Enhancing CPAchecker: A Framework for Distributed Analyses

Dirk Beyer, **Matthias Kettl**, Thomas Lemberger

2023-09-11

LMU Munich, Germany

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SoSy-Lab Software Systems

- Scalability
- Faster response time

Distributed CPAs

Definition (CPA [1])

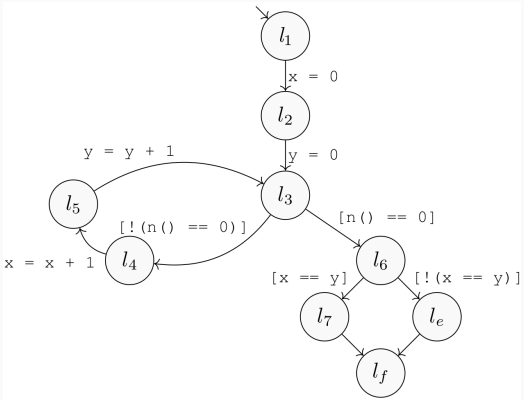
For abstract states A and edges G, a CPA is defined as a four-tuple $(\mathbb{D}, \rightsquigarrow \subseteq A \times G \times A, \text{merge} : \mathsf{A} \times \mathsf{A} \mapsto \mathsf{A}, \text{stop} : \mathsf{A} \times 2^{\mathsf{A}} \mapsto \{\text{tt}, \text{ff}\})$

Definition (DCPA)

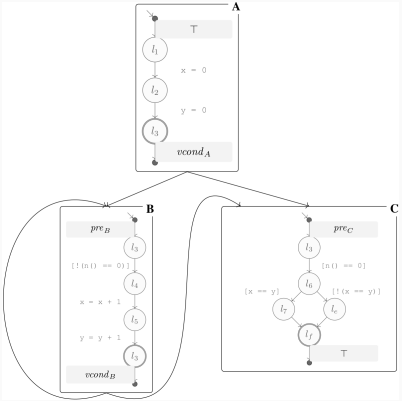
Distributed CPAs (DCPAs) enhance existing CPAs by two operators:

- $\text{pack} : A \mapsto (M \cup \bot_M)$; serializes abstract states to messages M
- $\text{unpack} : M \mapsto A$; deserializes messages M to abstract states

Block Graph from a CFA



CFA



Block Graph

DCPA Algorithm

Algorithm 1 Message processing of a $\mathrm{DCPA}_b = (\mathrm{CPA},\ \mathrm{pack},\ \mathrm{unpack})$

Input: Received messages M_r

Output: Messages from the computed reached set

1: $M_f = \mathsf{filter}_b(M_r)$
2: $A_{init} = \{\mathrm{unpack}(m) \mid m \in M_f\}$ // if empty, $\mathrm{CPA}(R_{init}) = \{\}$
3: $R_{init} = \mathsf{init}(A_{init})$ // runs the CPA algorithm on A_{init}
4: **return** $\{\mathrm{pack}(a) \mid a \in \mathrm{CPA}(R_{init})\} \setminus \{\bot_M\}$

DCPA Protocols

Definition (DCPA Protocol)

A protocol $\rho_{\mathsf{DCPA}_b} : 2^M \mapsto 2^M$ reacts to a set of messages N with

$$\rho_{\mathsf{DCPA}_b}(N) = \rho_{\mathsf{DCPA}_b}(N_A \dot{\cup} N_{\bar{A}}) = \mathsf{DCPA}_b(N_A) \cup p(N_{\bar{A}})$$

where $p : 2^M \mapsto 2^M$.

Example: Infer

- INFER [2] decomposes the CFA in function-wise blocks.
- INFER outputs FALSE iff there exists at least one violation.
- INFER does not care about the reachability of functions.

Example: DCPA for Infer

The INFER approach works for (any) DCPA($\mathbb{D}$, stop, merge, $\rightsquigarrow$, pack, unpack)

- Block nodes cover functions but ignore nested function calls.
- $\text{pack}(a) = \begin{cases} \text{loc}(a)_{\bar{A}}, \text{ if } a \text{ is a target state} \\ \top_A, \text{ if } a = \top_{\mathbb{D}} \\ \text{SAFE}_{\bar{A}}, \text{ otherwise} \end{cases}$
- $\text{unpack}(m) = \top_{\mathbb{D}}$
- Special worker tracks target locations and eventually broadcasts final verdict.

References

[1] Beyer, D., Henzinger, T.A., Théoduloz, G.: Configurable software verification: Concretizing the convergence of model checking and program analysis. In: Damm, W., Hermanns, H. (eds.) Computer Aided Verification. pp. 504–518. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[2] Calcagno, C., Distefano, D., Dubreil, J., Gabi, D., Hooimeijer, P., Luca, M., O'Hearn, P., Papakonstantinou, I., Purbrick, J., Rodriguez, D.: Moving fast with software verification. In: Havelund, K., Holzmann, G., Joshi, R. (eds.) NASA Formal Methods. pp. 3–11. Springer International Publishing, Cham (2015)