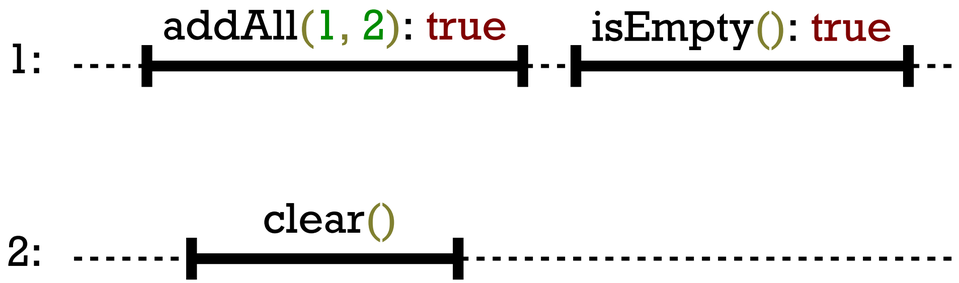
Checking Linearizability Using Hitting Families

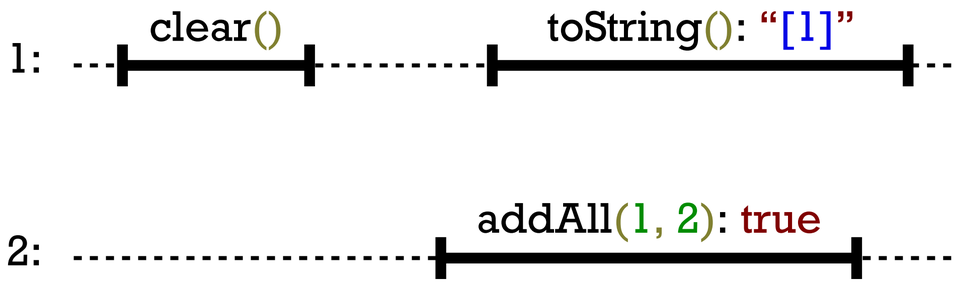
Burcu Kulahcioglu Ozkan[1], Rupak Majumdar[1], **Filip Niksic**[2]

[1] Max Planck Institute for Software Systems (MPI-SWS)
[2] University of Pennsylvania

Linearizability as a correctness condition

Two execution histories on a **concurrent list:**



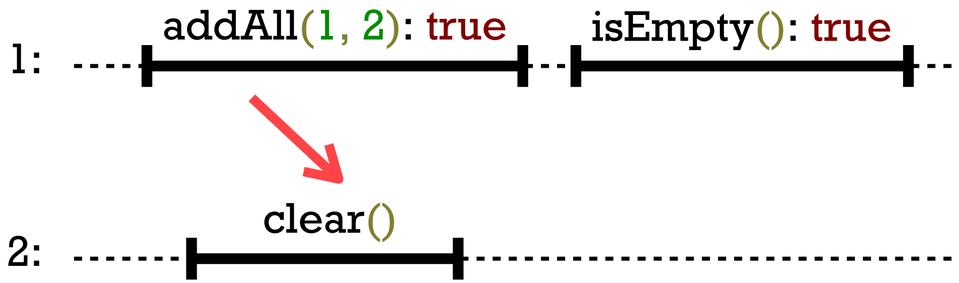


Linearizable history: Concurrent operations can be totally ordered in a consistent way

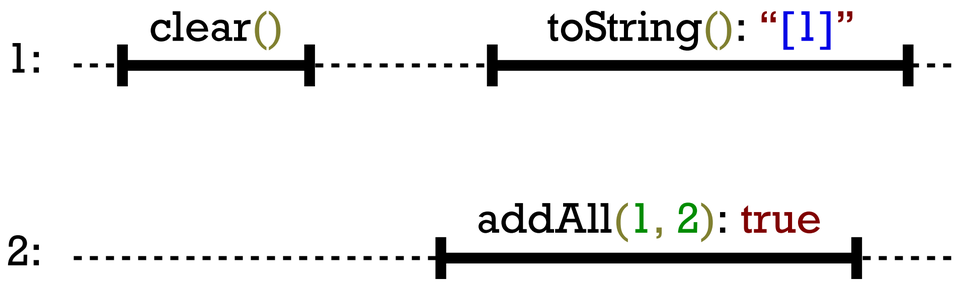
Linearizable concurrent object: All of its execution histories are linearizable

Linearizability as a correctness condition

Two execution histories on a **concurrent list:**



linearizable

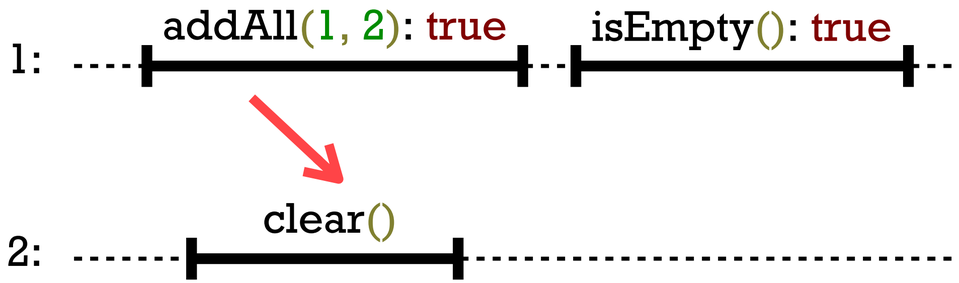


Linearizable history: Concurrent operations can be totally ordered in a consistent way

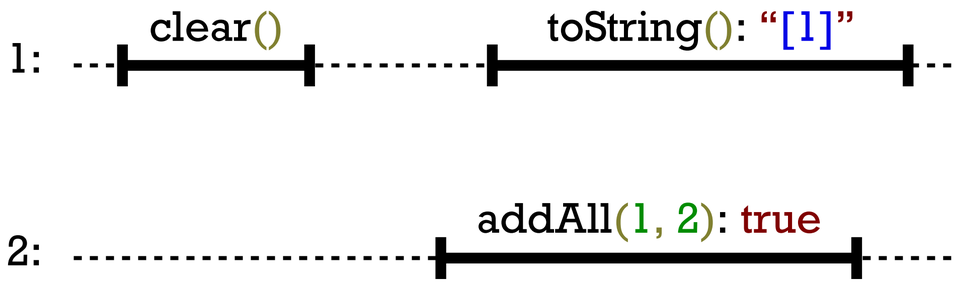
Linearizable concurrent object: All of its execution histories are linearizable

Linearizability as a correctness condition

Two execution histories on a **concurrent list:**



linearizable



non-linearizable

Linearizable history: Concurrent operations can be totally ordered in a consistent way

Linearizable concurrent object: All of its execution histories are linearizable

Checking linearizability

Linearizability of a concurrent object

- Pertains to **verification**
- **Undecidable**
- Approaches: Program logics, proof rules, semi-automated procedures

Linearizability of a single execution history

- Pertains to **testing**
- **NP-complete**
- Approaches: Exhaustive search for a linearizability witness with space-time trade-offs

Checking linearizability

Linearizability of a concurrent object

- Pertains to **verification**
- **Undecidable**
- Approaches: Program logics, proof rules, semi-automated procedures

Linearizability of a single execution history

- Pertains to **testing**
- **NP-complete**
- Approaches: Exhaustive search for a linearizability witness with space-time trade-offs

Our contribution

In a nutshell:

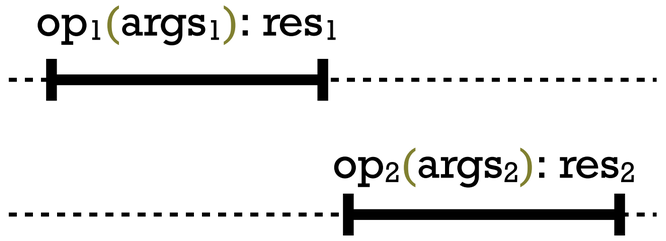
- Prioritize the search space to quickly find linearizability witnesses (if they exist)

In more detail:

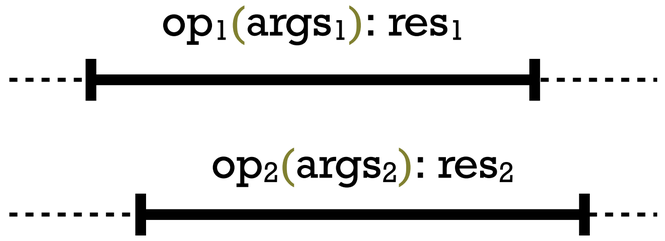
- Introduce **linearizability depth** of a history
- For a history of linearizability depth **d**, with **n** operations on **k** threads:
  Suffices to explore a **strong d-hitting family** of at most $\mathbf{O(kn^{d-1})}$ linearizations
- Experiments on **java.util.concurrent**:
  In most cases linearizability witnessed by strong **d**-hitting families with $\mathbf{d \leq 5}$!

Linearizability

Execution history induces a partial order of operations



op_1 happens before op_2



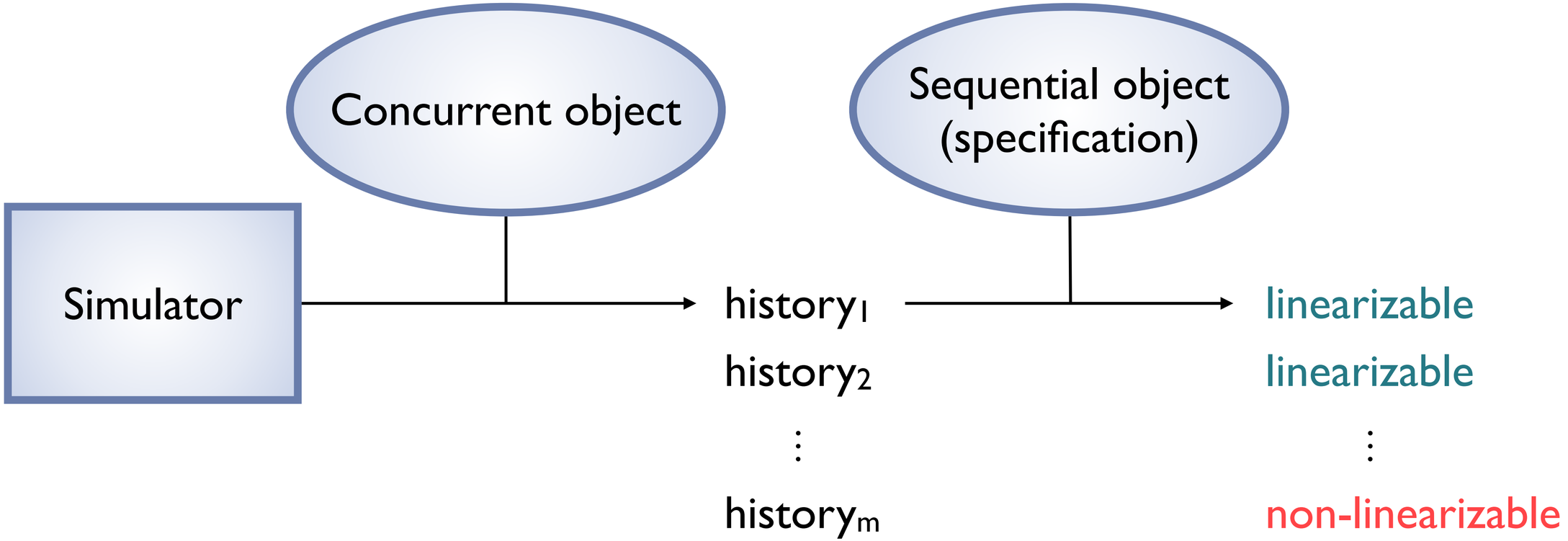
op_1 and op_2 are concurrent

Schedule: A total order of operations that extends the happens-before relation

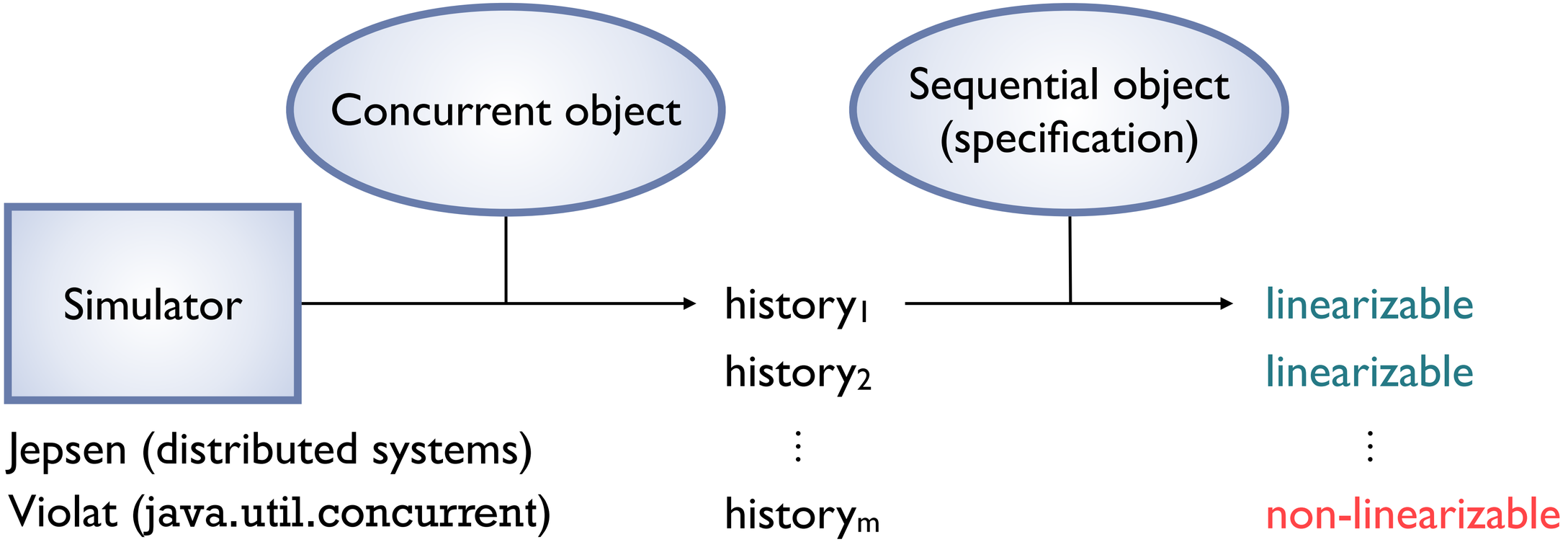
Linearizability witness: A schedule in which the results of operations satisfy the sequential specification

Linearizable history: A history that has a linearizability witness

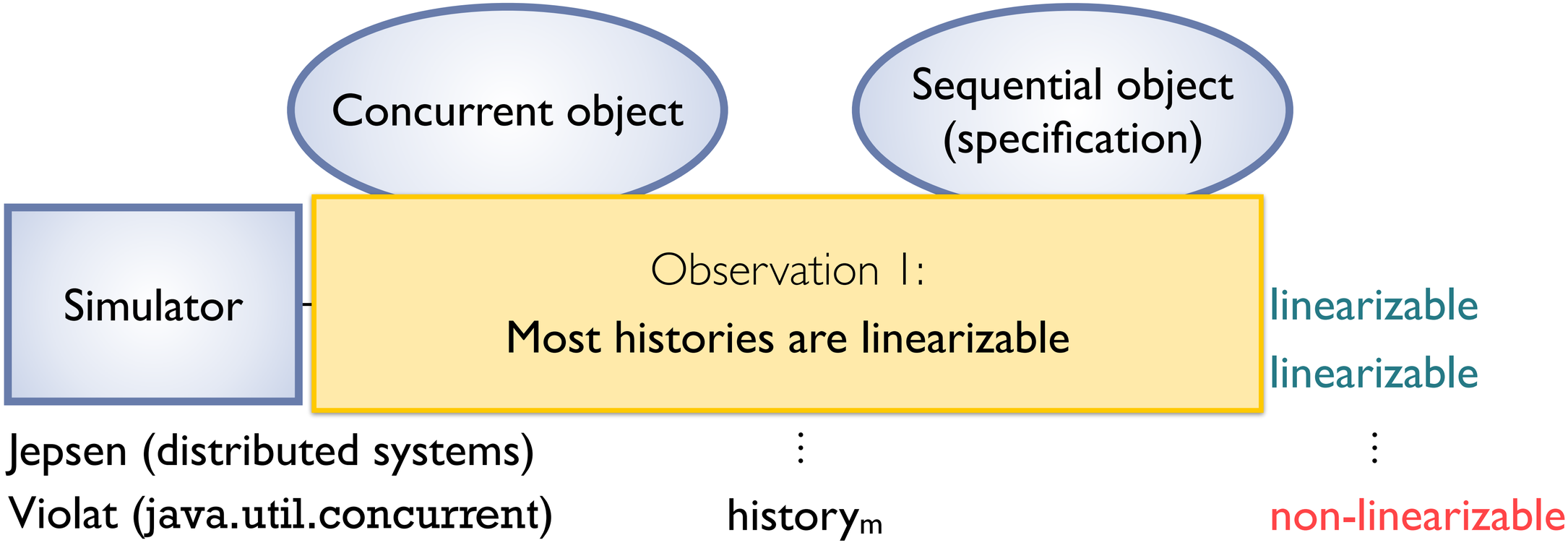
Testing concurrent objects [Wing and Gong '93]



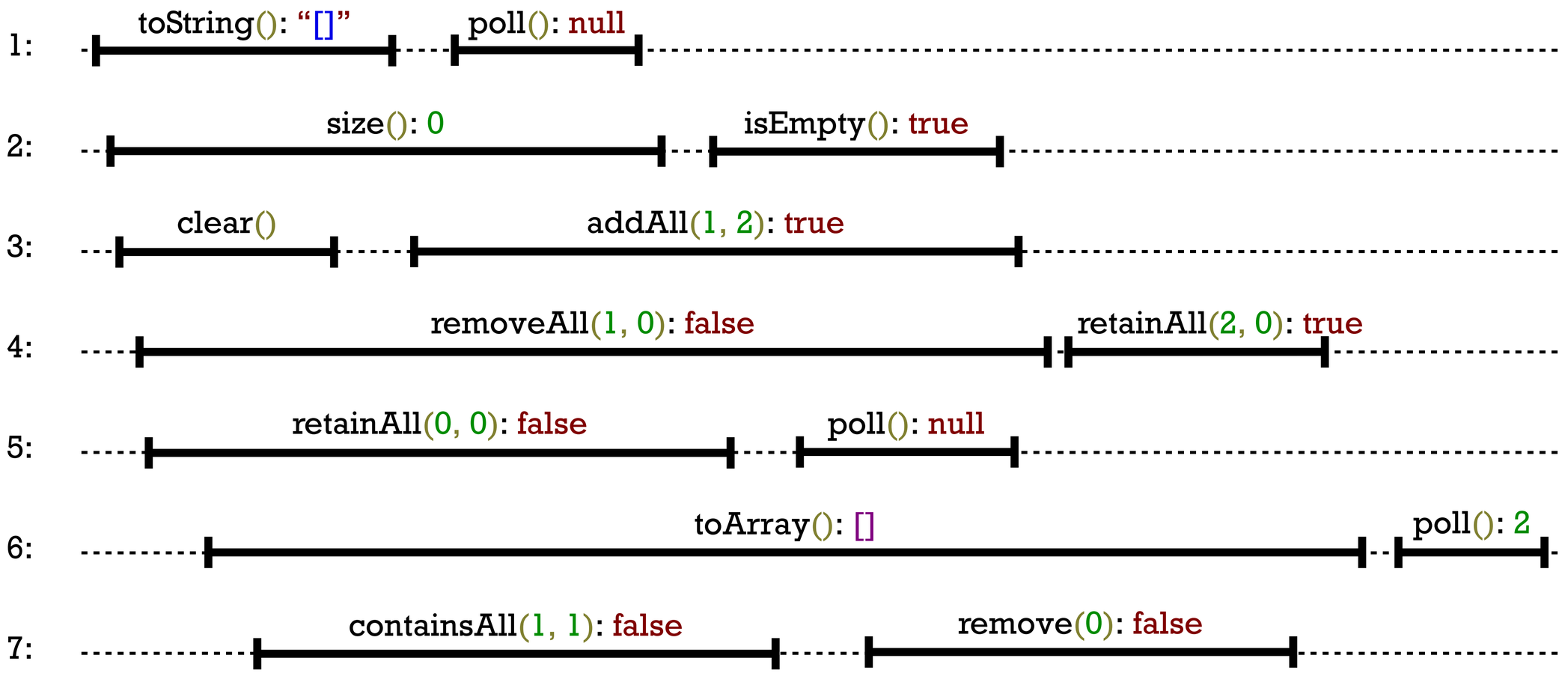
Testing concurrent objects [Wing and Gong '93]



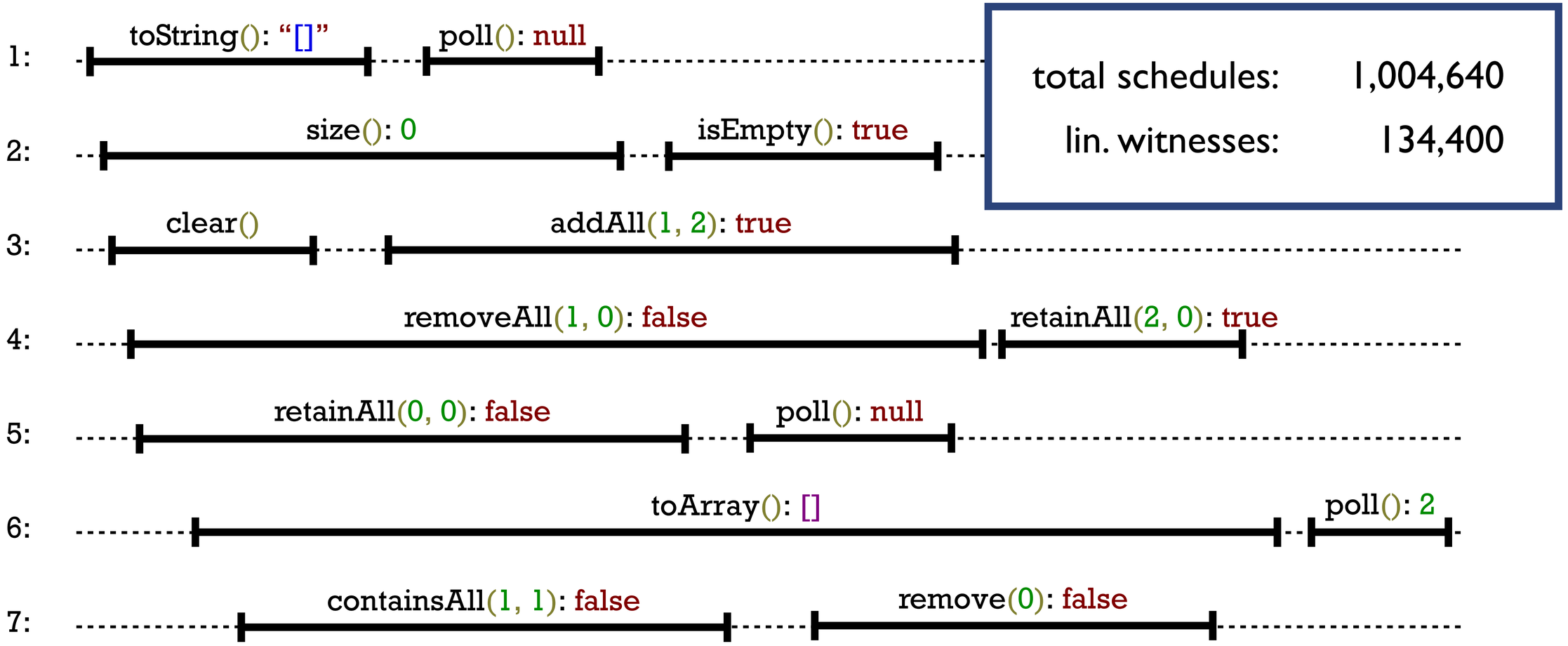
Testing concurrent objects [Wing and Gong '93]



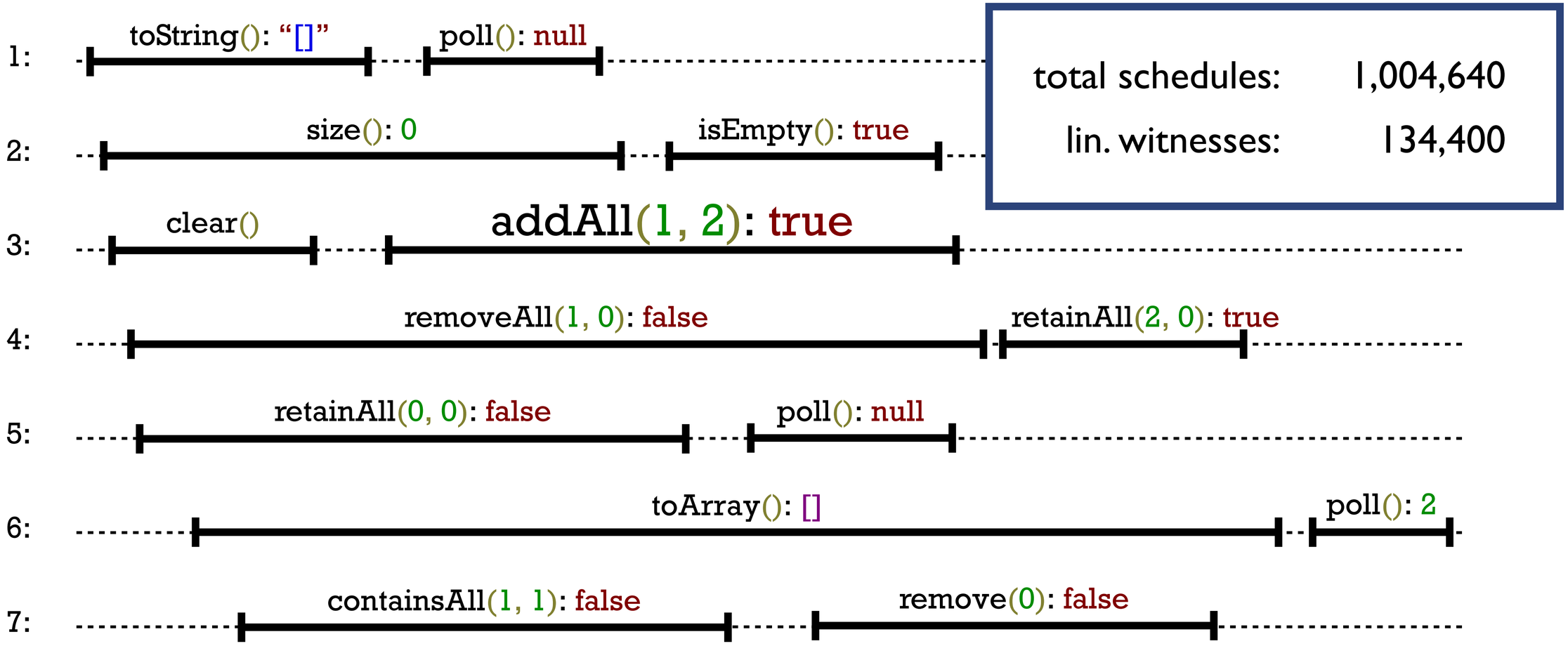
Execution history generated by Violat for ConcurrentLinkedQueue



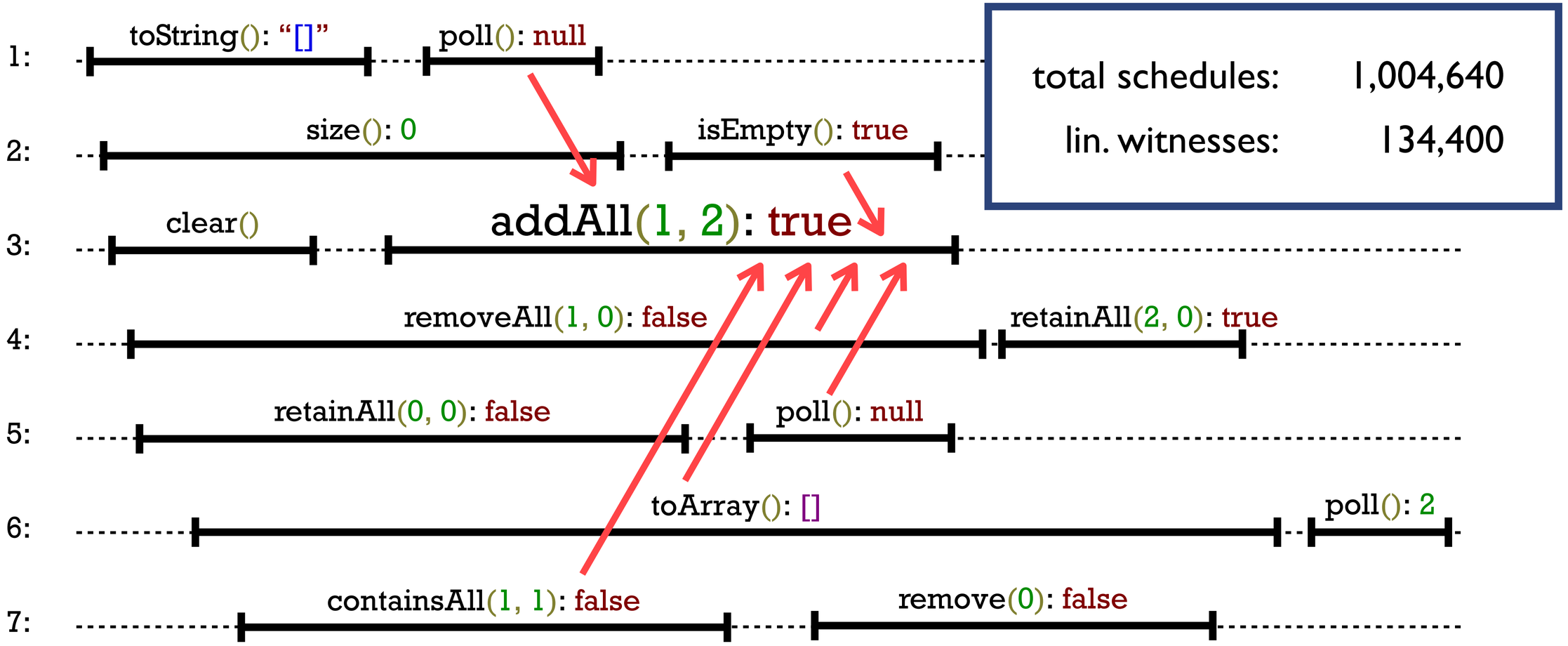
Execution history generated by Violat for ConcurrentLinkedQueue



Execution history generated by Violat for ConcurrentLinkedQueue



Execution history generated by Violat for ConcurrentLinkedQueue



Linearizability depth

Strong hitting:

Given $\mathbf{d \geq 1}$, a schedule $\boldsymbol{\alpha}$ **strongly hits** a **d**-tuple of operations $\mathbf{(op_0, \ldots, op_{d-1})}$ if it

- maximally delays each $\mathbf{op_i}$
- while maintaining the relative order $\mathbf{op_0 <_\alpha \ldots <_\alpha op_{d-1}}$

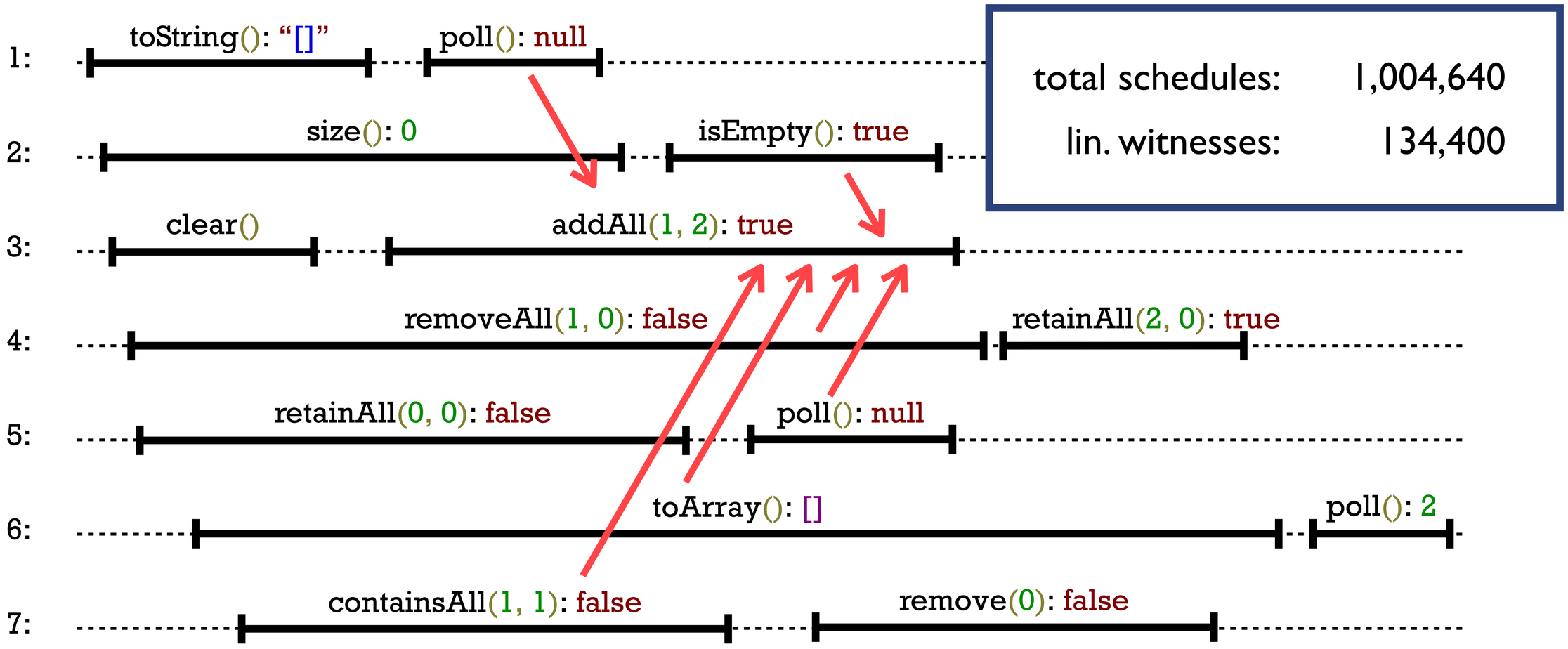
d-Linearizable history:

There exist operations $\mathbf{op_0, \ldots, op_{d-1}}$ such that every schedule that strongly hits $\mathbf{(op_0, \ldots, op_{d-1})}$ is a witness to linearizability

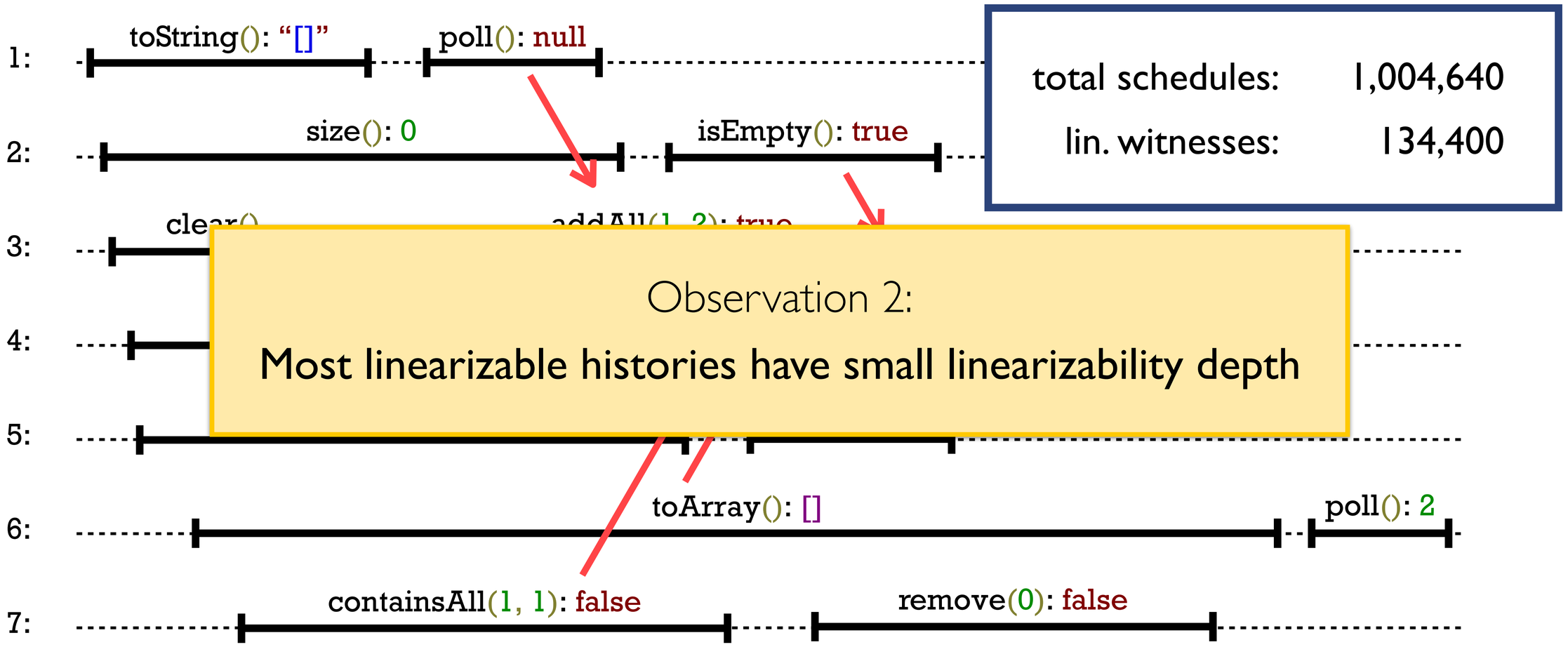
Linearizability depth of a history:

Smallest $\mathbf{d \geq 1}$ such that the history is **d**-linearizable

History from the example has linearizability depth 1



History from the example has linearizability depth 1



Checking d-linearizability

Recall: A history is **d**-linearizable if there exist operations $\mathbf{op_0}, \ldots, \mathbf{op_{d-1}}$ such that every schedule that strongly hits $\mathbf{(op_0}, \ldots, \mathbf{op_{d-1})}$ is a witness to linearizability

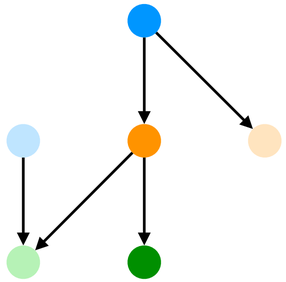
Strong d-hitting family: A set of schedules $\mathcal{F}$ is a **strong d-hitting family** if it strongly hits every **d**-tuple of operations

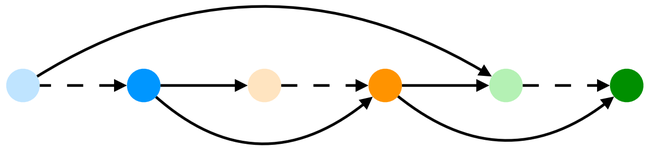
Conclusion: To check **d**-linearizability, it suffices to explore schedules from a strong **d**-hitting family

Size of strong hitting families [OOPSLA '18]

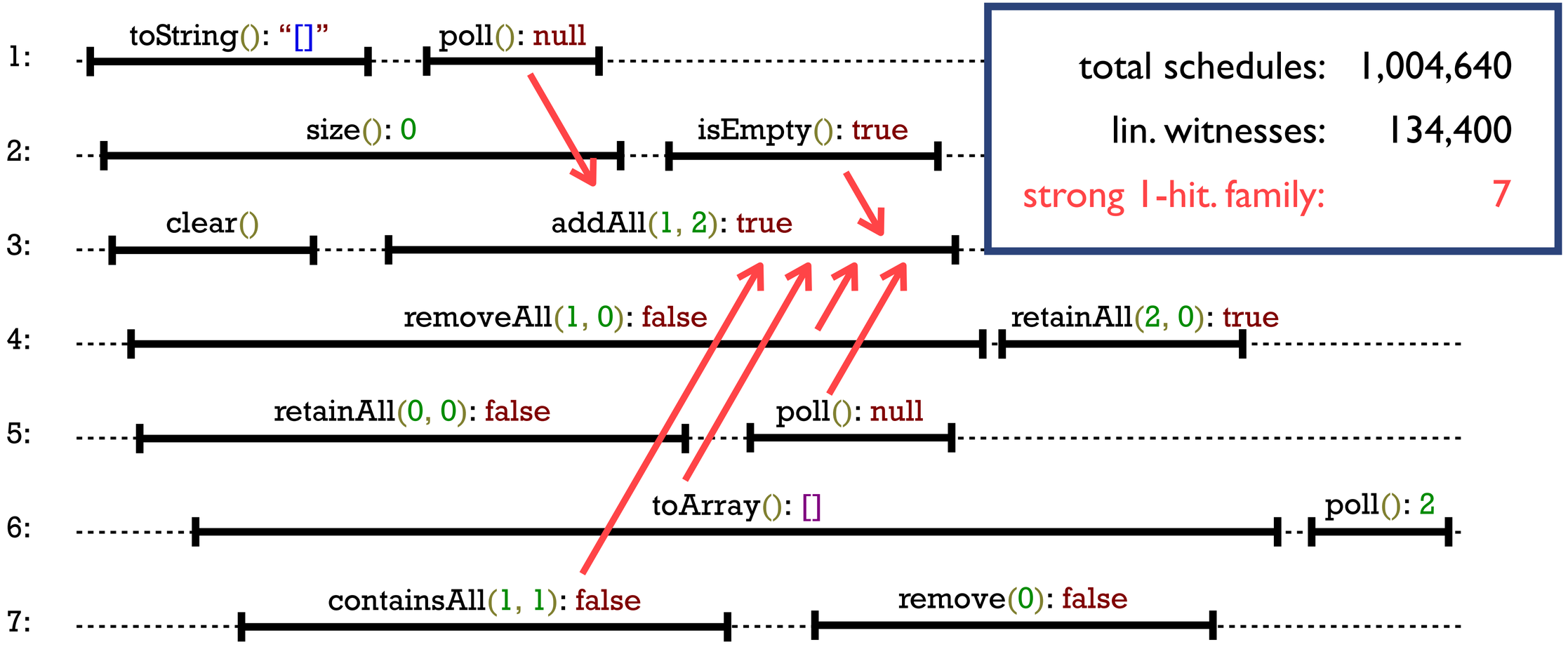
Given a history of **n** operations on **k** threads, and **d ≥ 1**:

- There is a strong **d**-hitting family of size **$O(n^d)$**
- There is a strong **d**-hitting family of size **$O(kn^{d-1})$**:
  A single schedule can strongly hit all operations in a thread





For the history from the example, 1-linearizability is shown by exploring 7 schedules!



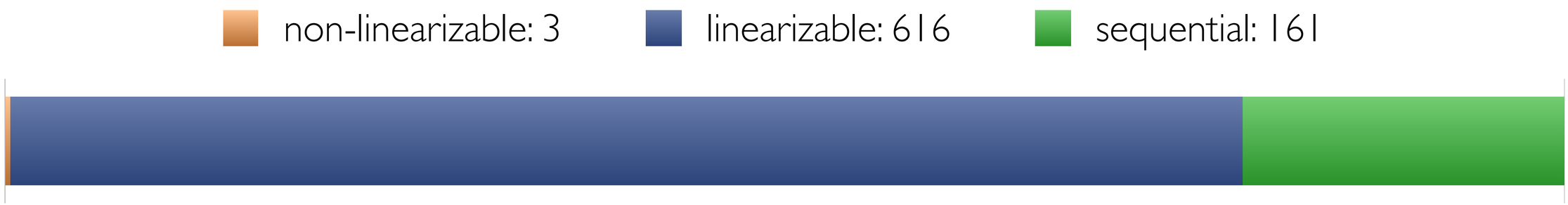
Experiments

Observation 1: Most execution histories are linearizable

Observation 2: Most linearizable histories have small linearizability depth

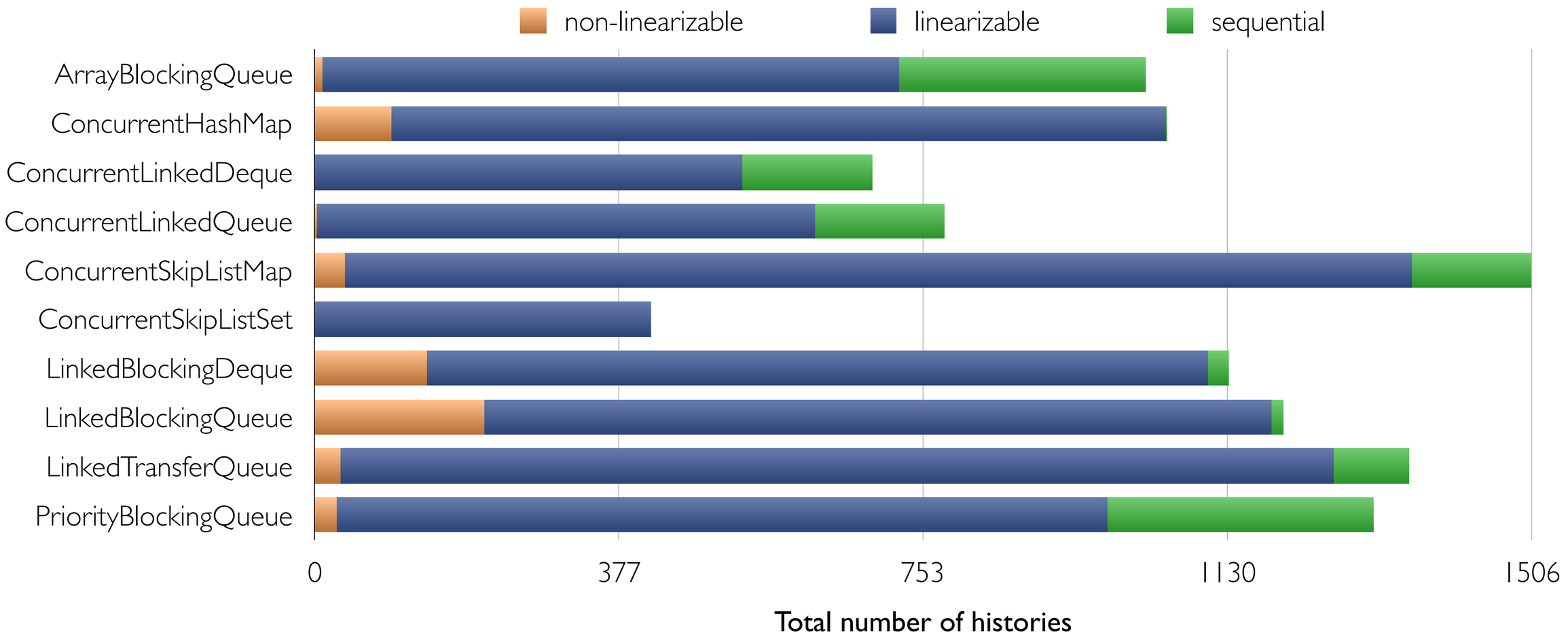
Goal:
Validate the observations for the histories generated by Violat on **java.util.concurrent**

Breakdown of histories generated by Violat for ConcurrentLinkedQueue

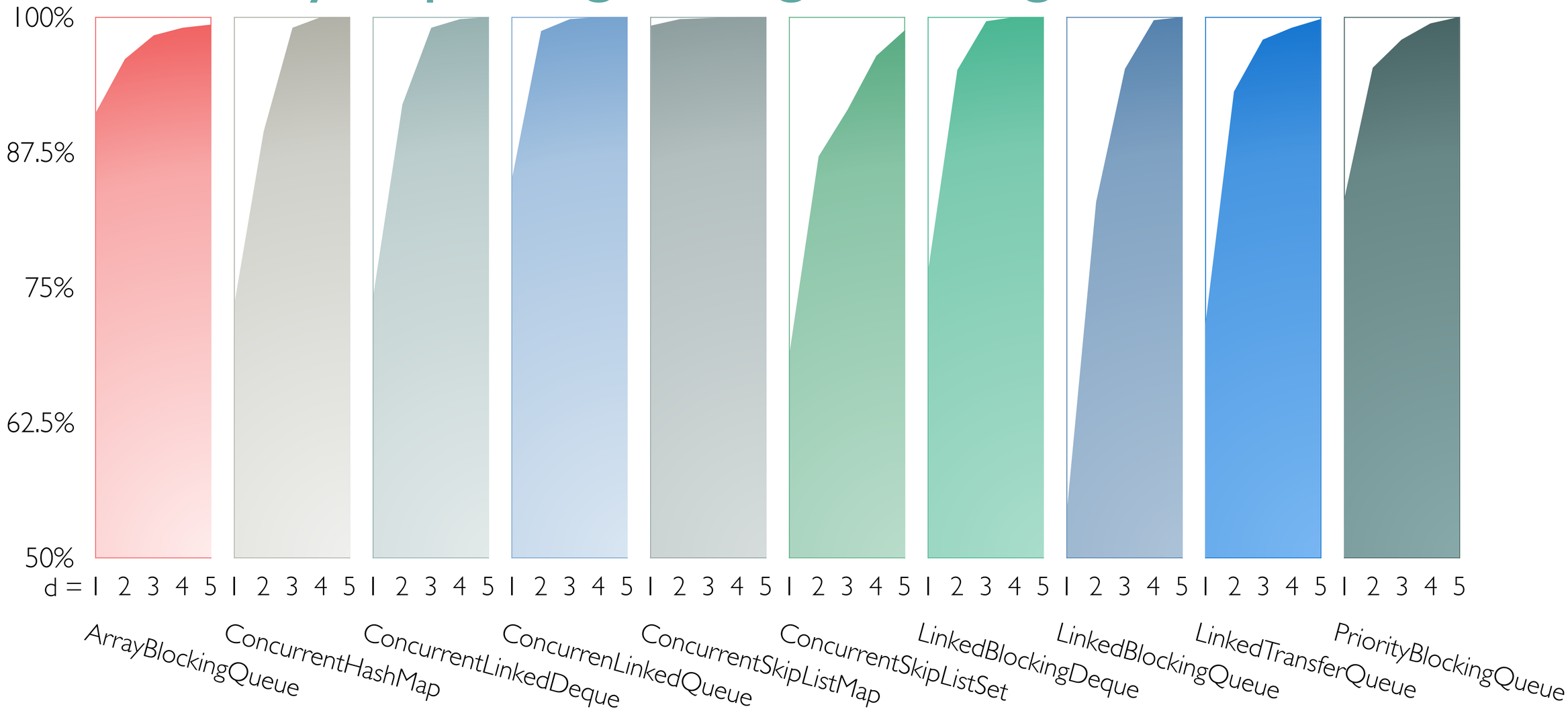


Total: 780 histories

Breakdown of histories generated by Violat



Linearizable histories whose linearizability is shown by exploring strong d-hitting families



Conclusion

Contributions

- Introduce **linearizability depth** of a history
- For a history of linearizability depth **d**, with **n** operations on **k** threads:
  Suffices to explore a **strong d-hitting family** of at most $\mathbf{O(kn^{d-1})}$ linearizations
- Experiments on **java.util.concurrent**:

  In most cases linearizability witnessed by strong **d**-hitting families with $\mathbf{d \leq 5}$!

Future work

- Optimized linearizability checker for **Jepsen**; experiments on distributed systems
- Weaker correctness conditions like causal consistency