**Just-Right Consistency**

As available as possible  
As consistent as necessary  
Correct by design

Marc Shapiro, UPMC-LIP6 & Inria  
Annette Bieniusa, U. Kaiserslautern  
Nuno Preguiça, U. Nova Lisboa  
Christopher Meiklejohn, U. Catholique de Louvain  
Valter Balegas, U. Nova Lisboa

---

**Bridging the CAP gap**

\[ CP \cap AP = \emptyset \]

No single consistency model is best for all applications

Insight: Maintain invariants  
- Preserve sequential patterns  
- Synchronise only when strictly necessary for application \( \Rightarrow \) tools

---

**FMK Fælles Medicinkort**

Dr Alice \rightarrow Rx \rightarrow  
- create (...)  
- add-med (...)  
- get-med (...)  
- process (...)  

Drug: 2 boxes \( \rightarrow \) 1

Causatin: 2 boxes \( \rightarrow \) 1

Dr Alice  
Aalborg Hospital  
Patient: Mr Bob  
Pharmacy: Byrum pharma

---

**FMK invariants**

Dr Alice \rightarrow Rx \rightarrow  
- create (...)  
- add-med (...)  
- get-med (...)  
- process (...)  

Drug: 2 boxes \( \rightarrow \) 1

Causatin: 2 boxes \( \rightarrow \) 1

Dr Alice  
Aalborg Hospital  
Patient: Mr Bob  
Pharmacy: Byrum pharma

---

[AntidoteDB & Just-Right Consistency]
Geo-distrib: invariants?

AP-compatible programming constructs

Available under partition
⇒ no synchronisation
⇒ asynchronous updates
⇒ fast response

AP-compatible:
• CRDT data model
• Relative-order pattern
• Joint-update pattern

AP data model: CRDTs

Concurrent, asynchronous updates
• Standard register model: assignments ⇒ CP
• AP ⇒ concurrent updates merged

CRDT: register, counter, set, map, sequence
• Extends sequential type
• Encapsulates convergent merge

Relative order is AP-Compatible

create-p before add-pp
• Referential integrity
  ‣ \( x \) valid ∧ \( x \) points to \( y \) ⇒ \( y \) valid
• \( \text{admin-login-enabled} \) ⇒ \( \text{non-default-password} \)

RHS := true; LHS := true
Transmit in the right order!
AP-compatible: Causal Consistency
Joint update is AP-Compatible

Create-p updates doctor, patient & pharmacy record
Transmit joint updates together
  • write-atomic
  + Read from common set of txns
  • snapshot property
= All-or-Nothing (A of ACID)
AP-compatible

CAP-sensitive invariants

process-p (..., nb) {
  if cnt ≥ nb    // precondition at source
    cnt = nb     // at every replica
} // cnt ≥ 0

Precondition stable w.r.t. concurrent add-med
Concurrency OK

CAP-sensitive invariants

process-p (..., nb) {
  if cnt ≥ nb    // precondition at source
    cnt = nb     // at every replica
} // cnt ≥ 0
**CAP-sensitive invariants**

```
process-p (... , nb) {
  if cnt ≥ nb // precondition at source
    cnt -= nb // at every replica
}
```

Precondition not stable w.r.t. concurrent `process-p`
- Forbid concurrency? Synchro, CP.
- Or remove invariant? AP, degraded semantics

---

**CISE tools**

Static analysis of any application:
- Operations, invariants
- Does each individual op maintain invariant?
- Do concurrent updates converge?
- Is precondition of `u` stable w.r.t. concurrent `v`?
  - If not:
    - Change specification (invariant)
    - or Synchronise
    - Designer decision, per pair `(u, v)`

Ex: medication count= `inc||inc, inc||dec, dec||dec`

---

**Just-Right Consistency**

Methodology for provably ensuring

As Available as Possible, Consistent Enough

TCC ⇒ AP-compatible invariants
CAP-sensitive invariants: Bounded Ctr, CISE

AntidoteDB:
- CRDTs
- Causal Consistency
- Transactions
- Bounded Counter
- AP-compatible
- CP when necessary

CRDT data model
- Register, counter, set, map, sequence
- Extend sequential semantics
- AP compatible

Transactional Causal Consistency (TCC)
- Strongest AP-compatible model
- Joint Updates / Transactional
- Partial Order / Causal Consistency

Open source, well engineered
Community of users