CPAchecker for Reachability, Memory Safety, Overflows, Concurrency, and Termination

(Competition Contribution for SVCOMP’17)

Matthias Dangl and Karlheinz Friedberger

Overview

CPAchecker is a modern framework for software verification and is based on well-known concepts like
- CEGAR
- Configurable program analysis (CPA) [2]
- Interpolation
- Predicate abstraction [3]
- Explicit-state model checking [4]
- \(k\)-induction [1]
- Lazy abstraction
- Abstract interpretation
- Block-abstraction memoization [7]

CPAchecker has support for several abstract domains, such as values, intervals, octagons, BDDs, predicates, and memory graphs, which can all be used to build an analysis that matches the user’s requirements.

Setup and Configuration

Download CPAchecker from https://cpachecker.sosy-lab.org

and execute

scripts/cpa.sh -sv-comp17
  -disable-java-assertions -heap 10000m
  -spec property.prp program.i

The configuration sv-comp17 is
- optimized for checking a wide range of properties,
- an effective approach for solving a heterogeneous set of verification tasks, and
- based on several verification approaches, from reachability analysis to synthesized ranking functions.

Sequential Composition

Up to four bit-accurate analyses are executed in sequence. After a time limit is exceeded the next analysis is started, if no result is available yet.

(1) Value analysis [4] tracks values of integer variables explicitly. It is efficient but imprecise for non-deterministic variables.
(4) \(k\)-induction [1] uses auxiliary invariants and extends bounded model checking from falsification to verification.

Counterexamples are cross-checked with a bit-accurate counterexample check, i.e., with predicate analysis for (1,2) and CBMC for (3,4).

References