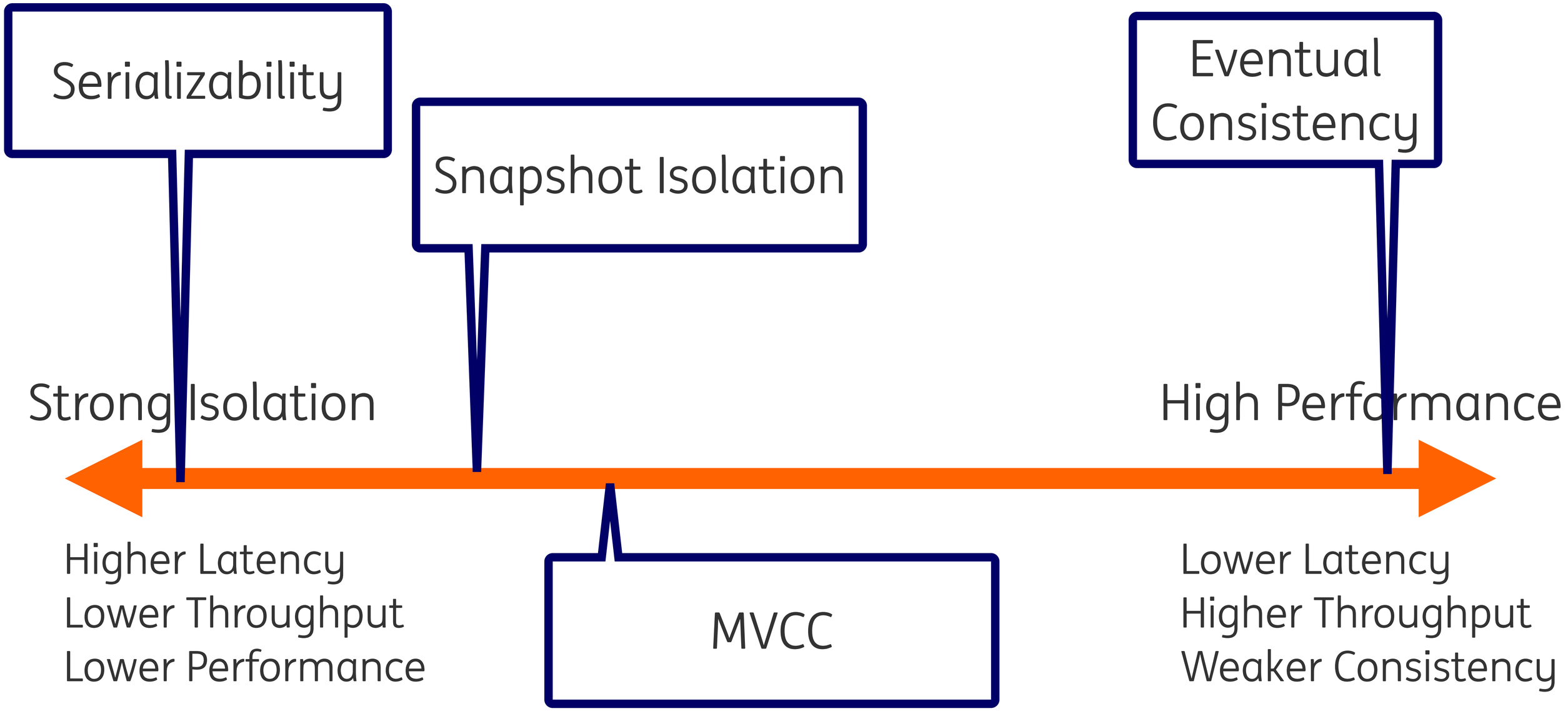
Automated Validation of State-Based Client-Centric Isolation with TLA+

Tim Soethout (tim.soethout@ing.com), Tijs van der Storm, Jurgen Vinju

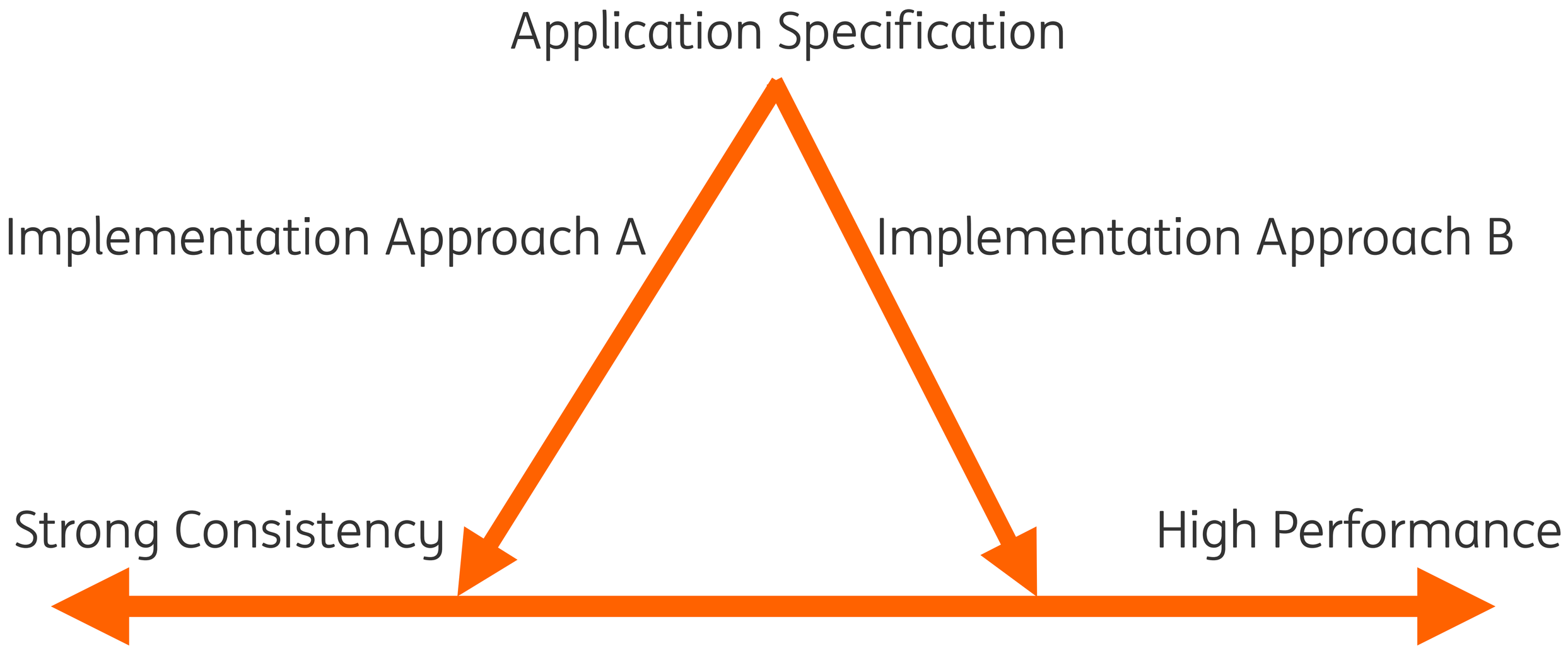
ASYDE'20, 15 September 2020

Centrum Wiskunde & Informatica

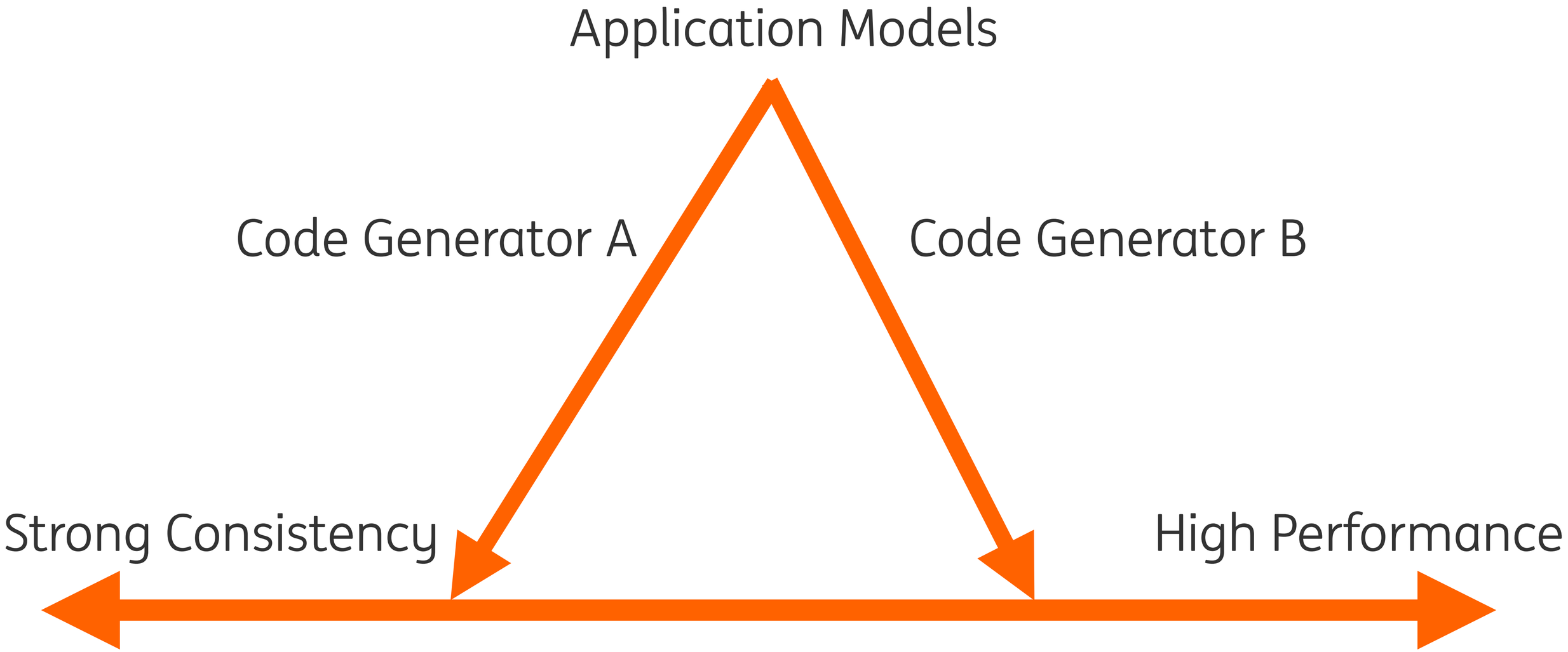
Trade-off: Isolation vs Performance



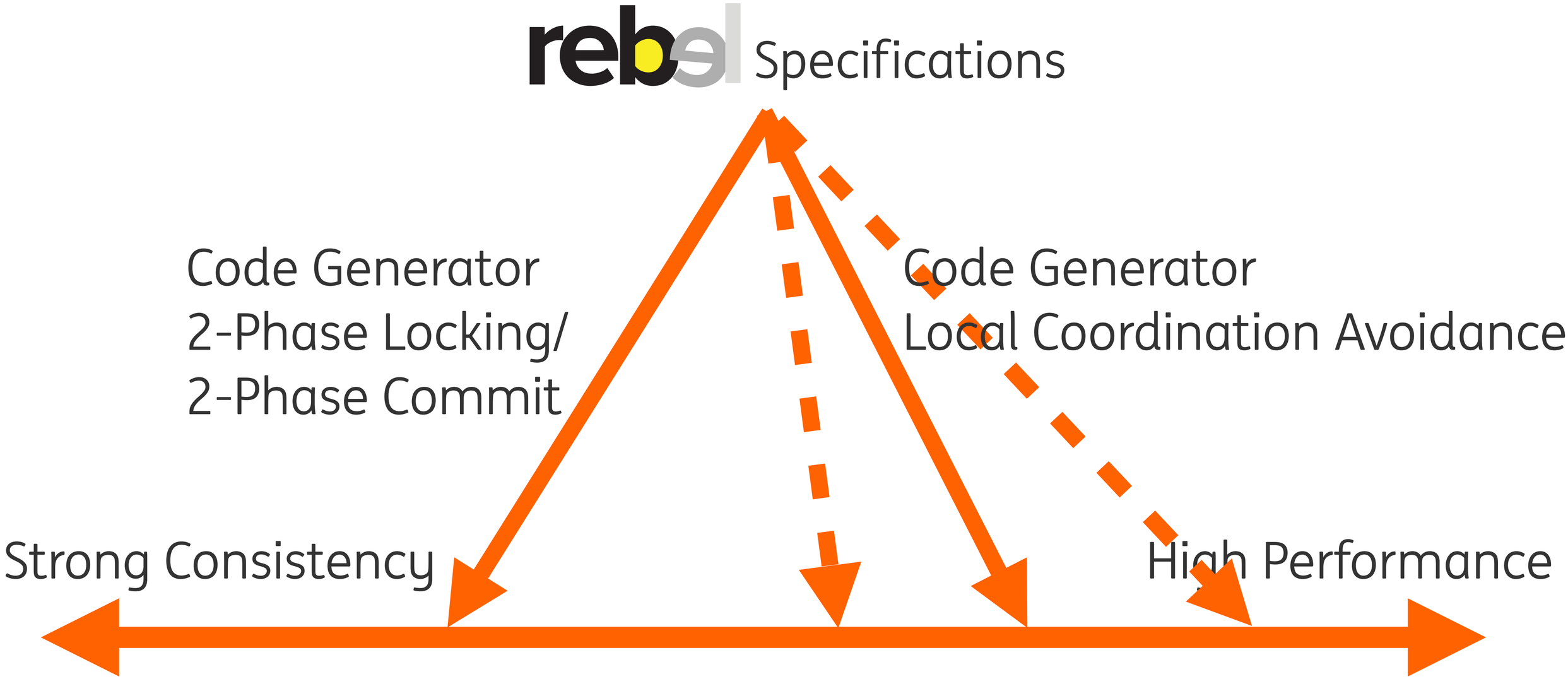
Context: Different Implementations with different guarantees



Context: Different Generators with different guarantees



Context: Different generators for Rebel with different guarantees



Goal

Determine Isolation guarantees
of different implementation strategies

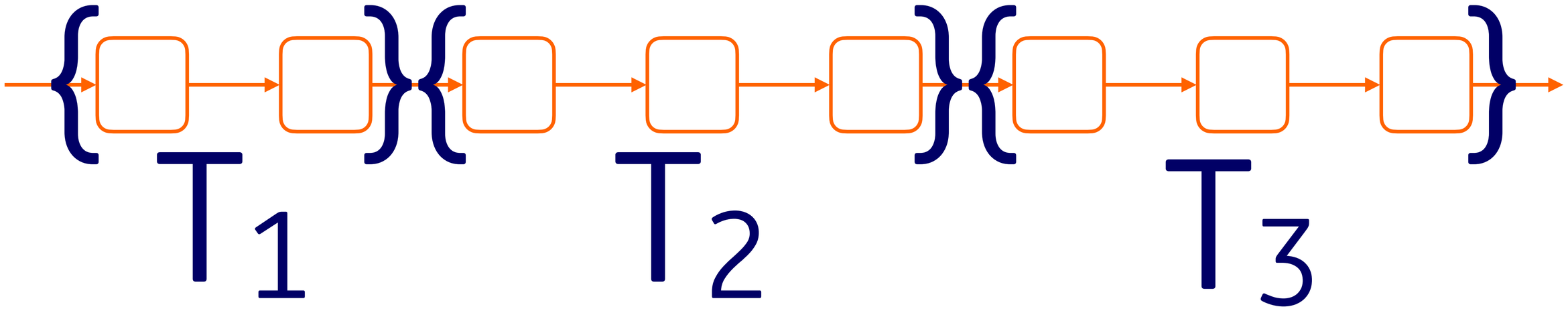
Approach

- model checker (TLC)
- isolation model
(Crooks' Isolation)

automatically
determine Isolation
of implementation

Transaction Isolation

Serializability: Order of operations in transactions occur in serial order



Snapshot Isolation: Reads from consistent snapshot of database & no conflicting writes

Crooks' Isolation (CI): State-Based Client-Centric Isolation Model

Database state

reads and writes observed by client

Conforms to model?

- Declarative logical view on Isolation guarantees
- Commit Tests for Isolation levels based on states and observed operations
- Defines if observed transactions could have been executed under Isolation level

Natacha Crooks, Youer Pu, Lorenzo Alvisi, Allen Clement: Seeing is Believing: A Client-Centric Specification of Database Isolation. PODC 2017

Transaction Example

- 2 bank accounts: Current & Saving with €30 balance
- Invariant: Sum of balances ⩾ €0
- 2 concurrent transactions:
  - Alice withdraws €40 from Current account
  - Bob withdraws €40 from Savings account
- Note: 1 transaction is allowed without violating invariant

Observed Transactions

- Operation: reads and writes of keys and values
- Transaction: sequence of Operations
- State: mapping from all keys to value
- Execution: States + Transactions

$$T_{alice} = \langle r(C,30), r(S,30), w(C, \text{-}10) \rangle$$

$$T_{bob} = \langle r(C, \text{-}10), r(S,30), abort \rangle$$

$$S_1 \left\{ \begin{array}{l} C \mapsto 30 \\ S \mapsto 30 \end{array} \right\}$$

CI Serializability

Does an execution exist where all transactions read from their direct parent state?

$$T_{alice} = \langle r(C,30), r(S,30), w(C, -10) \rangle$$
$$T_{bob} = \langle r(C, -10), r(S,30), abort \rangle$$

$$\langle T_{alice}, T_{bob} \rangle : \left\{ \begin{matrix} C \mapsto 30 \\ S \mapsto 30 \end{matrix} \right\}^{S_1} \xrightarrow{T_{alice}} \left\{ \begin{matrix} C \mapsto -10 \\ S \mapsto 30 \end{matrix} \right\}^{S_2} \xrightarrow{T_{bob}} \left\{ \begin{matrix} C \mapsto -10 \\ S \mapsto 30 \end{matrix} \right\}^{S_3}$$

$$\langle T_{bob}, T_{alice} \rangle : \left\{ \begin{matrix} C \mapsto 30 \\ S \mapsto 30 \end{matrix} \right\}^{S_1} \xrightarrow{T_{bob}\ ?} \left\{ \begin{matrix} C \mapsto 30 \\ S \mapsto -10 \end{matrix} \right\}^{S_2} \xrightarrow{T_{alice}} \left\{ \begin{matrix} C \mapsto 30 \\ S \mapsto -10 \end{matrix} \right\}^{S_3}$$

CI Serializability

Does an execution exist where all transactions read from their direct parent state?

$$T_{alice} = \langle r(C,30), r(S,30), w(C, -10) \rangle$$
$$T_{bob} = \langle r(C, -10), r(S,30), abort \rangle$$

$$\langle T_{alice}, T_{bob} \rangle : \left\{ \begin{array}{l} C \mapsto 30 \\ S \mapsto 30 \end{array} \right\}^{S_1} \xrightarrow{T_{alice}} \left\{ \begin{array}{l} C \mapsto -10 \\ S \mapsto 30 \end{array} \right\}^{S_2} \xrightarrow{T_{bob}} \left\{ \begin{array}{l} C \mapsto -10 \\ S \mapsto 30 \end{array} \right\}^{S_3}$$

Single valid execution is enough:
observed transactions are valid in a serial schedule

TLA+ and PlusCal

- TLA+: Action-based modeling of programs
  - States and transitions
  - Invariants on states — checkable by model checker TLC
- PlusCal: Models concurrent and distributed algorithms
  - Compiles to TLA+
  - (Interleaving) Processes and actions in “imperative style”
  - TLA+ invariants on states

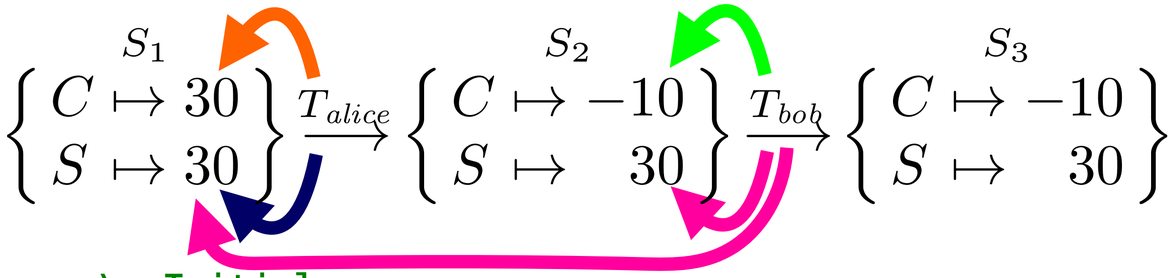
Formalization of CI in TLA+

- Formalization Process
  - Relatively straight forward, except choosing correct TLA+ base abstractions for CI, influencing whole formalization
  - Trade-off: Traceability vs Closeness to model
  - Improved definitions for incremental model checking with empty transactions
- Limitations
  - State space explosion — model check time increases rapidly
  - Small scope hypothesis — bugs have small counter examples

CI in TLA+

- Isolation levels as TLA+ properties: `Serializability(initialState, setOfTransactions)`
- Check with TLC
  - “Unit test”: Model check single `initialState` and set of `transactions` — possibly from runtime trace
  - Algorithm check: Check property on each model step

“Unit test” Example

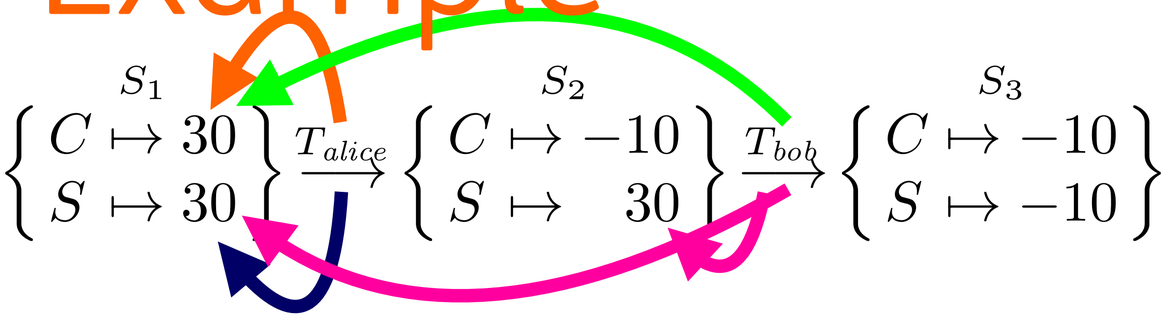



```
\* Initial
   ↪  state of Current and Savings accounts.
bankInit == (C :>  30)  (S :> 30)

talice == << r(S,30), r(C, 30), w(C,-10) >>
tbob   == << r(S,30), r(C,-10)
              (* w(S,-10) does not happen *) >>
bankTrx == {talice, tbob}

ASSUME Serializability(bankInit, bankTrx)
ASSUME SnapshotIsolation(bankInit, bankTrx)
ASSUME ReadCommitted(bankInit, bankTrx)
ASSUME ReadUncommitted(bankInit, bankTrx)
```

Serializability




```
tbAlice == << r(S, 30), r(C,30), w(C,-10) >>
tbBob   == << r(S, 30), r(C,30), w(S,-10) >>
bBankTrx == {tbAlice, tbBob}

ASSUME Serializability
      ↪ (bankInit, bBankTrx) == FALSE
ASSUME SnapshotIsolation(bankInit, bBankTrx)
ASSUME ReadCommitted(bankInit, bBankTrx)
ASSUME ReadUncommitted(bankInit, bBankTrx)
```

Snapshot Isolation

Checking Isolation of Algorithms

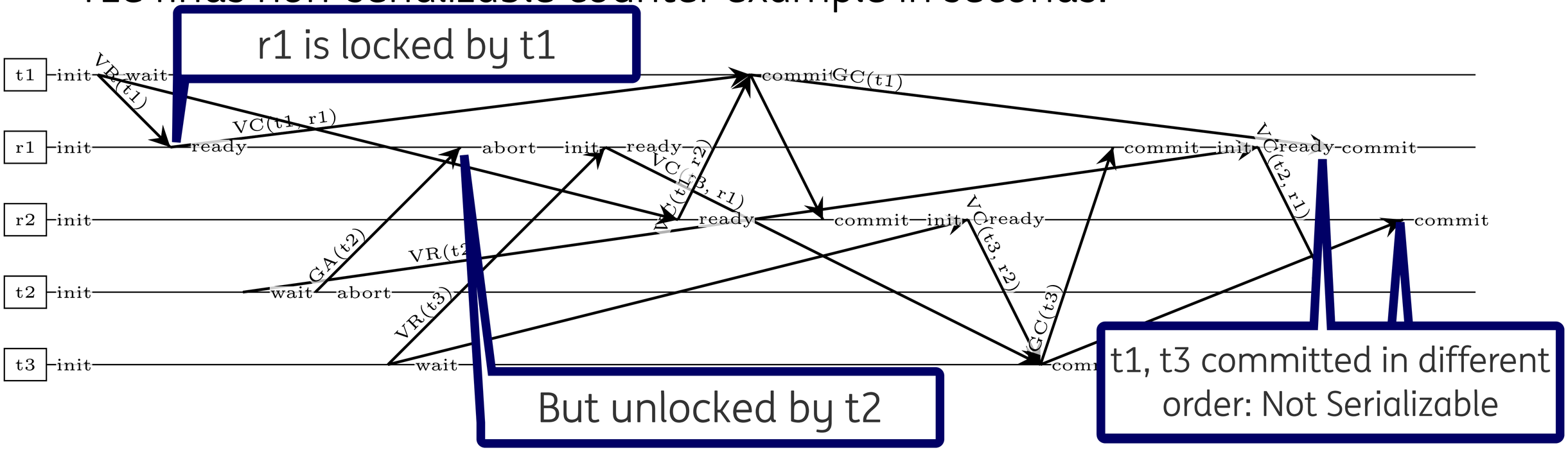
- Model Algorithm in PlusCal or TLA+
- Check isolation properties in each state of model: `Serializability(InitialState, ccTransactions)`
- Track `ccTransactions` in algorithm steps

2-Phase Locking & 2 Phase Commit

- 2PL: Serializable Isolation by locking data resources
- 2PC: Atomicity by resources first voting, then only committing if all agree
- Modeled in PlusCal by 2 processes with shared-channel message passing: Transaction Manager and Resource

Bug Seeding

- Can this approach actually find Isolation violations?
- Specification bug where Transaction Resource is locked by a transaction, but unlocks when another aborts
- TLC finds non-serializable counter example in seconds:



Current Work

- Reads/Writes do not capture Serializability of semantically higher-level operations, such as used in Rebel
  - e.g. 2 interleaving deposits are Serializable
- Improve formalization to leverage higher-level operations: Multi-level transactions with commutativity

Validating Isolation with TLA+

Tim Soethout (tim.soethout@ing.com), Tijs van der Storm, Jurgen Vinju

- Trade-off between Performance and Isolation
- Need to qualify isolation level of different implementations
- Model Checking can check conformance of run-time traces and specifications of algorithms
- Enables rapid experimentation with efficient synchronization strategies, without overlooking isolation

CWI
Centrum Wiskunde & Informatica

ING