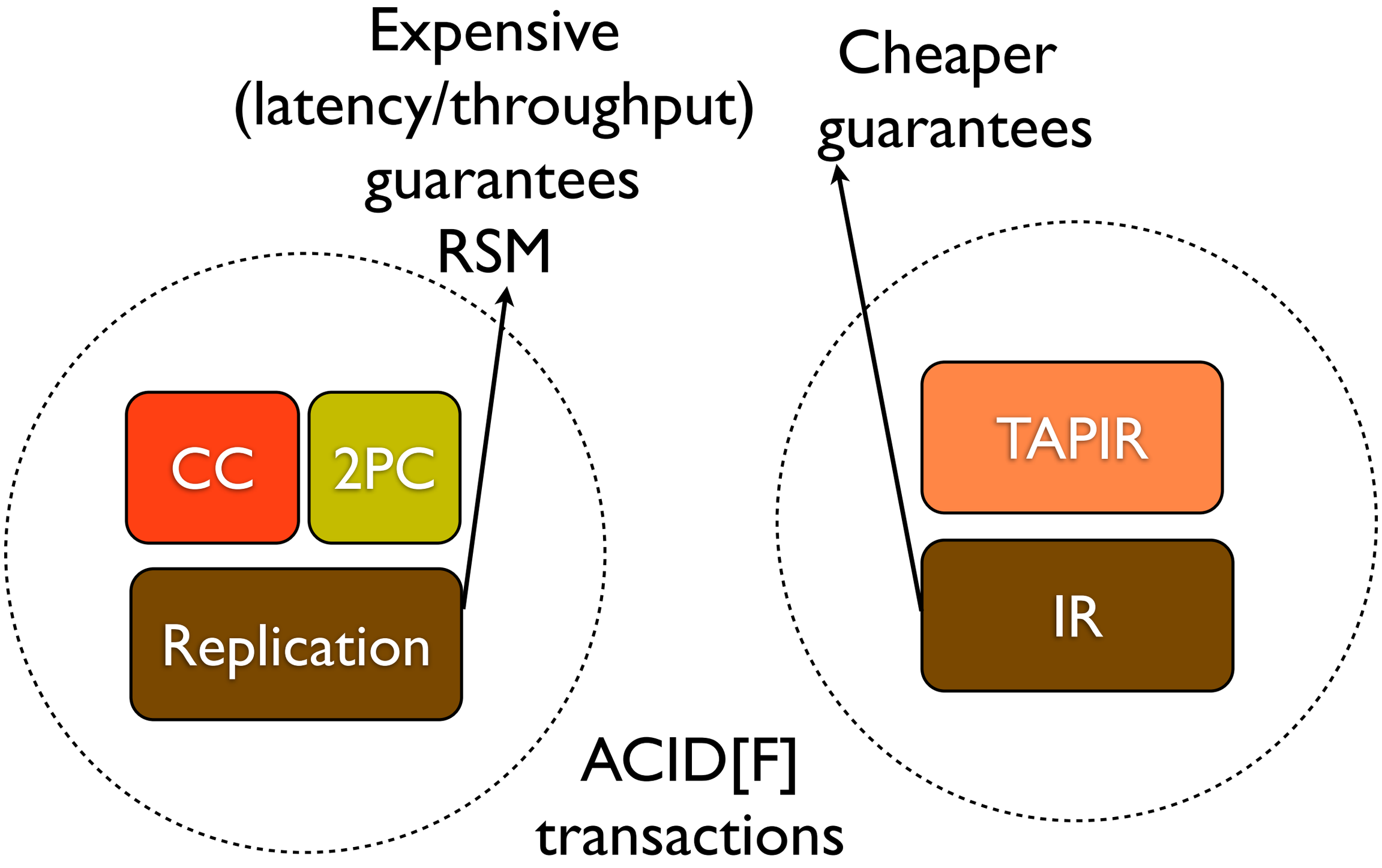
Transactions

- Operating system transactions
  - TxOS(SOSP'09)
- Transactional memory
  - transactions on app data structures
  - a bunch of references in TxOS paper
  - Sinfonia(SOSP'07)
- Storage/database transactions
  - consistent application state

TAPIR

- In-memory, distributed, fault-tolerant key-value store
- supports distributed ACID[F] transactions

The idea


Expensive
(latency/throughput)
guarantees
RSM
CC
2PC
Replication
Cheaper
guarantees
TAPIR
IR
ACID[F]
transactions

Architecture