Ultimate GemCutter: Commutativity in Concurrent Program Verification

Dominik Klumpp, Daniel Dietsch, Matthias Heizmann, Frank Schüssele, Azadeh Farzan, Andreas Podelski

ultimate-pa.org

github.com/ultimate-pa/ultimate

Commutativity Simplifies Proofs of Concurrent Programs

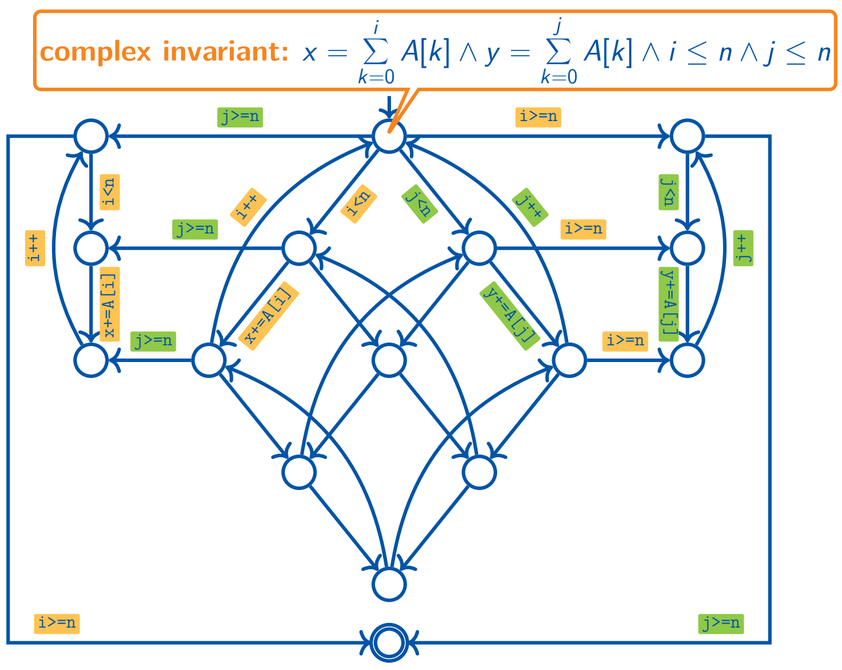
Concurrent Program

$$\{\ x = y = i = j = 0\ \}$$

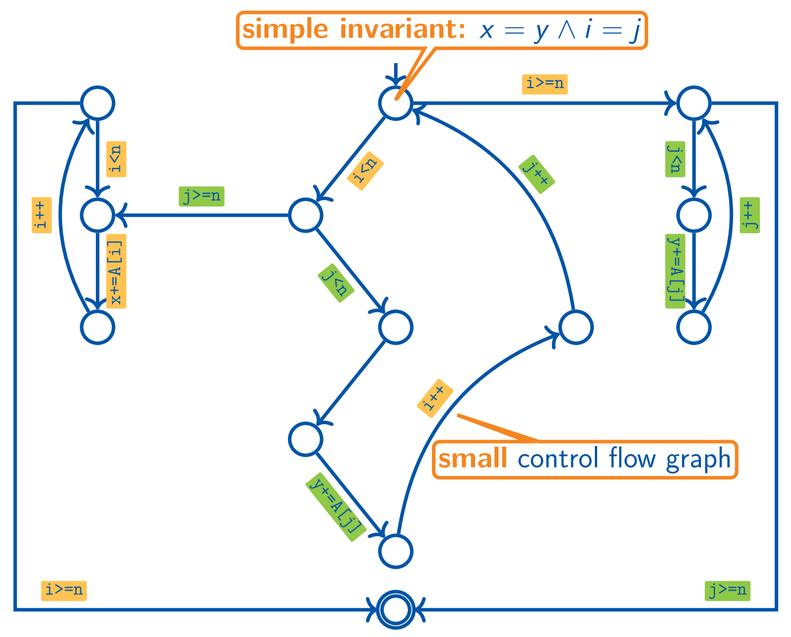
```
while (i < n) {
    x += A[i];
    i++;
}
```
‖
```
while (j < n) {
    y += A[j];
    j++;
}
```

$$\{\ x = y\ \}$$

All Interleavings



A Sound Reduction



Commutativity

Many pairs of statements **commute**:

i.e., order of execution does not matter

Example: `x+=A[i]` `y+=A[j]` ∼ `y+=A[j]` `x+=A[i]`

Extension: **proof-sensitive commutativity**

Example: `*x = 0` `*y = 1` ∼ `*y = 1` `*x = 0`

if we have proven that $x \neq y$

swapping adjacent commuting statements

$\rightsquigarrow$ **equivalent** traces

Reduction

representative subset of program traces: at least one representative per equivalence class

Soundness:

one trace correct ⇒ all equivalent traces correct

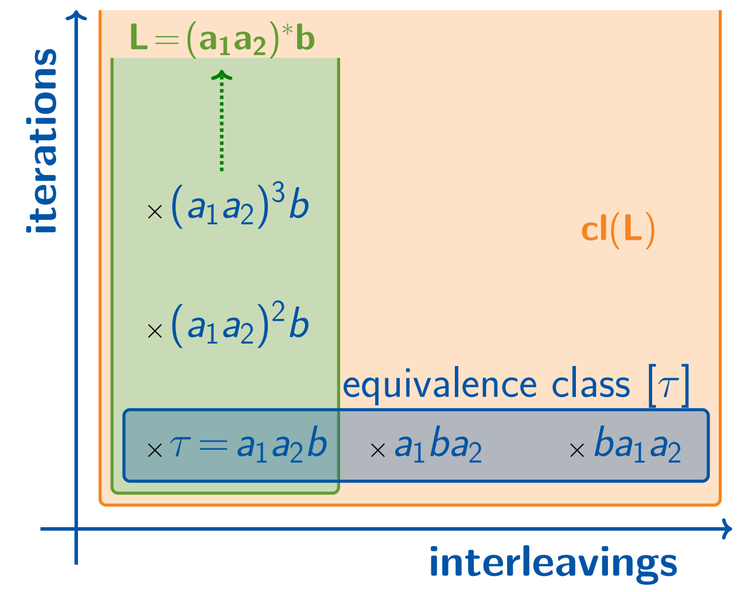
correctness of reduction ⇒ correctness of program

Verification Principle

GemCutter **generalizes** from spurious counterexamples τ to larger sets of correct traces:

trace abstraction generalizes across loop iterations to a set of traces L

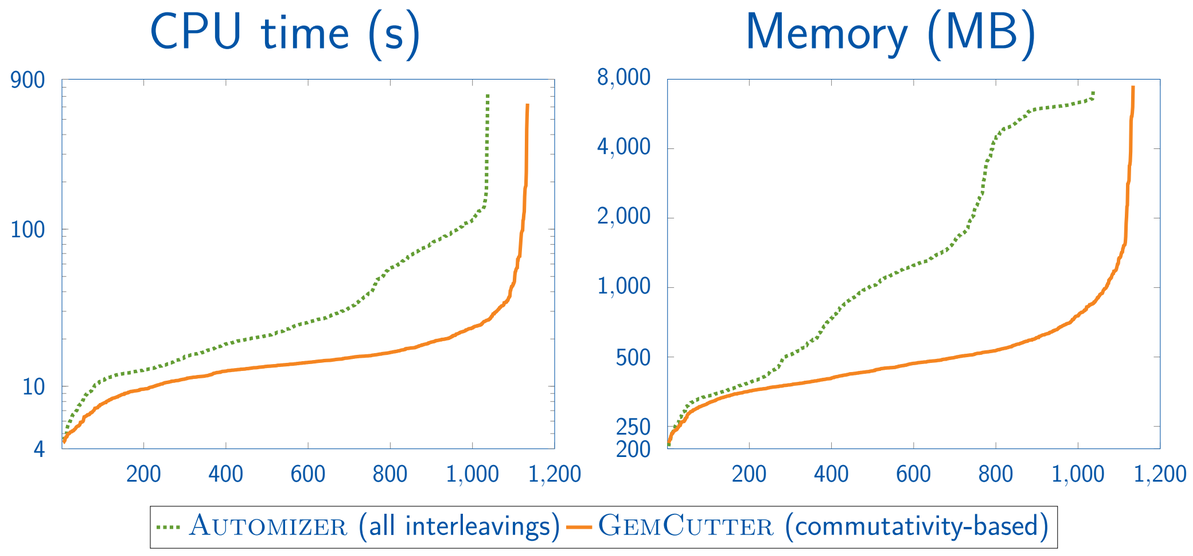
commutativity allows for generalization across interleavings to the set $cl(L)$ of all equivalent traces



If $cl(L)$ contains all program traces, the program is correct.
Equivalently: If L contains all traces *of a reduction*, then the program is correct.

Performance

Evaluation shows significant advantages over a state-of-the-art verifier (Ultimate Automizer):



Competitions:

- **SV-COMP'24:** **2nd place** in *ConcurrencySafety*
- **SV-COMP'23:** **3rd place** in *ConcurrencySafety*
- **SV-COMP'22:** **3rd place** in *ConcurrencySafety*, **1st place** in *NoDataRace (demo)*

Commutativity & Verification

choice of representatives affects proof simplicity

- **challenge:** select suitable representatives

choice of proof affects possible commutativity

- **challenge:** find useful *abstract* commutativity

partial order reduction algorithms speed up verification

- **challenge:** adapt classical POR algorithms

commutativity reasoning is widely applicable

- **challenge:** extend to more programs & properties

[SV-COMP'22] *Ultimate GemCutter and the Axes of Generalization,* Klumpp, Dietsch, Heizmann, Schüssele, Ebbinghaus, Farzan and Podelski, 2022

[PLDI'22] *Sound Sequentialization for Concurrent Program Verification,* Farzan, Klumpp and Podelski, 2022

[POPL'23] *Stratified Commutativity in Verification Algorithms for Concurrent Programs,* Farzan, Klumpp and Podelski, 2023

[POPL'24] *Commutativity Simplifies Proofs of Parameterized Programs,* Farzan, Klumpp and Podelski, 2024