CUBA: Interprocedural Context-UnBounded Analysis of Concurrent Programs

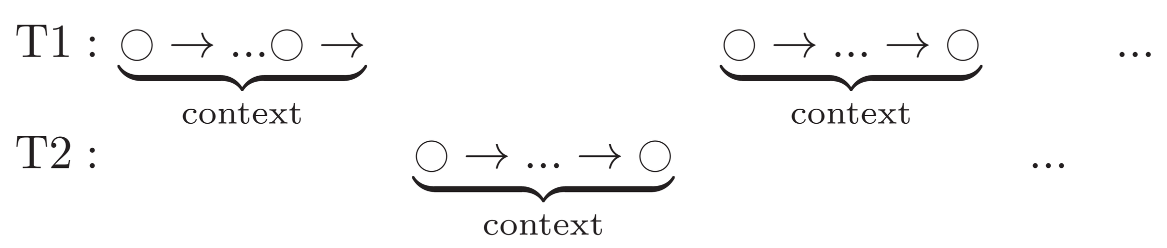
Northeastern University Khoury College of Computer Sciences

Peizun Liu and Thomas Wahl, Northeastern University, Boston, MA {lpzun, wahl}@ccs.neu.edu

Problem

- 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:



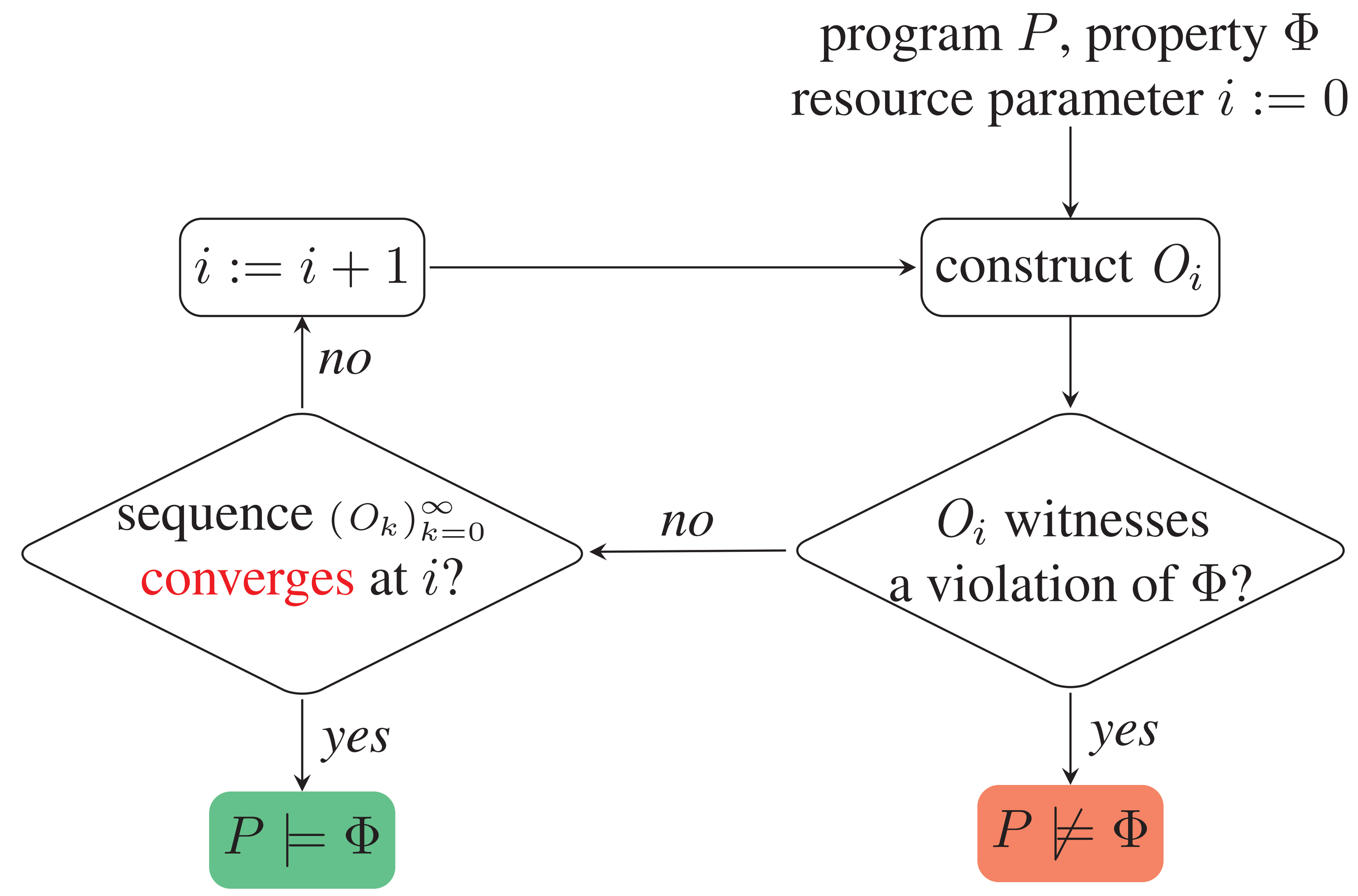
Contributions

- 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.

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* Φ *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

Shared states:

$Q = \{ 0, 1, 2, 3 \}$

Thread 1:

$\Sigma_1 = \{ 1, 2 \}$

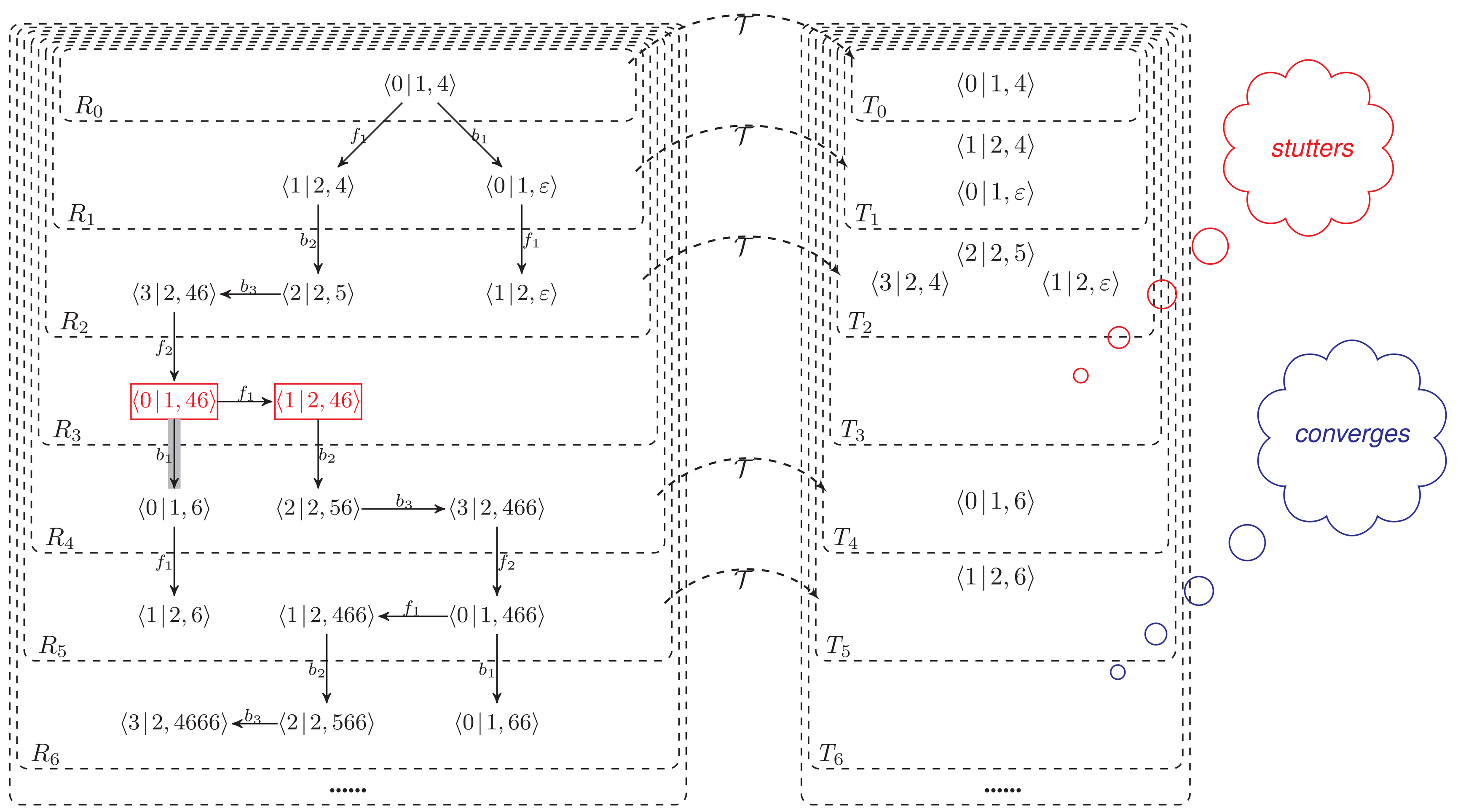
$\Delta_1 = \{ f_1 : (0, 1) \to (1, 2)$, $f_2 : (3, 2) \to (0, 1) \}$

Thread 2:

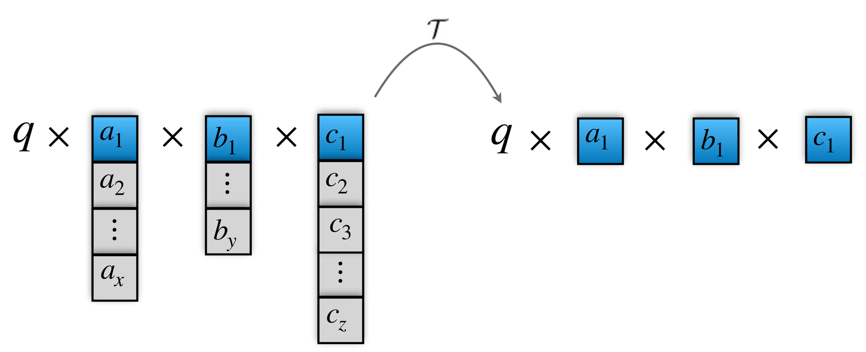
$\Sigma_2 = \{ 4, 5, 6 \}$

$\Delta_2 = \{ b_1 : (0, 4) \to (0, \varepsilon)$, $b_2 : (1, 4) \to (2, 5)$, $b_3 : (2, 5) \to (3, 46) \}$

$q^I = 0$



A 2-thread concurrent pushdown system P^2 (left) and its *reachability table* (right). We have $P^2 = \{P_1, P_2\}$ with $P_i = (Q, \Sigma_i, \Delta_i, q^I)$ for $i = 1, 2$; the initial state is $\langle 0 \,|\, 1, 4 \rangle$. 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 $k = 1, \dots, 6$, where $T_k := \mathcal{T}(R_k) := \{\mathcal{T}(s) : s \in R_k\}$, and $\mathcal{T}(s)$ is illustrated on the right.



Stuttering Detection using Generators

A set $\mathcal{G}$ of visible states is a **generator set** if the following formula is valid for every k:

$$T_{k-1} = T_k \wedge \mathcal{G} \cap T \subseteq T_k \Rightarrow T_k = T.$$

Challenges.

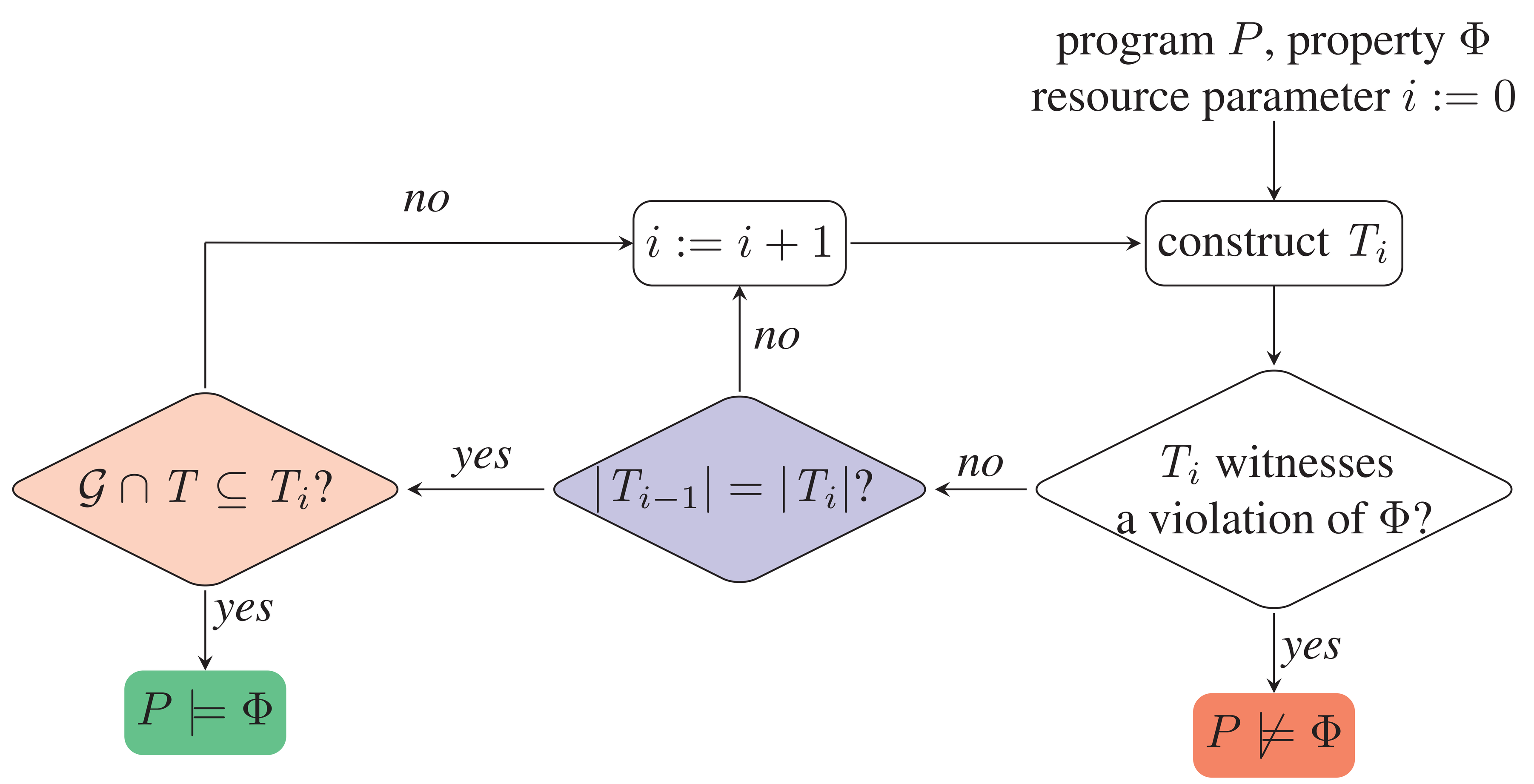
1. How to define $\mathcal{G}$?
2. T is equivalent to the final goal. A paradox?

Solutions.

1. We define $\mathcal{G}$ as follows:

$$\mathcal{G} = \{ \langle q \,|\, \sigma_1, \dots, \sigma_n \rangle \mid \text{there exists } i \text{ s.t. } (q, \varepsilon) \text{ is the target of a } pop \text{ edge in } \Delta_i \quad \textbf{and} \quad (\sigma_i = \varepsilon \ \textbf{or} \ (?, ?\sigma_i) \text{ is the target of a } push \text{ edge in } \Delta_i \ ) \}$$

2. Overapproximate T statically.



Implementation

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.

References

[1] G. Ramalingam. "Context-sensitive Synchronization-sensitive Analysis is Undecidable." *ACM TOPLAS*, pp. 416–430, 2000.
[2] Shaz Qadeer and Jakob Rehof. "Context-Bounded Model Checking of Concurrent Software." *TACAS*, pp. 93–107, 2005.
[3] Peizun Liu and Thomas Wahl. "The Homepage of CUBA". *http://www.ccs.neu.edu/home/lpzun/cuba/*, 2018.