CUBA: Interprocedural Context-UnBounded Analysis of Concurrent Programs
Peizun Liu and Thomas Wahl, Northeastern University, Boston, MA  {lpzun, wahl}@ccs.neu.edu

- Reachability for concurrent threads running recursive procedures is undecidable [1].
- Bounding the number of context switches allowed between the threads sidesteps the undecidability and leads to a bug-finding technique [2].
- Can above technique also prove the absence of bugs, for an arbitrary number of contexts?
- We call this challenge the Context-UnBounded Analysis (CUBA) problem and propose a solution.

Unbounded contexts:

\[ T_1: \emptyset \rightarrow \{0\} \rightarrow \{1\} \rightarrow \{0\} \rightarrow \{1\} \rightarrow \{0\} \rightarrow \ldots \]

\[ T_2: \emptyset \rightarrow \{0\} \rightarrow \{1\} \rightarrow \{0\} \rightarrow \{1\} \rightarrow \{0\} \rightarrow \ldots \]

Example

- We introduce a broad verification methodology for resource-parameterized programs that observes how changes to the resource parameter affect the behavior of the program.
- Applied to CUBA, the methodology results in partial verification techniques for procedural concurrent programs.
- Our solutions may not terminate, but can both refute and prove context-unbounded safety for concurrent recursive programs.
- We demonstrate the effectiveness of our method using a variety of examples, the safe of which cannot be proved safe by context-bounded methods.

Problem

- "Example Revisited"

Shared states:

\[ Q = \{ 0, 1, 2, 3 \} \]

Thread 1:

\[ \Sigma_1 = \{ 1, 2 \} \]

\[ \Delta_1 = \{ f_1: (0, 1) \rightarrow (1, 2), f_2: (3, 2) \rightarrow (0, 1) \} \]

Thread 2:

\[ \Sigma_2 = \{ 4, 5, 6 \} \]

\[ \Delta_2 = \{ b_1: (0, 4) \rightarrow (0, 5), b_2: (1, 4) \rightarrow (0, 5), b_3: (2, 5) \rightarrow (3, 4) \} \]

\[ q' = 0 \]

A 2-thread concurrent pushdown system \( P^2 \) (left) and its reachability table (right). We have \( P^2 = \{ P_1, P_2 \} \) with \( P_i = (Q, \Sigma_i, \Delta_i, q_i^0) \) for \( i = 1, 2 \); the initial state is \( (0|1, 4) \). The table shows the sets \( R_k \setminus R_{k-1} \) and \( T_k \setminus T_{k-1} \) of reachable states and of reachable visible states, resp., that are new at bound \( k \), for \( k = 1, \ldots, 6 \), where \( T_k := T(R_k) := \{ (T(s): s \in R_k) \} \), and \( T(s) \) is illustrated on the right.

Implemention

1. Published in PLDI 2018.
2. Tool is available online [3].

Future Work

1. Compute \( T_k \) via abstract interpretation.
2. Improve the scalability of CUBA.

Program Analysis using Observation Sequences

Definition. An observation sequence (OS) is a sequence \( (O_k)_{k=0}^\infty \) with the following properties:

- for all \( k \), \( O_k \subseteq O_{k+1} \) (monotonicity).
- for all \( k \), \( O_k \) is computable.
- for all \( k \), \( O_k \models \Phi \) is decidable, where \( \Phi \) is a property of interest.
- for all \( k \), \( O_k \models \Phi \Rightarrow P \models \Phi \).

Challenge. An OS may never converge.

Property. An OS over a finite domain converges.

Example

A set \( G \) of visible states is a generator set if the following formula is valid for every \( k \):

\[ T_{k-1} \cap \{ \sigma \} \subseteq \{ \sigma \} \]

Challenges.

1. How to define \( G \)?
2. \( T \) is equivalent to the final goal. A paradox?

Solutions.

1. We define \( G \) as follows:

\[ G = \{ (q|\sigma_1, \ldots, \sigma_n) | \text{there exists } i \text{ s.t.} \} \]

\[ \langle q, \epsilon \rangle \text{ is the target of a pop edge in } \Delta_i \text{ and } \langle \epsilon, \sigma_i \rangle \text{ is the target of a push edge in } \Delta_i \} \]

2. Overapproximate \( T \) statically.

References