A Constraint-Based Approach to Multi-Threaded Program Location Reachability

Konstantinos Athanasiou
Northeastern University, Boston

Introduction

- Safety properties, such as absence of assertion failures, over shared-memory concurrent programs with unknown number of threads are decidable, yet of EXPSPACE complexity.
- Systems that require extra threads to serve requests coming from their environment showcase such behavior, since the number of threads can't be determined a priori.
- We consider an incomplete method, that can effectively decide a large number of benchmarks.

Method Overview

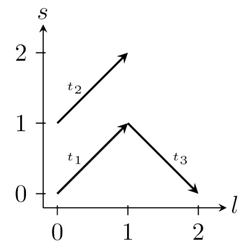
- Encode the reachability problem as a set of over-approximating integer linear constraints. If the constraints are unsatisfiable, then the safety property is verified.
- Otherwise, incrementally strengthen the constraints.
- If the constraints are still satisfiable, check if the safety property is violated, using a *guided* explicit-state search.

Thread-Transition Systems (TTS)

Finite-state models extracted through predicate abstraction of procedures executed by threads. Defined over

- the set of *thread states* $T = S \times L$, where S and L are the finite sets of *shared* (valuations of shared variables) and *local* states (valuations of local variables) respectively.
- the set of transitions $R \subseteq T \times T$

For n threads, a TTS gives rise to the state space $V_n = S \times L^n$ of configurations of the form $v = (s|l_1, \ldots, l_n) \in V_n$.
Transitions can asynchronously fire given that their source shared state is the same as the shared state of the current configuration, and they affect the local state of exactly one thread.



A witnessing sequence of configurations for $t_F = (2, 2)$:
$(0|0) \to (1|1) \to (0|2) \to (1|1, 2) \to (2|1, 1, 2)$

Problem Statement

Given an *initial* thread state $t_I = (s_I, l_I) \in T$, a *final* thread state $t_F = (s_F, l_F)$ is reachable if there exists a finite sequence of configurations in V_n starting from $v_I = (s_I|l_1, \ldots l_n)$ where for all $i \in \{1, \ldots, n\}$, $l_i = l_I$, and ending at $v_F = (s_F|l'_1, \ldots, l'_n)$ where there exists $i \in \{1, \ldots, n\}$ such that $l'_i = l_F$.

Thread-State Equation(TSE)

A set of linear integer constraints that over-approximates the set of reachable configurations, inspired by [1]. $\mathcal{C}_L$ models the changes in the local states and $\mathcal{C}_S$ the synchronization imposed by the shared states between configurations. $\text{TSE} := \mathcal{C}_L \wedge \mathcal{C}_S$.

$$\mathcal{C}_L := (\mathbf{l_0} + \mathbf{cr} = \mathbf{l}) \wedge (\mathbf{l} \geq 0) \wedge (\mathbf{r} \geq 0) \wedge (\mathbf{l}(F) \geq 1)$$

- $\mathbf{c}$: $|L| \times |R|$ incidence matrix defined as

$$\mathbf{c}(i, r) = \begin{cases} 1 & \text{if } r = ((s, l_k), (s', l_i)) \in R \\ -1 & \text{if } r = ((s, l_i), (s', l_k)) \in R \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{l_0}, \mathbf{l}$: $|L|$ vectors of threads in the initial and final configuration respectively,
- $\mathbf{r}$: $|R|$ vector of transitions.

$$\mathcal{C}_S := \bigwedge_{s \in S} syn(s)$$

$$syn(s) := \begin{cases} \sum_{r \in inc(s)} \mathbf{r}(r) - \sum_{r \in out(s)} \mathbf{r}(r) = 0 & \text{if } s \notin \{s_I, s_F\} \\ \sum_{r \in inc(s)} \mathbf{r}(r) - \sum_{r \in out(s)} \mathbf{r}(r) = -1 & \text{if } s = s_I \neq s_F \\ \sum_{r \in inc(s)} \mathbf{r}(r) - \sum_{r \in out(s)} \mathbf{r}(r) = 1 & \text{if } s = s_F \neq s_I \\ \sum_{r \in inc(s)} \mathbf{r}(r) - \sum_{r \in out(s)} \mathbf{r}(r) = 0 & \text{if } s = s_F = s_I \end{cases}$$

Connectivity Constraints for TSE

- No guarantee that the transitions of the satisfying assignment of TSE give rise to a connected path from t_I to t_F.
- Define as *active* the transitions with a non-zero assignment
- Require that they form a path in the *shared-state projection* of (T, R).
- Augment TSE with

$$\mathcal{C}_{CON} := \bigwedge_{u \in \{1, \ldots, |S|\}} act(s_I) \wedge act(s_u) \implies \mathcal{P}(s_I, s_u)$$

where $act(s_i)$ enforces that shared state s_i has *at least* an adjacent *active* transition, and $P(s_i, s_j)$ enforces existence of a path between shared states s_i and s_j.

Witness Generation

- The set of constraints can still be satisfiable for safe target states.
- The connectivity constraints guarantee the existence of a path in the shared state projection, and the solution to TSE provides a candidate number of required threads $\bar{n}$.
- Good indicators that an explicit-state forward search(FWS) will reveal a reachability witness violating the safety property.
- If not the satisfying assignment is spurious, and at least $\bar{n}$ number of threads are required for a witness.

Algorithm

Require: TTS (T, R), final thread-state t_F
Ensure: "unreachable", or "reachable" + witness
  TSE $:= \mathcal{C}_L \wedge \mathcal{C}_S$
  if TSE $= UNSAT$ **then**
   return "unreachable"
  else
   $\text{TSE}_{CON} := \text{TSE} \wedge \mathcal{C}_{CON}$
  while $\text{TSE}_{CON} = SAT$ **do**
   $\bar{n} := threadCnt(\text{TSE}_{CON})$
   if $\text{FWS}((T, R), \bar{n}) = reachable$ **then**
    return "reachable" + witness path
   $\text{TSE}_{CON} := \text{TSE}_{CON} \wedge (n > n_m)$
  return "unreachable"

Evaluation Goals and Results

- Use a benchmark suite comprised of TTS originating from Boolean Programs and Petri Nets.
- Evaluate our method on both safe and unsafe instances.
- Identify how precise our incomplete method is.
- Measure the method's efficiency and compare it against other tools, both complete and incomplete.
  - TSE proves efficiently *all* safe TTS.
  - TSE proves efficiently *most of* safe Petri Nets.
  - TSE proves *most of* unsafe TTS with comparable execution time.
  - TSE proves efficiently *all* unsafe Petri Nets.

Instance	Safe	S	L	R	TSE	BFC	Petrinizer	MIST	IIC
Boop_simple2	○	129	201	7488	0.65	0.07	6.29	TO	TO
Function_Pointer3	○	9	2817	8960	1.21	840.54	45.7	5.16	TO
pthread5	○	513	131	17408	13.13	3.05	48.46	TO	TO
over_depth_2	○	266	1355	2397	3.94	13.96	44.62	TO	MO
conditionals2	●	5	209	280	0.03	13.88	0.01	0.03	2.01
double_lock_p2_3	●	1025	63	16448	1.12	MO	7.69	MO	MO
QRCU_4_5	●	33	203	2344	0.18	MO	2.53	5.36	3.27

[1] Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp Meyer, and Filip Niksic. An SMT-based approach to coverability analysis. In *CAV*, 2014.

Joint work with Peizun Liu and Thomas Wahl. Supported by NSF grant no. 1253331.