Theta: Various Approaches for Concurrent Program Verification (Competition Contribution)

Csanád Telbisz, *Levente Bajczi, Dániel Szekeres, András Vörös*

Critical Systems Research Group

Original Theta Approach

Original Theta Approach

- **CEGAR**: counterexample-guided abstraction refinement
  - Different **abstract domains**, **refinement** strategies

Original Theta Approach

- **CEGAR**: counterexample-guided abstraction refinement
  - Different **abstract domains**, **refinement** strategies
- Concurrency:
  - **Abstraction-aware Partial Order Reduction** (POR)
  - **On-the-fly cone-of-influence** (COI)

Original Theta Approach

- **CEGAR**: counterexample-guided abstraction refinement
  - Different **abstract domains**, **refinement** strategies
- Concurrency:
  - **Abstraction-aware Partial Order Reduction** (POR)
  - **On-the-fly cone-of-influence** (COI)
- Interprocedural verification
  - **Stack abstraction**

Original Theta Approach

- **CEGAR**: counterexample-guided abstraction refinement
  - Different **abstract domains**, **refinement** strategies
- Concurrency:
  - **Abstraction-aware Partial Order Reduction** (POR)
  - **On-the-fly cone-of-influence** (COI)
- Interprocedural verification
  - **Stack abstraction**

See SPIN Thursday afternoon session

New approach

New approach

- Bounded Model Checking (BMC) for handling concurrency

New approach

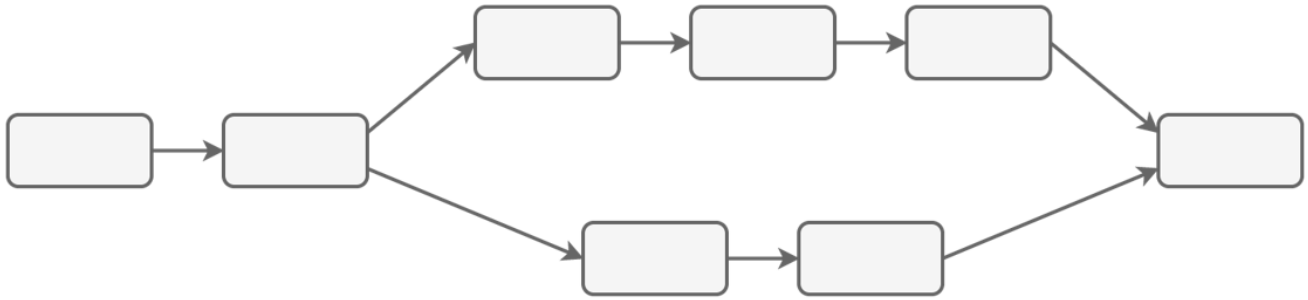
- Bounded Model Checking (BMC) for handling concurrency
- Reasoning with **partial orders** (happens-before relation)

New approach

- Bounded Model Checking (BMC) for handling concurrency
- Reasoning with **partial orders** (happens-before relation)
- Key idea:
  - Build a **happens-before relation**
    (program order + dataflow + axioms)

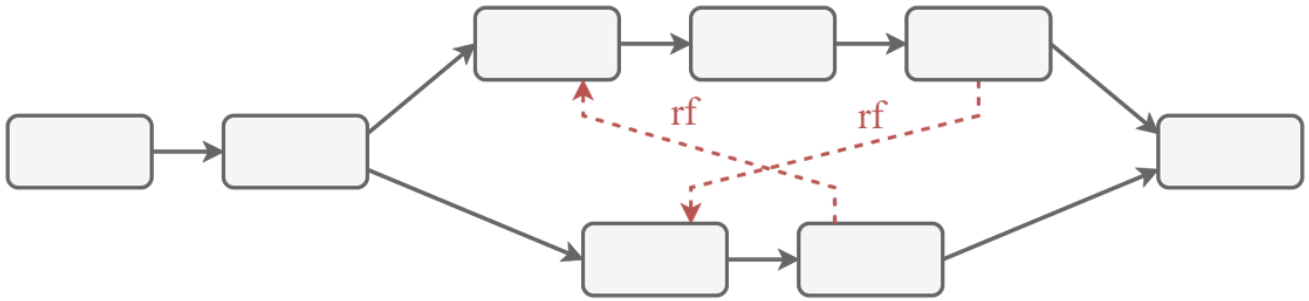
New approach

- Bounded Model Checking (BMC) for handling concurrency
- Reasoning with **partial orders** (happens-before relation)
- Key idea:
  - Build a **happens-before relation**
    (program order + dataflow + axioms)



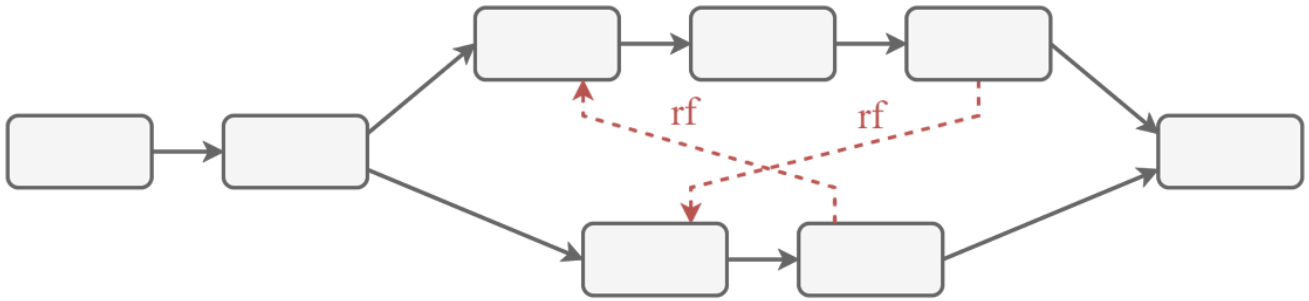
New approach

- Bounded Model Checking (BMC) for handling concurrency
- Reasoning with **partial orders** (happens-before relation)
- Key idea:
  - Build a **happens-before relation**
    (program order + dataflow + axioms)



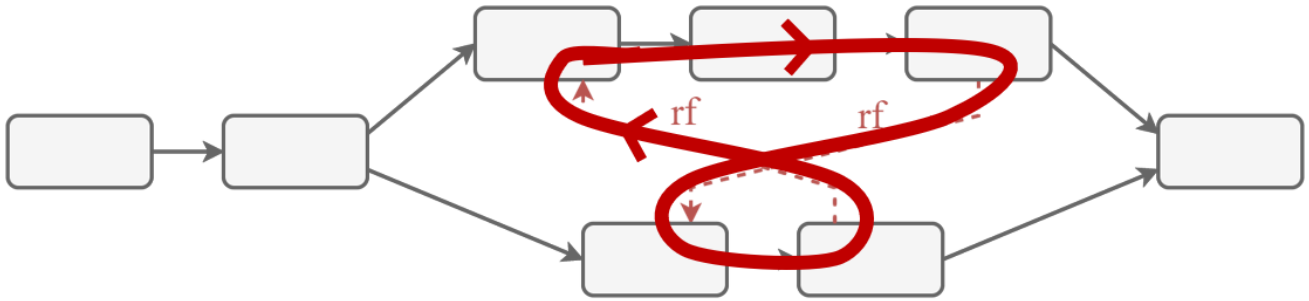
New approach

- Bounded Model Checking (BMC) for handling concurrency
- Reasoning with **partial orders** (happens-before relation)
- Key idea:
  - Build a **happens-before relation**
    (program order + dataflow + axioms)
  - **Avoid cycles**
    (exclude cycles/backtrack in the search space)



New approach

- Bounded Model Checking (BMC) for handling concurrency
- Reasoning with **partial orders** (happens-before relation)
- Key idea:
  - Build a **happens-before relation**
    (program order + dataflow + axioms)
  - **Avoid cycles**
    (exclude cycles/backtrack in the search space)



New approach

- Bounded Model Checking (BMC) for handling concurrency
- Reasoning with **partial orders** (happens-before relation)

New approach

- Bounded Model Checking (BMC) for handling concurrency
- Reasoning with **partial orders** (happens-before relation)
- Different **decision procedures**:
  - **Refinement**-based: detect cycles on **full models** provided by an SMT solver
  - **Propagator**-based: detect cycles **on-the-fly** in a **custom SMT theory** (Z3 user propagator interface via JavaSMT)

New approach

- Bounded Model Checking (BMC) for handling concurrency
- Reasoning with **partial orders** (happens-before relation)
- Different **decision procedures**:
  - **Refinement**-based: detect cycles on **full models** provided by an SMT solver
  - **Propagator**-based: detect cycles **on-the-fly** in a **custom SMT theory** (Z3 user propagator interface via JavaSMT)
- **Automatic cycle avoidance**: find&exclude cycles before starting the search space exploration

Portfolio & Results

Portfolio & Results

- Base strategy:
  - **BMC first**
  - If no (reliable) result: **fallback on abstraction-based methods**

Portfolio & Results

- Base strategy:
  - **BMC first**
  - If no (reliable) result: **fallback on abstraction-based methods**
- Solved tasks by Theta in concurrency compared to last year:
  (on common tasks)