On-the-fly Parameterized Boolean Program Exploration

Peizun Liu and Zhaoliang Liu
Northeastern University, USA, \{lpzun, zliu\}@ccs.neu.edu

Abstract—Reachability analysis for replicated Boolean programs run by an unbounded number of threads is decidable in principle via a reduction of the Boolean program families to well-structured transition systems (WSTS). The obtained transition systems would, however, in general be intractably large, due to local state explosion. Basler et al. give an on-the-fly algorithm that solves this problem for Boolean programs run by a fixed, finite number of threads [1]. In this paper, we extend this idea to families with unbounded thread counts, based on Abdulla’s backward reachability analysis [2]. The challenges are to traverse the Boolean programs backwards, computing infinite-state covering pre-images, all while building the transition system being explored on the fly, to avoid local state explosion.

I. INTRODUCTION

We consider the class of replicated finite-state programs executed by an unbounded number of threads. An important practical instance of this class is given by non-recursive concurrent Boolean programs with dynamic thread creation. Safety properties of such programs can in principle be checked via reduction to the coverability problem for WSTS.

An approach to solving the latter problem is the backward reachability algorithm proposed by Abdulla et al. [2]. To apply this algorithm to Boolean program families, these families need to be translated into infinite-state machines. This translation can cause a huge blow-up, triggered by the need to track a thread’s local state changes in updates to local state counters. Basler et al. [1] proposed a context-aware strategy, which performs the translation on the fly, instead of constructing the entire transition model upfront. Their method is, however, restricted to finite thread families.

Contribution. In this work, we present a novel, sound and complete solution to this problem, by extending the on-the-fly idea from bounded to unbounded thread counts. State space exploration is performed using covering pre-images [2], which traverse the infinite state space backwards. We then propose two optimizations to limit the size of obtained covering pre-images. To our knowledge this is the first work that applies on-the-fly techniques to sound and complete infinite Boolean program safety checking problems.

II. PRELIMINARIES

A. Basic Definitions

Let $\mathbb{B}$ be a non-recursive Boolean program. We adopt the Boolean program syntax from [3]; conditionals and loops are encoded implicitly using assume and goto statements. We

Research supported by the National Science Foundation, grant no. 1253331.
For any state $\tau$, we define $\hat{\tau} := \{ r \mid \tau \preceq r \}$. The covering predecessors of $\tau$ can be defined as follows: $\text{CovPre}(\tau) := \{ r \mid \exists r \to \tau', \tau' \in \hat{\tau} \}$, and $\text{C-Pre}(\tau) := \min \{ r : r \in \text{CovPre}(\tau) \}$.

III. ON-THE-FLY BACKWARD EXPLORATION

A. BWRA in Boolean Program

BWRA can be used in Boolean programs using the definition of $\preceq$. However, how to compute $\text{C-Pre}$ in a Boolean program is another impediment since we cannot do so by backward-executing the program as the forward search does.

1) Control Flow Graph and Weakest Precondition: In this project, we propose to compute $\text{C-Pre}$ based on control flow graph (CFG) and weakest precondition (WP). CFG is a directed graph constructed from the program's execution flow. The nodes and edges correspond to PCs and execution flows respectively. Given a statement $\text{stmt}_p$ with $p = p$ and a thread state $(s, \ell)$, in this paper, WP of $\text{stmt}_p$ is a function mapping $(s, \ell)$ to potential predecessors, which is denoted as $\text{WP}(\text{stmt}_p, s, \ell)$. We use CFG to navigate the backward search, and weakest precondition to approximate $\text{C-Pre}$.

2) On-the-fly Exploration: Suppose current state is $\tau = (s, F)$ with $F = \{ (\ell_1, n_1), \ldots, (\ell_k, n_k) \}$. Let $\Gamma = \{ \ell_1, \ldots, \ell_k \}$, CFG $G = (V, E)$, then on-the-fly BWRA algorithm executes as follows: for each $\ell \in L$, $\ell.pc = q$, if $\exists (p, q) \in G.E$, locate $\text{stmt}_p$, and then

(1) if $\ell \in \Gamma$ and $\text{WP}(\text{stmt}_p, s, \ell) \neq \emptyset$, suppose $(s', \ell') \in \text{WP}(\text{stmt}_p, s, \ell)$, then $(s', F') \in \text{C-Pre}(\tau)$, where $F' = F\setminus \ell \cup \ell'$.

(2) if $\ell \notin \Gamma$ and $\text{WP}(\text{stmt}_p, s, \ell) \neq \emptyset$, suppose $(s', \ell') \in \text{WP}(\text{stmt}_p, s, \ell)$, then $(s', F \cup \{(\ell', 1)\}) \in \text{C-Pre}(\tau)$.

Another technique to approximate reachable local configuration is borrowed from compilers, which sometimes optimize program behavior by confining the number of values that a local variable can have at some program point. Emerson and Wahl discussed this technique detailed in [4].

REFERENCES


