DepthK: A k-Induction Verifier Based on Invariant Inference for C Programs
(Competition Contribution)

Williame Rocha, Herbert Rocha, Hussama Ismail, Lucas Cordeiro, and Bernd Fischer

ETAPS
EUROPEAN JOINT CONFERENCES ON
THEORY & PRACTICE OF SOFTWARE

TACAS 2017
COMPETITION ON SOFTWARE VERIFICATION (SV-COMP)
DepthK employs **Bounded Model Checking (BMC)** and **k-Induction** based on program invariants, which are automatically generated using **polyhedral constraints**

- **✓** DepthK uses ESBMC, a context-bounded symbolic model checker that verifies single- and multi-threaded C programs
- **✓** The k-induction step: base case, forward condition and inductive step
- **✓** DepthK uses PAGAI (SVCOMP’17) and PIPS tools to infer program invariants
- **✓** DepthK integrates the witness checkers CPAchecker and Ultimate Automizer for checking verification results
Verification Approach

DepthK is a **source-to-source transformation** tool that extends ESBMC to falsify or prove correctness of a given (safety) property for any depth without manual annotation of **loop invariants**

- **PAGAI** applies source code analysis to infer invariants for each control-flow point of a C program using LLVM

- In **PIPS**, for each program instruction, the polyhedral invariants are propagated along with instructions, using the previously computed transformers
Verification Approach

```
int x = 0;
int t = 0;
int phase = 0;
while (t < 100)
    if (phase == 0)
        x = x + 2;
    if (phase == 1)
        x = x - 1;
    phase = 1 - phase;
    t++;
assert(t <= 100);
```
Verification Approach

polyhedral abstraction

source-to-source transformation

```
int x = 0;
int t = 0;
int phase = 0;
while (t < 100)
    if (phase == 0)
        x = x + 2;
    else if (phase == 1)
        x = x - 1;
    phase = 1 - phase;
    ++t;
assert(t <= 100);
```
Verification Approach

Overapproximates behavior

Re-checking procedure

k-Induction algorithm

Warning in state exploration

**Invariant Generator**

**Invariant Translation**

**ESBMC Verifier**

**Witness Validator**

**Base Case** \( P, \phi, k = I \land T \land \sigma \Rightarrow \phi \)

**Forward Cond** \( P, \phi, k = I \land T \Rightarrow \sigma \land \phi \)

**Inductive Step** \( P, \phi, k = \gamma \land \sigma \Rightarrow \phi \)

**Instrumentation with:**

```c
__ESBMC_ASSUME(invariant)
```

```c
int x = 0;
int t = 0;
int phase = 0;
while (t < 100)
  if (phase == 0)
    x = x + 2;
  if (phase == 1)
    x = x - 1;
  phase = 1 - phase;
  +t;
assert(t <= 100);
```
Strengths and Weaknesses

✓ The **tool lies in the combination** of the *k-induction algorithm with program invariants* to specify pre- and post-conditions

✓ In preliminary experiments, **PAGAI/PIPS** tools were unable to produce **inductive invariants** for the *k*-induction algorithm, either due to a **weak transformer** or **not convex invariants**

✓ All incorrect answers produced by our tool in the competition are due to **bugs in its implementation**
  - Witness validation issues to confirm DepthK results
  - Trace back the data in the source code transformation
Strengths and Weaknesses

SV-COMP’17 results:

✓ Improvements over “plain” ESBMC

✓ DepthK outperforms all ESBMC versions in:
  § ReachSafety-BitVectors
  § ReachSafety-Heap
  § ReachSafety-Loops
  § MemSafety-Arrays

✓ DepthK outperforms CPA-kInd:
  § ReachSafety-Heap
  § ReachSafety-Recursive
  § Overflows-BitVectors
  § Category FalsificationOverall
DepthK tool is available at [https://github.com/hbgit/depthk/archive/depthk_v3.tar.gz](https://github.com/hbgit/depthk/archive/depthk_v3.tar.gz)
Thank you for your attention!

herberthb12@gmail.com

https://github.com/hbgit/depthk