How to specify and prove the correctness of concurrent objects (libraries)?

**Linearizability**

- Standard correctness criterion for concurrent objects $O$
- $O \leq_{\text{lin}} S$: All concurrent executions of $O$ are “equivalent” to some sequential executions of abstract object $S$

**Our Approach to Verifying $O \leq_{\text{lin}} S$**

- $C$: instrument $O$ with auxiliary commands (ACs) at LPs
- ACs manipulate auxiliary states ($S$ and abstract states)
- Execute $S$ simultaneously with $O$'s LP step
- Reason about $C$ using our program logic
- Extend an existing logic (e.g., Rely-Guarantee) with inference rules for ACs
- Ensure $O$'s LP is the single step with the same effect as $S$

**Linself for Fixed LPs**

**Challenges in Verification**

1. **Non-Fixed LPs**
   - Helping mechanism
   - LP is in other threads’ code
   - Lose thread-modularity?
2. **Future-dependent (FD) LPs**
   - LP is at prior access, but only if later validation succeeds
   - Refer to unpredictable future behaviors?

**Pending Thread Pool for Helping**

- HSY Elimination-Backoff Stack
- $O \leq_{\text{lin}} S$: All concurrent executions of $O$ are “equivalent” to some sequential executions of abstract object $S$
- $C$: instrument $O$ with auxiliary commands (ACs) at LPs
- ACs manipulate auxiliary states ($S$ and abstract states)
- Execute $S$ simultaneously with $O$'s LP step
- Reason about $C$ using our program logic
- Extend an existing logic (e.g., Rely-Guarantee) with inference rules for ACs
- Ensure $O$'s LP is the single step with the same effect as $S$

**Try-Commit for Future-Dependent LPs**

- Speculate (trylin) at potential LP, keep both result and original abstract code & abstract states $\Rightarrow$ speculation set
- Commit to correct branch at later validation and discard others

**Verify Using Our Logic**

- A program logic for linearizability
- Support non-fixed LPs
- A light instrumentation mechanism to help verification
- Try-commit clause as an alternative to prophecy variables
- Logic ensures contextual refinement $\Rightarrow$ linearizability
- A new forward-backward simulation as meta-theory
- Verified 12 well-known algorithms
- Some are used in Java.util.concurrent (JUC)