A Program Logic for Obstruction-freedom

Zhao-Hui LI, Xin-Yu FENG

Frontiers of Computer Science, DOI: 10.1007/s11704-023-2774-9

Problems & Ideas

- Reasoning about and verifying obstruction-freedom is a big challenge
  - Obstruction-free implementations are practical for software transactional memory(STM) and in anonymous and fault-tolerant distributed computing.
  - Existing work can only verify obstruction-freedom of specific data structures.
  - Following traditional idea (i.e., treat obstruction-freedom as a liveness property and verify it by proving “something good will eventually happen”) encounters thorny problems.

- Our idea: verify obstruction-freedom by proving “some bad thing never happens”
  - Inspired by the method of verifying safety properties.
  - The bad things should be prevented: *divergences caused by finite interference*.
  - The implementation in the right figure is not obstruction-free because the “bad thing” occurs.

```
f() {
1 i:=0;
2 while (i=1) skip;
3 return
 }
g() {
4 i:=i+1;
5 return
 }
client:
6 f() || g()
```

Main Contributions

- A program logic that can formally verify obstruction-freedom

$$\frac{R \vdash \{p \wedge B\}\ C\ \{p\}}{R \vdash \{p\}\ \mathbf{while}(B)\{C\}\ \{p \wedge \neg B\}}$$

(Standard WHILE rule)

prevent *divergences caused by finite interference*

adding a sequential total-correctness judgment

$$\frac{R \vdash \{p \wedge B\}\ C\ \{p\} \quad \vdash [p \wedge B]\ \mathbf{while}(B)\{C\}\ [p \wedge \neg B]}{R \vdash \{p\}\ \mathbf{while}(B)\{C\}\ \{p \wedge \neg B\}}$$

(New WHILE rule)

- Informal principles to extend a logic for verifying linearizability to verifying obstruction-freedom
  - can reuse the existing proof for linearizability

- A case study: have verified a practical obstruction-free double-ended queue
  - The implementation is from a classic paper, where the concept of obstruction-freedom was proposed for the first time.
  - We prove the implementation satisfies both obstruction-freedom and linearizability.