Specification and verification of consistency models

Paolo Viotti
UPMC - work done at Eurecom

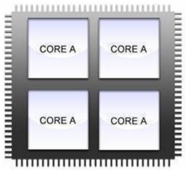
RainbowFS kickoff meeting
May 4, 2017

Consistency: an introduction

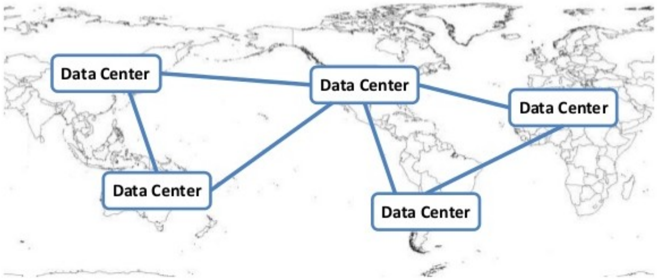
The ability of a storage system of maintaining a certain *correct* state (despite concurrency, partial failures and asynchrony).

- both *safety* and *liveness*

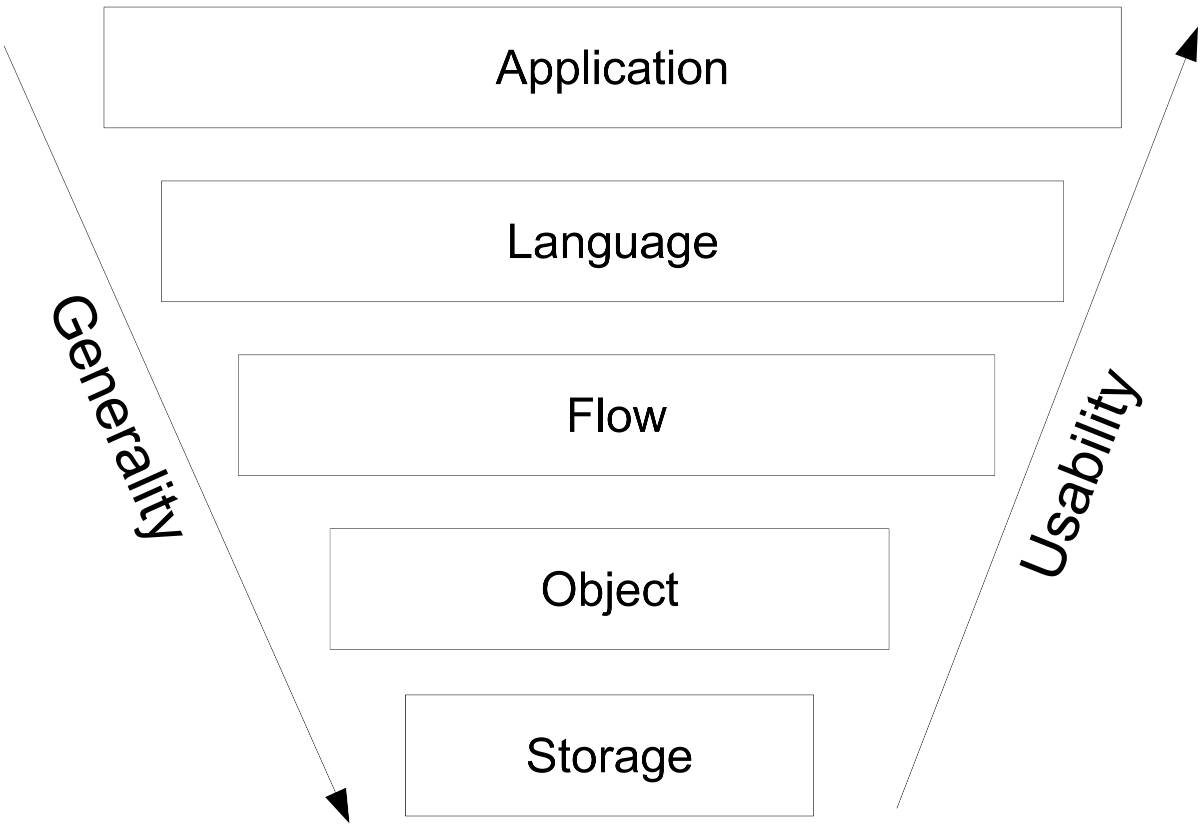
- Where
  - Architectures: shared memory, shared disk, shared nothing
  - Databases: relational, NoSQL
    - Transactional and non-transactional
  - From CPU caches to geo-replicated systems



- Tradeoffs
  - CAP, latency



Full stack consistency



CISE tool

Bloom, Lasp

Blazes

CRDTs, LVars

Survey, Conver

Outline

Specifying consistency

An axiomatic approach

[ACM Comput. Surv. '16]

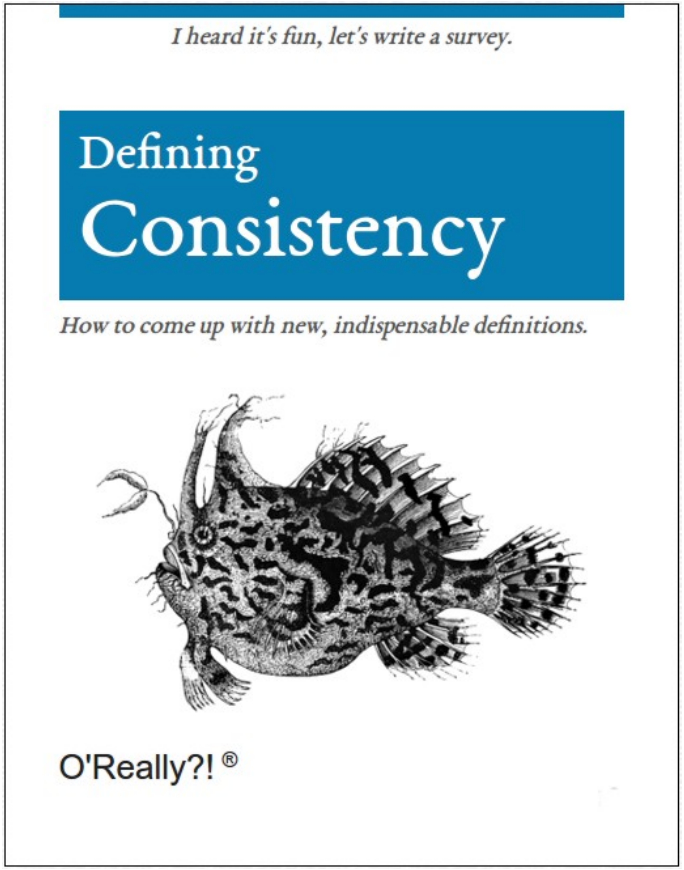
Verifying consistency

Conver

[ACM PaPoC '16]

Specifications of consistency models

- State-of-the-art definitions
  - Informal, imprecise
  - Incompatible
- Different contexts
  - Shared-memory systems
  - Databases
  - Outsourced/cloud storage



Consistency specifications:

Operational vs. Axiomatic

<u>Based on ref. implementation</u>

- ✔ Robust specifications
- ✔ Refinement mappings to prove implementations correct

- ✗ Easy to over-specify
- ✗ Weak models spec. are unwieldy

<u>Based on logical conditions</u>

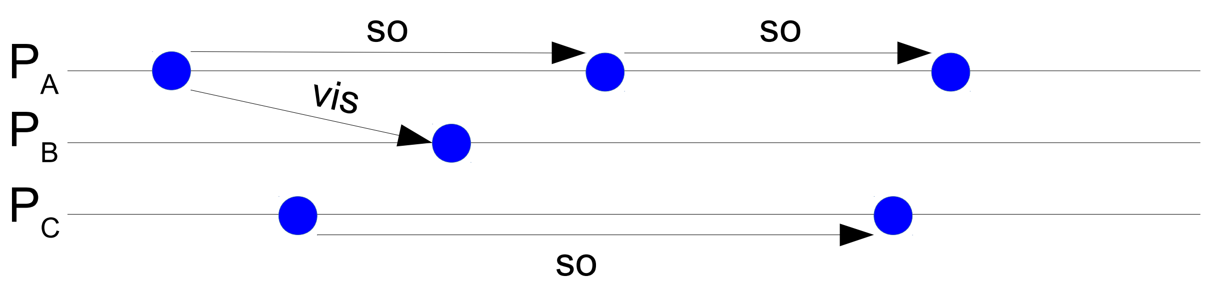
- ✔ Composable
- ✔ Meaningful for users/designers
- ✔ Concise (even for weak models)
- ✔ Abstract away implementation

- ✗ Easy to get axioms wrong

Axiomatic consistency specifications

Consistency semantics as logic *predicates*

about ***ordering*** and ***visibility*** of events



A model for axiomatic specifications

refining and extending [Burckhardt ‘14]

- Processes, objects, operations
- Execution ↔ **History** (set of operations)

- Relations on history
  - **rb**: returns-before partial order
  - **ss**, **so**: eq. relation and partial order on sessions
  - **ob**: equivalence relation on objects

- **Abstract execution** = history, vis, ar
  - **vis**: visibility, tracks propagation of writes
  - **ar**: arbitration total order, how system resolves conflicts

A model for axiomatic specifications
refining and extending [Burckhardt ‘14]

- Operation **context**
  - Model state of execution as graph
  - Projection on abstract execution

- Return-value consistency
  - “Expected” set of return values according to context and…
  - …to the replicated data type (set, queue, register…)

- <u>Consistency models as logic predicates on abstract executions</u>

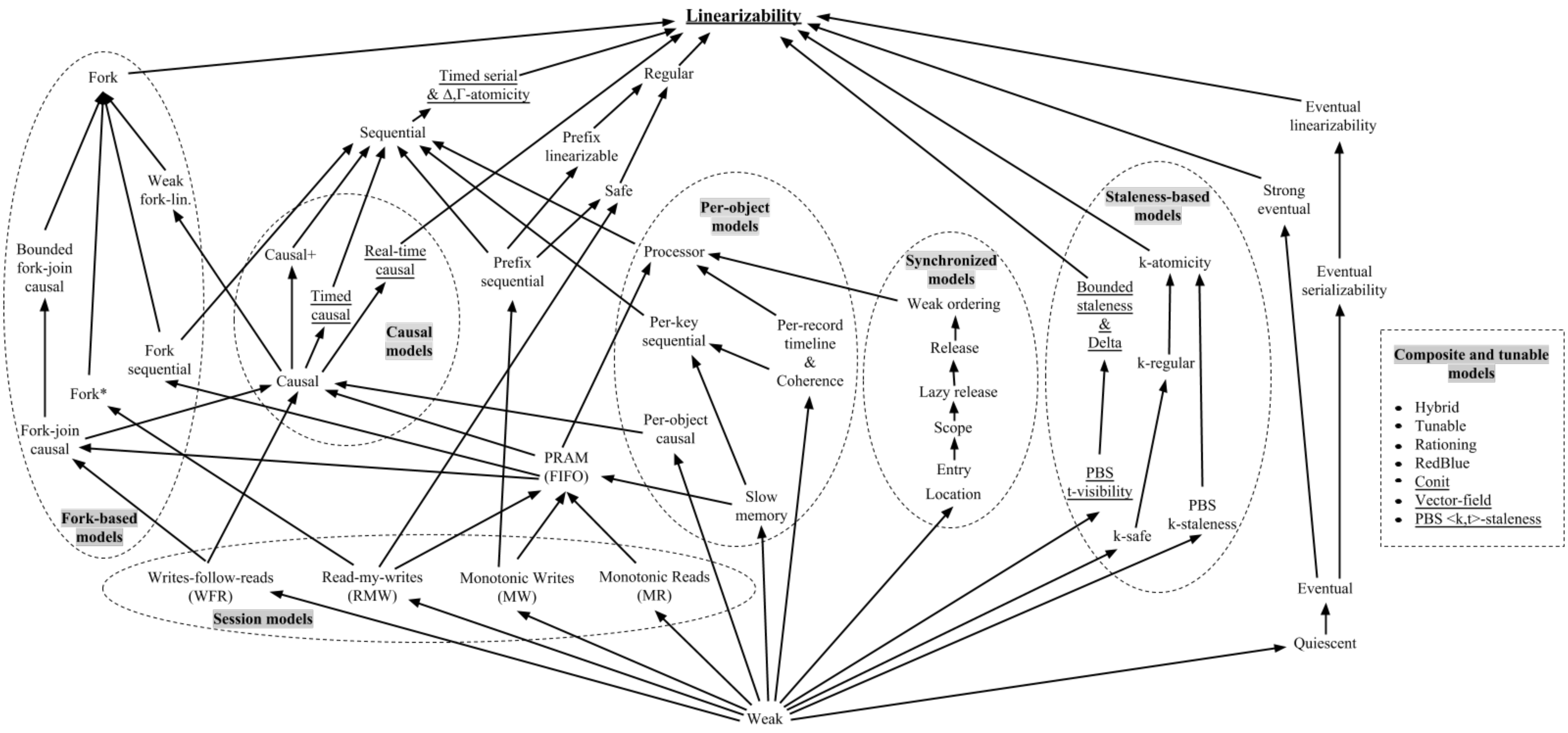
$$H \models \mathcal{P}_1 \wedge \cdots \wedge \mathcal{P}_n \Leftrightarrow \exists A \in \mathcal{A} : \mathcal{H}(A) = H \wedge A \models \mathcal{P}_1 \wedge \cdots \wedge \mathcal{P}_n$$

A survey of consistency semantics

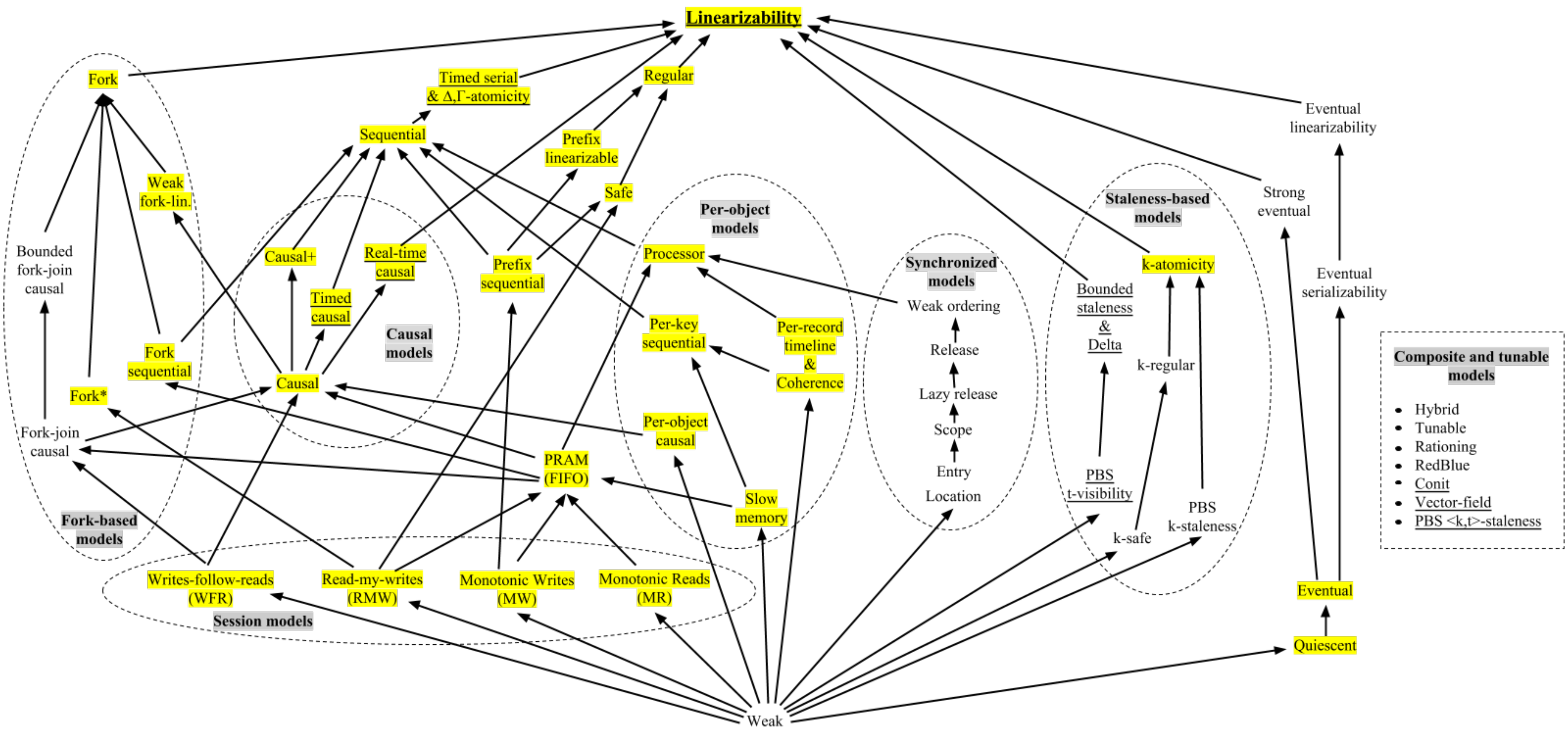
40+ predicates from 30+ years of research

Linearizability($\mathcal{F}$)	SingleOrder $\wedge$ RealTime $\wedge$ RVal($\mathcal{F}$)
SingleOrder	$\exists H' \subseteq \{op \in H : op.oval = \nabla\} : vis = ar \setminus (H' \times H)$
RealTime	$rb \subseteq ar$
Regular($\mathcal{F}$)	SingleOrder $\wedge$ RealTimeWrites $\wedge$ RVal($\mathcal{F}$)
Safe($\mathcal{F}$)	SingleOrder $\wedge$ RealTimeWrites $\wedge$ SeqRVal($\mathcal{F}$)
RealTimeWrites	$rb\|_{wr \to op} \subseteq ar$
SeqRVal($\mathcal{F}$)	$\forall op \in H : Concur(op) = \emptyset \Rightarrow op.oval \in \mathcal{F}(op, cxt(A, op))$
EventualConsistency($\mathcal{F}$)	EventualVisibility $\wedge$ NoCircularCausality $\wedge$ RVal($\mathcal{F}$)
EventualVisibility	$\forall a \in H, \forall [f] \in H/\approx_{ss}: \|\{b \in [f] : (a \xrightarrow{rb} b) \wedge (a \xrightarrow{vis}\!\!\!\!\!\!\!/\;\; b)\}\| < \infty$
NoCircularCausality	$acyclic(hb)$
StrongConvergence	$\forall a, b \in H\|_{rd} : vis^{-1}(a)\|_{wr} = vis^{-1}(b)\|_{wr} \Rightarrow a.oval = b.oval$
StrongEventualCons.($\mathcal{F}$)	EventualConsistency($\mathcal{F}$) $\wedge$ StrongConvergence
QuiescentConsistency($\mathcal{F}$)	$\|H\|_{wr}\| < \infty \Rightarrow \exists C \in \mathcal{C} : \forall [f] \in H/\approx_{ss}: \|\{op \in [f] : op.oval \notin \mathcal{F}(op, C)\}\| < \infty$
PRAM	$so \subseteq vis$
SequentialConsistency($\mathcal{F}$)	SingleOrder $\wedge$ PRAMConsistency $\wedge$ RVal($\mathcal{F}$)

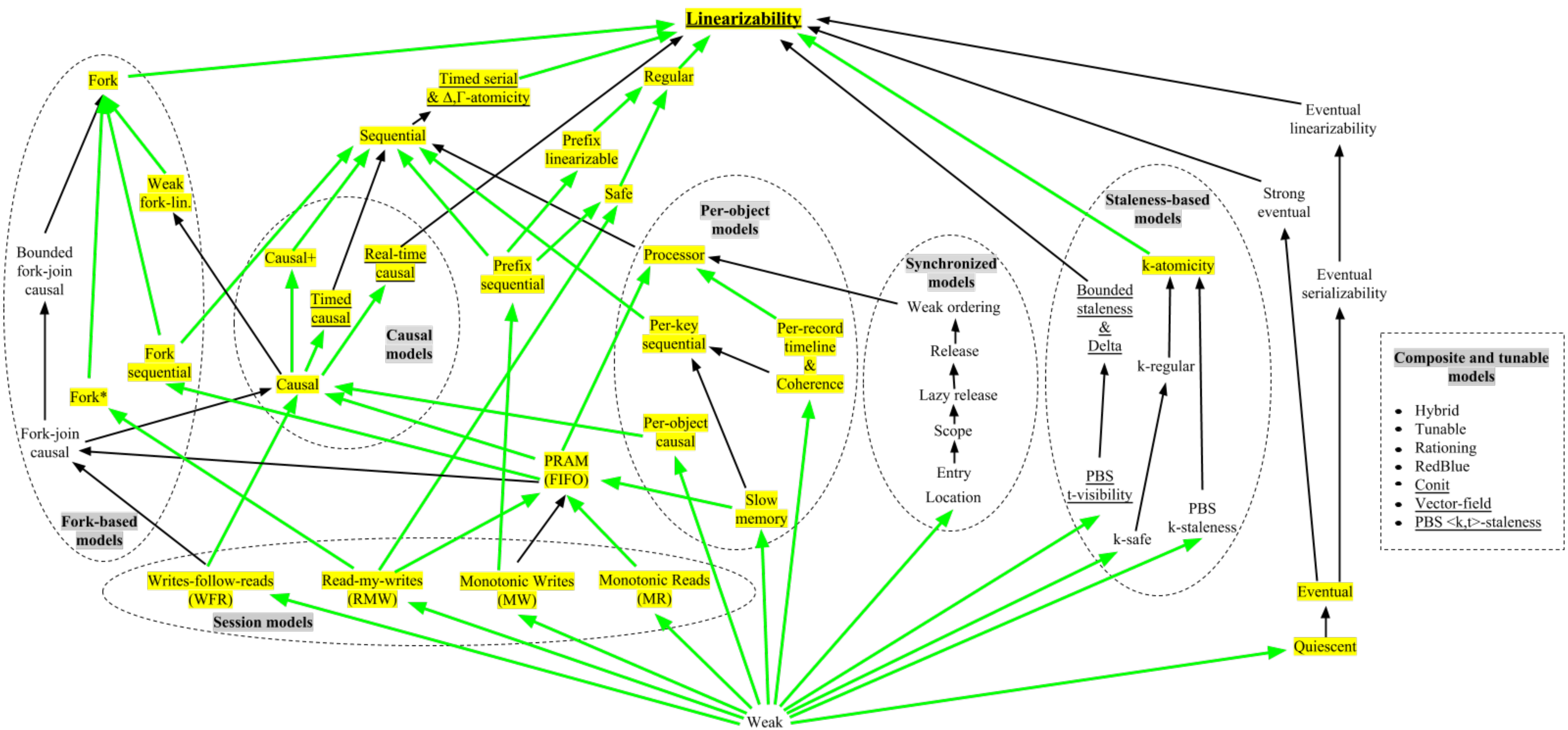
A partial ordering of models



A partial ordering of models



A partial ordering of models



Outline

Specifying consistency	**Verifying** consistency
An axiomatic approach	*Conver*
[ACM Comput. Surv. ‘16]	[ACM PaPoC ‘16]
	

Consistency and the real world

CONSISTENT!*

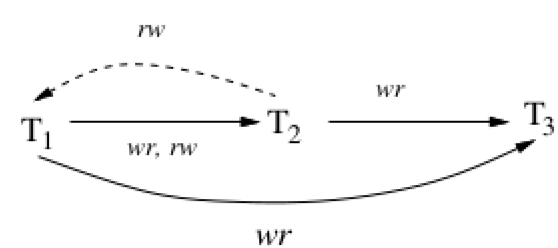
*: terms and conditions may apply.

Verifying consistency: state of the art

- Strong consistency checkers
  - binary decision problem

- Staleness
  - for eventually consistent clouds

- Precedence graph
  - transactional systems



- Application-level invariant checkers

Verifying consistency: theoretical results

- Linearizability
  - NP-complete (**polynomial**)* [Gibbons et al., ‘97]
  - Model checking, 1 execution: EXPSPACE [Alur et al., ‘00]
- Sequential consistency
  - Combinatorial problem, 1 execution: NP-complete [Gibbons et al., ‘92]
  - Model checking, 1 execution: undecidable [Alur et al. ‘00]
- Causal consistency [Bouajjani et al., ‘17]
  - Implementation: undecidable (decidable)*
  - 1 execution: NP-complete (**polynomial**)*
- Eventual consistency*
  - Model checking, 1 execution: EXSPACE-hard [Bouajjani et al., ‘14]

Testing distributed systems

CONFIDENCE

- Traditional testing, distributed tracing, monitoring
  - *Dapper*, *Zipkin*, ...printf( )
- “Smart testing”
  - Property-based testing, fault injection (*Jepsen*), directed random tests, deterministic simulations

Conver

- Formal methods
  - Model checking
  - Correctness-by-construction (Coq, TLA+…): *Verdi, IronFleet, Chapar*
  - “Lightweight FM”: invariants verification through SMT: CISE tool

EASE OF USE

Property-based consistency verification

Verify consistency semantics as

axiomatic invariants of executions

Property-based testing

A simple example (in Erlang):

Function to reverse a list

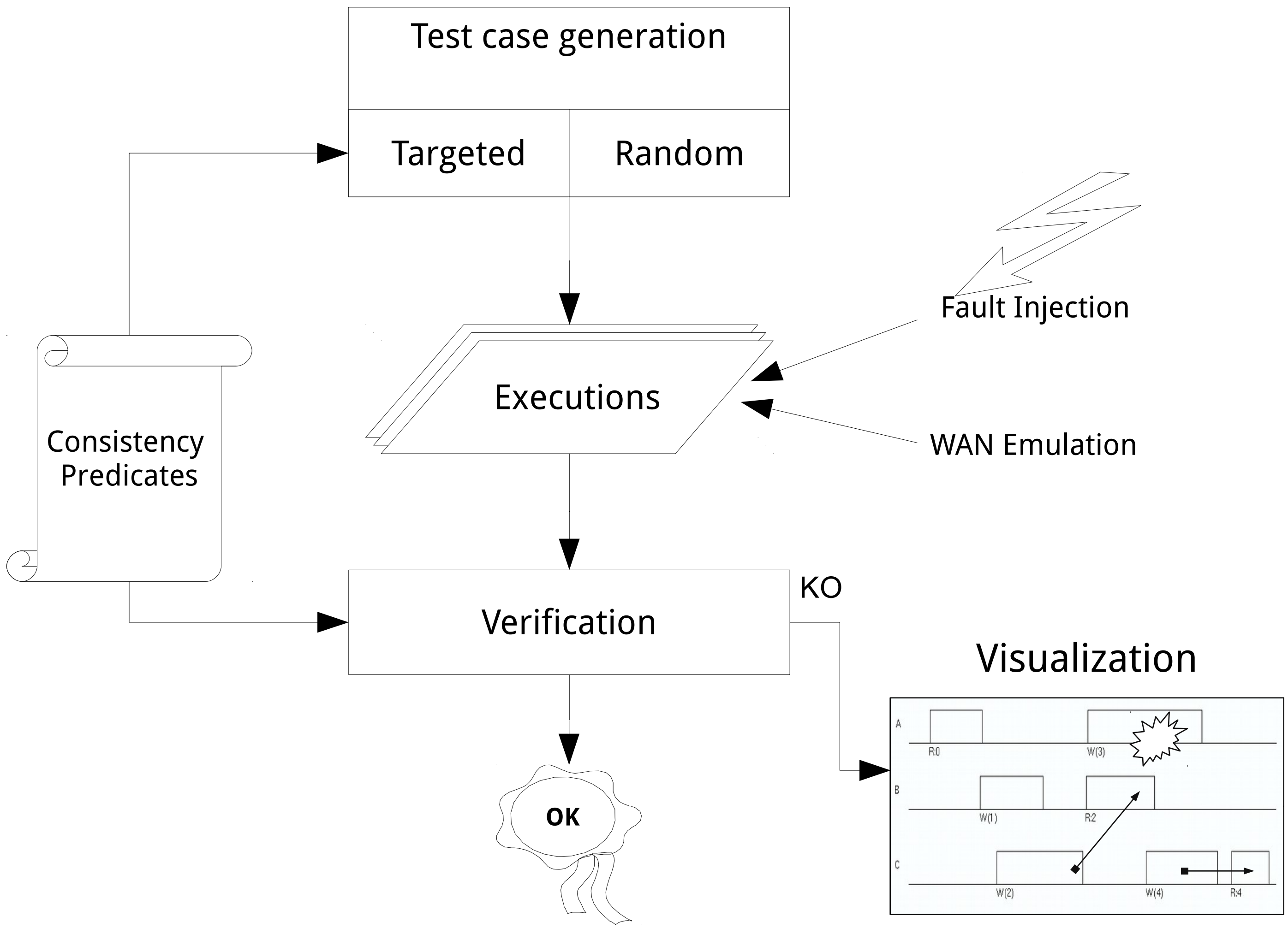
```
reverse([]) ->
    [];
reverse([X|Xs]) ->
    reverse(Xs) ++ [X].
```

Property:
for every list Xs, reverse(reverse(Xs)) == Xs

```
prop_reverse() ->
    ?FORALL(Xs, list(int()),
            reverse(reverse(Xs))==Xs).
```

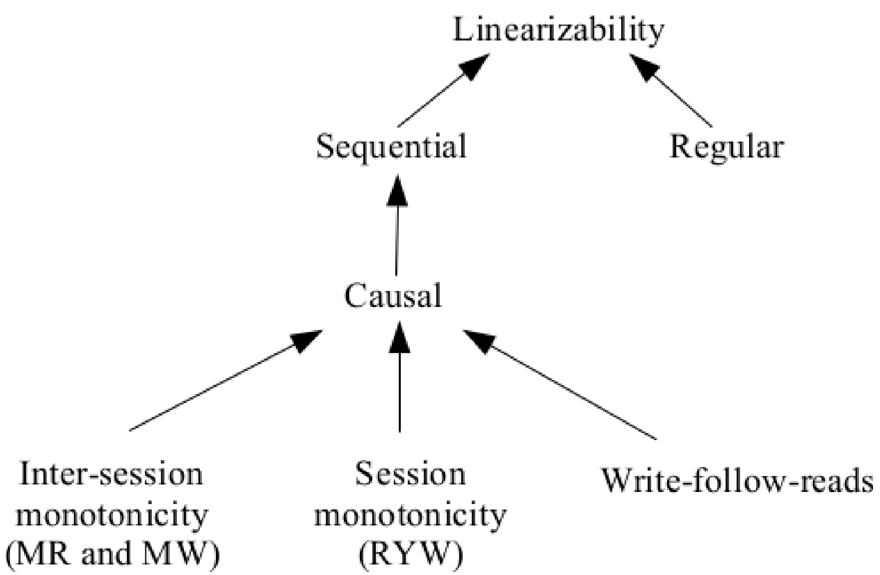
```
3> proper:quickcheck(qc_test:prop_reverse()).
...............................................................
OK: Passed 100 test(s).
true
```

Conver: architecture



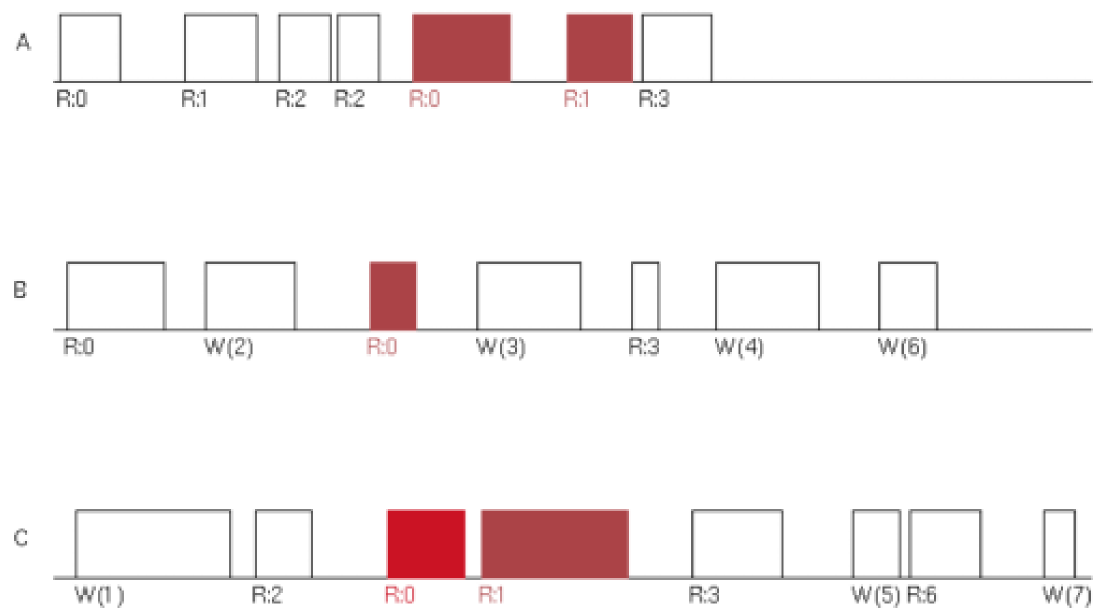
Conver prototype

- Open source prototype in Scala
  - github.com/pviotti/conver-scala
- Automatic local deployment with Docker
- Can verify 7 consistency models
- 2 data stores (Riak, ZooKeeper)
  - easily extensible



Conver - outputs

```
Started. Database: zk, n. clients: 10, avg op/client: 10
Server zk1 started: 172.18.0.2/16
Server zk2 started: 172.18.0.3/16
Server zk3 started: 172.18.0.4/16
Client connecting to 172.18.0.2:2181,172.18.0.3:2181,172.18.
Client connecting to 172.18.0.2:2181,172.18.0.3:2181,172.18.
Client connecting to 172.18.0.2:2181,172.18.0.3:2181,172.18.
Client connecting to 172.18.0.2:2181,172.18.0.3:2181,172.18.
Client connecting to 172.18.0.2:2181,172.18.0.3:2181,172.18.
Client connecting to 172.18.0.2:2181,172.18.0.3:2181,172.18.
Client connecting to 172.18.0.2:2181,172.18.0.3:2181,172.18.
Client connecting to 172.18.0.2:2181,172.18.0.3:2181,172.18.
Client connecting to 172.18.0.2:2181,172.18.0.3:2181,172.18.
Client connecting to 172.18.0.2:2181,172.18.0.3:2181,172.18.
Cycle found: Cycle(b:w:26, b:w:26~>d:w:19 ''ar, d:w:19, d:w:
Removing edge from cycle: b:w:26~>d:w:19 ''ar
Cycle found: Cycle(j:w:23, j:w:23~>d:w:19 ''ar, d:w:19, d:w:
No vertex ordered by rb found in cycle, removing random edge
Total order (tentative): a:r:0 f:w:1 e:r:1 c:w:2 h:r:1 j:w:3
0 a:w:7 c:r:7 f:w:12 j:w:14 d:w:11 e:r:14 b:r:14 h:w:13 i:w:
:20 d:w:19 c:r:19 b:r:19 h:w:21 f:r:19 j:w:23 c:r:23 i:w:22
 h:w:27 d:w:28 i:r:31 b:w:30 a:w:31 g:r:31 e:w:32 j:w:33 c:r
w:35 i:w:36 b:w:37 g:r:36 a:r:36 c:r:37 j:r:36 e:w:38 f:w:40
0 i:w:42 h:r:42 a:w:43 g:w:44 i:r:43 g:r:43 g:w:47 h:w:46 d:
:r:47 g:w:48 a:r:48
Anomalies: d:r:19 i:r:31
Linearizability.............................[KO]
Regular.....................................[KO]
Sequential..................................[OK]
Causal......................................[OK]
Session causality (WFR).....................[OK]
Inter-Session Monotonicity (MR, MW).........[OK]
Intra-Session Monotonicity (RYW)............[OK]
```



Summary

- Declarative/axiomatic specifications of consistency models
  - To reason about and compare them
  - To verify real-world implementations

Future work

- Prove strength relations between consistency models
- Extend Conver
  - Transactional semantics
  - Map application invariants to storage semantics