Domain-Independent Multi-threaded Software Model Checking

Dirk Beyer and Karlheinz Friedberger

LMU Munich, Germany

ASE 2018, Montpellier
Software Verification

C Program

```c
int main() {
    int a = foo();
    int b = bar(a);
    assert(a == b);
}
```

Verification Tool

- TRUE
  i.e., specification is satisfied
- FALSE
  i.e., bug found

General method:
Create an overapproximation of the program states /
compute program invariants
Iterative fixpoint successor computation

\[ R_0 \]
Iterative fixpoint successor computation
Iterative fixpoint successor computation
Software Verification by Model Checking
[Clarke/Emerson, Sifakis 1981]

Iterative fixpoint successor computation
Basic Challenges with Software Verification

- more domains
- larger tasks
- bigger hardware
Basic Challenges with Software Verification

configurable program analysis ✓

more domains

larger tasks

bigger hardware
Basic Challenges with Software Verification

- More domains
- Larger tasks
- Bigger hardware
- Configurable program analysis
- Abstraction
- Caching
- Block summaries
Basic Challenges with Software Verification

- More domains
- Configurable program analysis
- Abstraction
- Larger tasks
- Caching
- Block summaries
- Portfolio
- Bigger hardware
- Parallel analysis?
- Scheduling?
- Synchronization?
Configurable Program Analysis (CPA)
[Beyer/Henzinger/Théoduloz, 2007]

- CPA algorithm computes a fixed-point based on two sets of abstract states
  - *reached*: already analyzed abstract states
  - *waitlist*: frontier states to be analyzed
Configurable Program Analysis (CPA)
[Beyer/Henzinger/Theoduloz, 2007]

- CPA algorithm computes a fixed-point based on two sets of abstract states
  - reached: already analyzed abstract states
  - waitlist: frontier states to be analyzed
- Operators defined for specific domain:
  - transfer: successor computation
  - merge: combination of two abstract states
  - stop: coverage of abstract states
Configurable Program Analysis (CPA)
[Beyer/Henzinger/Théoduloz, 2007]

- CPA algorithm computes a fixed-point based on two sets of abstract states
  - reached: already analyzed abstract states
  - waitlist: frontier states to be analyzed
- Operators defined for specific domain:
  - transfer: successor computation
  - merge: combination of two abstract states
  - stop: coverage of abstract states

✓ Independent of used domain
Configurable Program Analysis (CPA)
[Beyer/Henzinger/Théoduloz, 2007]

- CPA algorithm computes a fixed-point based on two sets of abstract states
  - *reached*: already analyzed abstract states
  - *waitlist*: frontier states to be analyzed
- Operators defined for specific domain:
  - *transfer*: successor computation
  - *merge*: combination of two abstract states
  - *stop*: coverage of abstract states

✓ Independent of used domain
✗ Operators strictly sequential (per analysis!)
Schematic Example of an Analysis
Schematic Example of an Analysis

A

plain analysis

A1

time
Basic Challenges with Software Verification

- more domains
- bigger hardware
- larger tasks
- configurable program analysis
- abstraction
- caching
- block summaries
- parallel analysis?
- scheduling?
- synchronization?
- portfolio

Dirk Beyer and Karlheinz Friedberger
LMU Munich, Germany
Block Summarization

- Block-abstraction memoization (BAM) defined as CPA
  [Wonisch/Wehrheim, 2012]
- Split large verification task into smaller problems
  and solve them separately
- Use CPA algorithm for a domain-specific analysis
- Cache intermediate analysis results
Block Summarization

- Block-abstraction memoization (BAM) defined as CPA
  [Wonisch/Wehrheim, 2012]
- Split large verification task into smaller problems
  and solve them separately
- Use CPA algorithm for a domain-specific analysis
- Cache intermediate analysis results

✅ Independent of domain-specific analysis
Block Summarization

- Block-abstraction memoization (BAM) defined as CPA [Wonisch/Wehrheim, 2012]
- Split large verification task into smaller problems and solve them separately
- Use CPA algorithm for a domain-specific analysis
- Cache intermediate analysis results

✔ Independent of domain-specific analysis
✗ Dependencies between block abstractions
Block Summarization

- Block-abstraction memoization (BAM) defined as CPA [Wonisch/Wehrheim, 2012]
- Split large verification task into smaller problems and solve them separately
- Use CPA algorithm for a domain-specific analysis
- Cache intermediate analysis results

✅ Independent of domain-specific analysis
❌ Dependencies between block abstractions
✅ Configurable block size
Schematic Example of an Analysis

plain analysis

Dirk Beyer and Karlheinz Friedberger
LMU Munich, Germany
Schematic Example of an Analysis

plain analysis
Schematic Example of an Analysis

plain analysis

BAM
Basic Challenges with Software Verification

- more domains
- bigger hardware
- larger tasks
- configurable program analysis
- abstraction
- portfolio
- caching
- block summaries
- parallel analysis?
- scheduling?
- synchronization?
Parallel Block-Abstraction Memoization

Challenges with an efficient parallel algorithm:

× CPA operators strictly sequential (per analysis!)
× Dependencies between block abstractions
Parallel Block-Abstraction Memoization

Challenges with an efficient parallel algorithm:
- CPA operators strictly sequential (per analysis!)
- Dependencies between block abstractions

Our contribution: Parallel BAM
- Parallel computation of block abstractions (asynchronously)
- Lazy application of computed block abstractions
- Simple dynamic scheduler
Parallel Block-Abstraction Memoization

Challenges with an efficient parallel algorithm:
- CPA operators strictly sequential (per analysis!)
- Dependencies between block abstractions

Our contribution: Parallel BAM
- Parallel computation of block abstractions (asynchronously)
- Lazy application of computed block abstractions
- Simple dynamic scheduler

✓ Combines benefits of existing approaches
Parallel Block-Abstraction Memoization

Challenges with an efficient parallel algorithm:

- CPA operators strictly sequential (per analysis!)
- Dependencies between block abstractions

Our contribution: Parallel BAM

- Parallel computation of block abstractions (asynchronously)
- Lazy application of computed block abstractions
- Simple dynamic scheduler

- Combines benefits of existing approaches
- Small synchronization overhead (depends on block size)
Schematic Example of an Analysis

plain analysis

BAM
Schematic Example of an Analysis

plain analysis

BAM

parallel BAM
Basic Challenges with Software Verification

- more domains
- bigger hardware
- larger tasks
- configurable program analysis
- abstraction
- caching
- block summaries
- portfolio
- parallel analysis?
- scheduling?
- synchronization?
Evaluation

- Configuration
  - CPAchecker r28809
  - Explicit Value Domain

- Environment
  - Intel Xeon E3-1230 v5 CPU with 4 physical cores
  - 5400 tasks from SV-COMP benchmark set

- Limitations
  - 15 GB RAM
  - 15 minutes
Evaluation

![Graph showing evaluation results with x-axis representing n-th fastest result and y-axis representing wall time (s). The graph includes lines for different numbers of threads: 1 (number of threads), 2, 4, and 8. The x-axis ranges from 0 to 3000, and the y-axis ranges from 0 to 1000.](image-url)
Evaluation

![Graph showing the relationship between n-th fastest result and wall time (s) for different numbers of threads.]

Dirk Beyer and Karlheinz Friedberger LMU Munich, Germany
Basic Challenges with Software Verification

- More domains
- Configurable program analysis
- Abstraction
- Caching
- Block summaries
- Larger tasks
- Bigger hardware
- Portfolio
- Parallel analysis
- Scheduling
- Synchronization
Conclusion

- Configurable program analysis
- Configurable block size
- Small overhead for synchronization in parallel analysis
- Elegant integration into the framework CPAchecker
- No changes necessary to existing analyses and components
  - CEGAR, proof and counterexample witnesses
Future work

- Scheduling: Prefer parts deeper in the program?
- Processes instead of threads
  - Cluster instead multi-core machine
Questions?
BAM in CPAchecker

Source Code → Parser & CFA Builder → CPA Algorithm

CPA Algorithm

wait for nested analysis

Spec → Spec CPA, Location CPA, Callstack CPA, Predicate CPA

BAM CPA → Results
Parallel BAM in CPAchecker

Source Code → Parser & CFA Builder → CEGAR Algorithm → Results

Parallel BAM Algorithm

CPA Algorithm

enqueue missing block abstraction

Spec

N instances

Spec CPA → Location CPA → Callstack CPA → Value CPA

Dirk Beyer and Karlheinz Friedberger
LMU Munich, Germany