SMT-based Software Model Checking: Experimental Comparison of Four Algorithms

Matthias Dangl
Joint work with Dirk Beyer

University of Passau, Germany

UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

SoSy-Lab
Software Systems

SMT-based Software Model Checking

- Bounded Model Checking
  (Cbmc, CPAchecker, Esbmc, ...)
- k-Induction
  (CPAchecker, Esbmc, 2LS, ...)
- Predicate Abstraction
  (Blast, CPAchecker, Slam, ...)
- Impact
  (CPAchecker, Impact, Wolverine, ...)
- Property-Directed Reachability (PDR, also known as IC3)
  (Seahorn, VVT, ...)
- ...

SMT-based Software Model Checking

- Bounded Model Checking
  (CBMC, CPACHECKER, ESBMC, ...)
- k-Induction
  (CPACHECKER, ESBMC, 2LS, ...)
- Predicate Abstraction
  (BLAST, CPACHECKER, SLAM, ...)
- Impact
  (CPACHECKER, IMPACT, WOLVERINE, ...)

Our Goals

- Perform an extensive comparative evaluation
- Confirm intuitions about strengths
- Determine potential of extensions and combinations

Approach

- Understand, and, if necessary, re-formulate the algorithms
- Implement all algorithms in one tool (CPAchecker)
- Run the algorithms on a large set of benchmarks
- Measure efficiency and effectiveness

Experimental Validity: All Algorithms in one Tool

Compare algorithms, not tools:

- Share same front-end code
- Share same utilities
- Share same SMT-solver integration
- Share algorithm-independent optimizations

→ Differences in performance must be caused by algorithms

Bounded Model Checking

- Bounded Model Checking:
  - Biere, Cimatti, Clarke, Zhu: [TACAS'99]
  - No abstraction
  - Unroll loops up to a loop bound k
  - Check that P holds in the first k iterations:

$$\bigwedge_{i=1}^{k} P(i)$$

  - Good for finding bugs

k-Induction

- k-Induction generalizes the induction principle:
  - No abstraction
  - Base case: Check that P holds in the first k iterations:
    $\rightarrow$ Equivalent to BMC with loop bound k
  - Step case: Check that the safety property is k-inductive:

$$\forall n : \left(\left(\bigwedge_{i=1}^{k} P(n+i-1) \right) \implies P(n+k) \right)$$

  - Stronger hypothesis is more likely to succeed
  - Add auxiliary invariants
  - Kahsai, Tinelli: [PDMC'11]
  - Heavy-weight proof technique

k-Induction with Auxiliary Invariants

Induction:

$k = 1$
while !finished **do**
    BMC(k)
    Induction(k, invariants)
    k++

Invariant generation:

prec $=$ $<$weak$>$
invariants $= \emptyset$
while !finished **do**
    invariants $=$ GenInv(prec)
    prec $=$ RefinePrec(prec)

Predicate Abstraction

- Predicate Abstraction
  - Graf, Saïdi: [CAV'97]
  - Abstract-Interpretation technique
  - Abstract domain constructed from a set of predicates π
  - Use CEGAR to add predicates to π (refinement)
  - Derive new predicates using Craig interpolation
  - Good for finding proofs

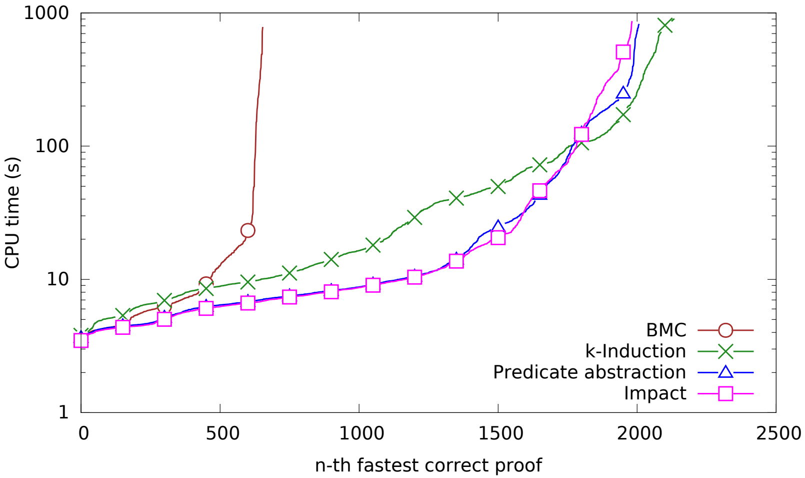
Impact

- Impact
  - "Lazy Abstraction with Interpolants"
  - McMillan: [CAV'06]
  - Counter-draft to predicate abstraction
  - Abstraction is derived dynamically/lazily
  - Solution to avoiding expensive abstraction computations
  - Compute fixed point over three operations
    - Expand
    - Refine
    - Cover
  - Quick exploration of the state space
  - Good for finding bugs

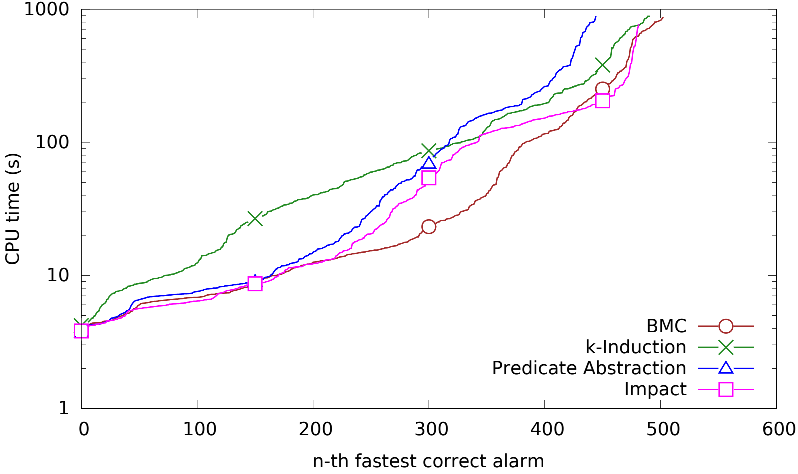
Experimental Comparison

- 4 779 verification tasks taken from SV-COMP'16
- 15 min timeout (CPU time)
- 15 GB memory
- Measured with BenchExec

All 3 459 bug-free tasks



All 1 320 tasks with known bugs

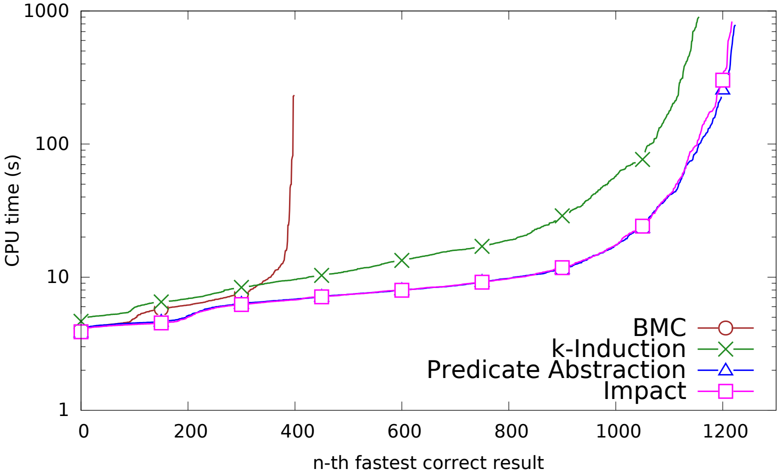


Category: Device Drivers

- Several thousands LOC per task
- Complex structures
- Pointer arithmetics

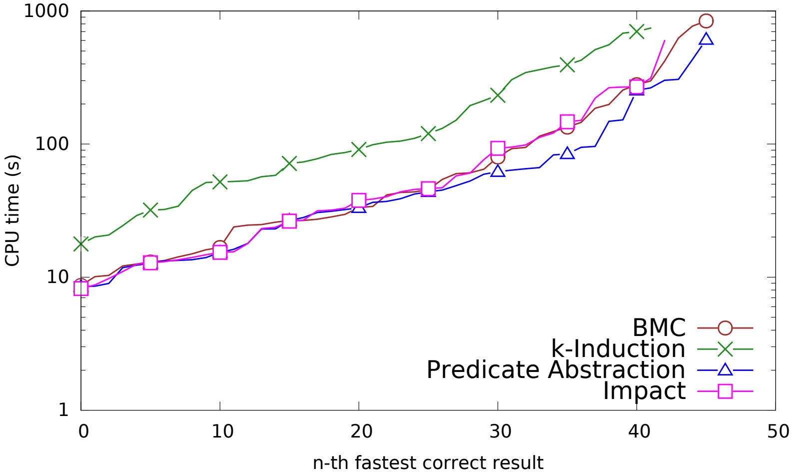
Category: Device Drivers

1 857 bug-free tasks:



Category: Device Drivers

263 tasks with known bugs:



Category: Event Condition Action Systems

- Several thousand LOC per task
- Auto-generated
- Only integer variables
- Linear and non-linear arithmetics
- Complex and dense control structure

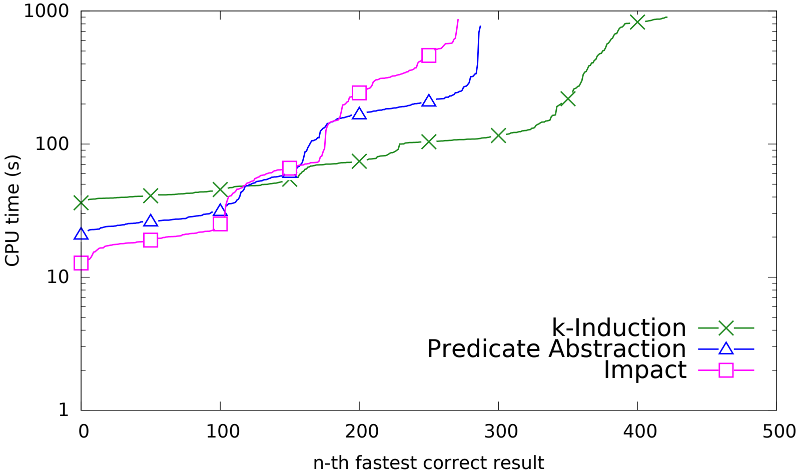
Category: Event Condition Action Systems

- Several thousand LOC per task
- Auto-generated
- Only integer variables
- Linear and non-linear arithmetics
- Complex and dense control structure

```
if (((a24==3) && (((a18==10) && ((input == 6)
    && ((115 < a3) && (306 >= a3))))
    && (a15==4)))) {
  a3 = (((a3 * 5) + -583604) * 1);
  a24 = 0;
  a18 = 8;
  return -1;
}
```

Category: Event Condition Action Systems

734 bug-free tasks:



Category: Event Condition Action Systems

406 tasks with known bugs:

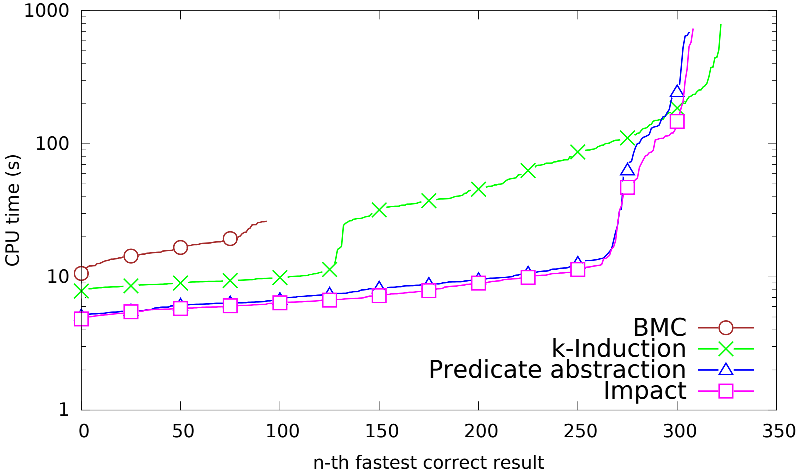
Only BMC and k-Induction find one bug (the same one).

Category: Product Lines

- Several hundred LOC
- Mostly integer variables, some structs
- Mostly simple linear arithmetics
- Lots of property-independent code

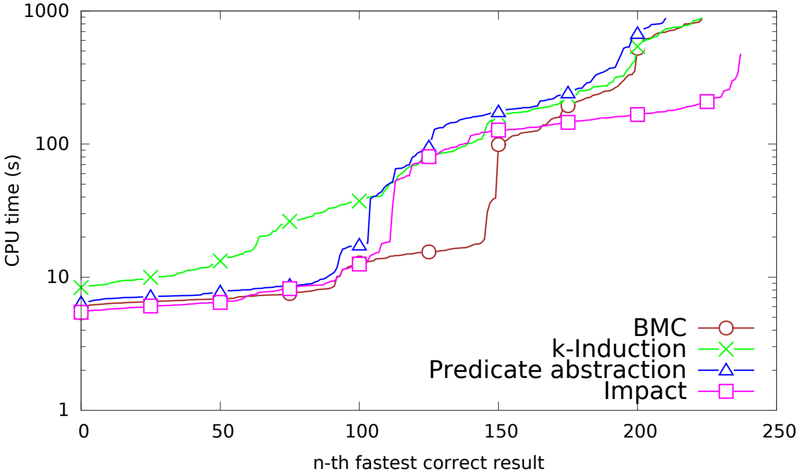
Category: Product Lines

332 bug-free tasks:



Category: Product Lines

265 tasks with known bugs:



Summary

We reconfirm that

- BMC is a good bug hunter
- k-Induction is a heavy-weight proof technique: effective, but slow
- CEGAR makes abstraction techniques (Predicate Abstraction, Impact) scalable
- Impact is lazy, and explores the state space and finds bugs quicker
- Predicate Abstraction is eager, and prunes irrelevant parts and finds proofs quicker

Outlook

- Abstraction is required for scalability
- k-Induction needs some form of abstraction
- Maybe the ideas of k-Induction can be transferred to PDR