Infinite-state rlive

Christopher Johannsen

September 12, 2024

1 Preliminaries

We focus on model checking of infinite state systems. A system is defined as $S = \langle X, I, T\rangle$ where X is a set of variables, I is a one-state formula over X defining the initial condition and T is a two-state formula over X, X' defining the transition relation where $X' = \{x'\}_{x\in X}$ is a set of next-state variables. A trace $\pi = s_0, s_1, \ldots$ of the system S is a sequence of states such that $s_0 \models I$ and $s_i \wedge T \models s'_{i+1}$.

1.1 Invariant Model checking

The invariant model checking problem asks whether a property P holds in all reachable states of a system $S = \langle I, T\rangle$. A procedure $\mathsf{model\text{-}check}(I, T, P)$ will return a finite trace $\pi = s_0, s_1, \ldots, s_n$ of S where $s_n \models \neg P$, or an inductive invariant inv showing that $inv \models P$, $I \models inv$, and $inv \wedge T \models inv'$.

1.2 Abstract Invariant Model Checking

Abstract model checking takes a concrete system and a set of predicates $\mathbb{P}$ then returns either the result *unsafe* and an abstract trace $\widehat{\pi} = \widehat{s_0}, \widehat{s_1}, \ldots$ over $\mathbb{P}$ that violates the property or the result *safe* and an inductive invariant $\widehat{inv}$ over $\mathbb{P}$.

We assume access to a procedure $\mathsf{model\text{-}check\text{-}abstract}(I, T, P, \mathbb{P})$ that returns an abstract trace $\widehat{\pi}$ in case of violation of P or an inductive invariant inv over $\mathbb{P}$ otherwise.

1.3 Liveness Checking

We focus on properties of the form FGq, i.e., liveness properties. The problem under consideration is whether some system S satisfies a property $P = FGq$, in other words, we compute $S \models FGq$. When S is finite, $S \not\models FGq$ if and only if there exists a loop $\alpha \cdot \beta^\omega$ in S where some state $b \in \beta$ is such that $b \models \neg q$. If S is infinite, then the existence of a loop $\alpha \cdot \beta^\omega$ proves that $S \not\models FGq$, but there may be other non-loop counterexamples.

1.4 rlive

One algorithm for checking liveness properties is rlive. As opposed to other algorithms (Liveness-to-safety, FAIR, k-liveness, k-FAIR), rlive performs a depth-first search for a loop that violates FGq, i.e., one that satisfies $GF\neg q$.

The algorithm incrementally searches for a path that satisfies $\neg q$ infinitely often via a series of invariant model checking queries. If the algorithm determines that $\neg q$ is unreachable from the current state in the search, rlive adds the newly-found inductive invariant to a set of states known as *shoals*, otherwise it reaches another state s that satisfies $\neg q$. If s has already been seen, then a loop satisfying $GF\neg q$ as been found, so rlive terminates with *unsafe*. Otherwise rlive adds this state to its stack and continues the search. The algorithm terminates with *safe* once it shows that the initial set of states is in the *shoals*. We present a version of the algorithm in Figure 1.

Algorithm 1: rlive algorithm

```
Procedure rlive($I$, $T$, $FGq$, $C_{init}$) begin
    $C := C_{init}$ // initial shoals
    $B :=$ empty stack of states
    while model-check-concrete($I, T \wedge (\neg C \wedge \neg C'), \neg q \wedge T^{-1}(\neg C)$) is unsafe do
        $s :=$ final state of get-mc-cex()
        $B.push(s)$
        while $B$ is not empty do
            $s := B.top()$
            if $s \in B$ then
                return (Unsafe, $C$)
            if model-check-concrete($T(s), T \wedge (\neg C \wedge \neg C'), \neg q \wedge T^{-1}(\neg C)$) is unsafe then
                $t :=$ final state of get-mc-cex()
                $B.push(t)$
            else
                $inv :=$ get-mc-inv()
                $C := C \vee inv$
                $B.pop()$
    return Safe
```

2 Algorithms

Here we give an overview of the proposed algorithms.

Alg.	Model Abstraction	Refinement	Predicates
Basic	N/A	N/A	N/A
EA-LR	Explicit	Lazy	Global
IA-LR	Implicit	Lazy	Global
IA-ER	Implicit	Eager	Global
IA-ER*	Implicit	Eager	Local

Basic runs rlive as it is, except over the infinite state system. Functionally this means that a model checker will use SMT queries rather than SAT ones.

EA-LR constructs an abstract system $\widehat{S} = \langle \widehat{I}, \widehat{T} \rangle$ explicitly over a fixed set of predicates $\mathbb{P}$, using techniques like ALLSAT, then standard rlive on the abstract system. If the query returns safe or a feasible counterexample, the corresponding result is returned. If the query returns an infeasible counterexample, the set of predicates is refined to eliminate this counterexample, the process is repeated with the newly refined set of predicates.

IA-LR is the same as Algorithm 1 except that it uses implicit abstraction to avoid explicitly computing the abstract model.

IA-ER performs model checking on the abstract state space using implicit abstraction, but checks that each resulting abstract counterexample is feasible.

IA-ER* uses a local set of predicates for each model checking query. In other words, the model checker returns both a counterexample and an updated set of predicates in the case of a violation, and the returned set of predicates is used as the initial set of predicates for the next model checking query.

Algorithm 2: Explicit abstraction, lazy refinement

```
Procedure rlive_EA-LR($I$, $T$, $FGq$) begin
    $\mathbb{P}$ := predicate symbols in $I$, $q$
    $R_T, \widehat{R_T} := true$ // formula for $T$ refinement
    $\widehat{I} := \mathsf{abstract}(I, \mathbb{P})$, $\widehat{T} := \mathsf{abstract}(T, \mathbb{P})$, $\widehat{q} := \mathsf{abstract}(q, \mathbb{P})$
    $C := false$ // initialize shoals
    while $\mathit{rlive}(\widehat{I}, \widehat{T} \wedge \widehat{R_T}, FG\widehat{q}, C)$ is unsafe do
        $C := \mathsf{get\text{-}rlive\text{-}shoals}()$
        $\widehat{\pi_\alpha}, \widehat{\pi_\beta} := \mathsf{get\text{-}rlive\text{-}cex}()$
        if $\mathit{feasible}(T \wedge R_T, \mathbb{P}, \widehat{\pi_\alpha})$ and $\mathit{feasible\text{-}loop}(T, \mathbb{P}, \widehat{\pi_\beta})$ then
            return Unsafe
        $R_T, \mathbb{P} := \mathsf{refine}(T, \mathbb{P}, \widehat{\pi})$
        $\widehat{I} := \mathsf{abstract}(I, \mathbb{P})$, $\widehat{T} := \mathsf{abstract}(T, \mathbb{P})$,
        $\widehat{q} := \mathsf{abstract}(q, \mathbb{P})$, $\widehat{R_T} := \mathsf{abstract}(R_T, \mathbb{P})$
    return Safe
```

The algorithms assume access to a procedure model-check(I,T,P) that performs model checking for property P and returns a concrete counterexample in the case of violation, as well as model-check$_\alpha$(I,T,P) that returns an abstract counterexample. model-check$_\alpha$ can also optionally take a set of initial predicates as in model-check$_\alpha$(I,T,P,$\mathbb{P}$). The procedure abstract(ϕ, $\mathbb{P}$) explicitly constructs an abstract formula $\widehat{\phi}$ using the predicates in $\mathbb{P}$. get-rlive-cex returns the counterexample from rlive separated into its prefix and lasso portions.

This is similar to the previous algorithm except we do *eager refinement*, i.e., we check that each path segment of the loop is feasible before continuing. If any path segment is infeasible, we return to the CEGAR loop and refine. Otherwise, we continue until we find an abstract loop of the form:

$$\widehat{\pi_0} \cdot \widehat{\pi_1} \cdots \widehat{\pi_i} \cdots \widehat{\pi_j}$$

where the first state of $\widehat{\pi_i}$ is the same as the last state of $\widehat{\pi_j}$: $\widehat{\pi_i}[0] = \widehat{\pi_j}[-1]$. The algorithm maintains the invariant that $\mathsf{feasible}(I, T, \mathbb{P}, \widehat{\pi_k})$ is true for all path segments $k \in 0..j$, but does not guarantee that the interfaces between them are feasible. In other words,

$$\mathsf{feasible}(I, T, \mathbb{P}, \widehat{\pi_k}) \wedge \mathsf{feasible}(I, T, \mathbb{P}, \widehat{\pi_{k+1}}) \not\Rightarrow \mathsf{feasible}(I, T, \mathbb{P}, \widehat{\pi_k} \cdot \widehat{\pi_{k+1}}).$$

However, the feasible check on line 21 will check this.

To solve this, we guarantee the feasibility of the full abstract trace found so far during the search and check $\mathsf{feasible}(I, T, \mathbb{P}, \widehat{\pi_0} \cdots \widehat{\pi_k})$ for each newly found path segment $\widehat{\pi_k}$.

3 Abstract Counterexample Refinement

There are four outcomes to consider given a system S. Let $\widehat{\pi} = \alpha \cdot \beta^\omega$ be a candidate abstract counterexample.

1. $\exists \pi : \pi = \mathsf{concretize}(S, \widehat{\pi})$, return such a π.

2. $\nexists \pi : \pi = \mathsf{concretize}(S, \widehat{\pi})$, refine predicates to block $\widehat{\pi}$. Could use interpolants ala ic3ia [1].

3. $\forall k : \exists \pi : \pi = \mathsf{concretize}(S, \alpha \cdot \beta^k)$ and $\nexists \pi : \pi = \mathsf{concretize}(S, \alpha \cdot \beta^\omega)$, synthesize a ranking function and use this to block $\widehat{\pi}$. We may consider using the techniques presented in [2].

4. There exists a non-lasso-shaped counterexample, use well-founded funnels to find these.

Algorithm 3: Implicit abstraction, lazy refinement

```
Procedure rlive_{IA-LR}($I$, $T$, $FGq$) begin
    $\mathbb{P} :=$ predicate symbols in $I$, $q$
    $R_T := true$ // formula for $T$ refinement
    $C := false$ // initialize shoals
    while $\mathit{rlive\text{-}rec}_{IA\text{-}LR}(I, T \wedge R_T, FGq, \mathbb{P}, C)$ is unsafe do
        $\widehat{\pi_\alpha}, \widehat{\pi_\beta} :=$ get-rlive-cex()
        if $\mathit{feasible}(T \wedge R_T, \mathbb{P}, \widehat{\pi_\alpha})$ and $\mathit{feasible\text{-}loop}(T, \mathbb{P}, \widehat{\pi_\beta})$ then
            return Unsafe
        $R_T, \mathbb{P} :=$ refine$(I, T \wedge R_T, \mathbb{P}, \widehat{\pi})$
        $C :=$ get-rlive-shoals()
    return Safe

Procedure rlive-rec_{IA-LR}($I$, $T$, $FGq$, $C$) begin
    $\widehat{B} :=$ empty stack of abstract states
    while $\mathit{model\text{-}check\text{-}abstract}(I, T \wedge (\neg C \wedge \neg C'), \neg q \wedge T^{-1}(\neg C), \mathbb{P})$ is unsafe do
        $\widehat{s} :=$ final state of get-abs-mc-cex()
        $\widehat{B}.push(\widehat{s})$
        while $\widehat{B}$ is not empty do
            $\widehat{s} := \widehat{B}.top()$
            if $\widehat{s} \in \widehat{B}$ then
                return (Unsafe, $C$)
            if $\mathit{model\text{-}check\text{-}abstract}(\mathit{concretize}(\widehat{T}(\widehat{s})), T \wedge (\neg C \wedge \neg C'), \neg q \wedge T^{-1}(\neg C), \mathbb{P})$ is unsafe then
                $\widehat{t} :=$ final state of get-abs-mc-cex()
                $\widehat{B}.push(\widehat{t})$
            else
                $\widehat{B}.pop()$
                $inv :=$ concretize(get-mc-inv())
                $C := C \vee inv$
    return Safe
```

The algorithms outlines previously only cover cases 1 and 2, but can be outfitted to handle the other cases. For case 3, if we can synthesize such a ranking function, we must refine the system to block the abstract loop in the next model checking calls. Let $\widehat{\pi_0} \cdot \widehat{\pi_1} \cdots \widehat{\pi_i}$ be an abstract loop made up of path segments such that there exists a ranking function R that includes every state in the loop. Is there a naive approach to start from?

3.1 Example

We consider an example system to illustrate how to use ranking functions to block spurious counterexamples in rlive.

$$
\begin{aligned}
V &= \{x, n\} \\
I &:= x = 0 \wedge n > 0 \\
T &:= x' = x + 1 \wedge n' = n \\
P &:= GF(x = n)
\end{aligned}
$$

In a naive version of the algorithm (Algorithm 1), the call to model-check-concrete will find

A ranking function that can be used to prove $F(x = n)$ is $R(V) = n - x$.

Algorithm 4: Implicit abstraction, eager refinement

```
1  Procedure rlive$_{\text{IA-ER}}$($I$, $T$, $FGq$) begin
2      $\mathbb{P}$ := predicate symbols in $I$, $q$
3      while rlive-3-rec($I, T, FGq, \mathbb{P}$) is infeasible do
4          $\widehat{\Pi}$ := infeasible counterexample from rlive-3-rec
5          $I, T, \mathbb{P}$ := refine($T, \mathbb{P}, \widehat{\Pi}$)
6      return Result from rlive-3-rec
7
8  Procedure rlive-rec$_{\text{IA-ER}}$($I, T, FGq, \mathbb{P}$) begin
9      $C := false$ // Shoals
10     $\widehat{B}$ := empty stack of states
11     $\widehat{\Pi} := \emptyset$ // empty trace
12     while model-check-abstract($I, T \wedge (\neg C \wedge \neg C'), \neg q \wedge T^{-1}(\neg C), \mathbb{P}$) is unsafe do
13         $\widehat{\pi}$ := get-abs-mc-cex()
14         if $\neg$feasible($T \wedge (\neg C \wedge \neg C'), \mathbb{P}, \widehat{\pi}$) then
15             return Infeasible $\widehat{\pi}$
16         $\widehat{s}$ := final state of $\widehat{\pi}$
17         $\widehat{B}.push(\widehat{s})$
18         while $\widehat{B}$ is not empty do
19             $\widehat{s} := \widehat{B}.top()$
20             if $\widehat{s} \in \widehat{B}$ then
21                 $\widehat{\Pi_\alpha} := \widehat{\Pi}$ up to and including $\widehat{s}$
22                 $\widehat{\Pi_\beta} := \widehat{\Pi}$ including and after $\widehat{s}$
23                 if feasible($T \wedge (\neg C \wedge \neg C'), \mathbb{P}, \widehat{\Pi_\alpha}$) and feasible-loop($T \wedge (\neg C \wedge \neg C'), \mathbb{P}, \widehat{\Pi_\beta}$) then
24                     return Unsafe
25                 else
26                     return Infeasible $\widehat{\Pi}$
27             if model-check-abstract($T(s), T \wedge (\neg C \wedge \neg C'), \neg q \wedge T^{-1}(\neg C), \mathbb{P}$) is unsafe then
28                 $\widehat{\pi}$ := get-abs-mc-cex()
29                 if $\neg$feasible($T \wedge (\neg C \wedge \neg C'), \mathbb{P}, \widehat{\Pi} \cdot \widehat{\pi}$) then
30                     return Infeasible $\widehat{\Pi}$
31                 $\widehat{\Pi} := \widehat{\Pi} \cdot \widehat{\pi}$
32                 $\widehat{t}$ := final state of $\widehat{\pi}$
33                 $\widehat{B}.push(\widehat{t})$
34             else
35                 $inv$ := invariant from model-check-abstract
36                 $C := C \vee inv$
37                 $\widehat{B}.pop()$
38     return Safe
```

Algorithm 5: Feasibility checking

```
Procedure feasible($T, \mathbb{P}, \widehat{\pi}$) begin
    $\widehat{s_0}, \ldots, \widehat{s_n} := \widehat{\pi}$
    if $\bigwedge_{0\leq i<n} T(X^i, X^{i+1}) \wedge \bigwedge_{0<i\leq n} \widehat{s_i}(X^i_{\mathbb{P}})[X^i_{\mathbb{P}} \mapsto \mathbb{P}^i]$ is unsat then
        // Compute sequence interpolant (Section 3.3 [1])
        return New set of predicates
    else
        return Satisfiable assignment to if condition

Procedure feasible-loop($T, \mathbb{P}, \widehat{\pi}$) begin
    $\widehat{s_0}, \ldots, \widehat{s_n} := \widehat{\pi}$
    if $\bigwedge_{0\leq i<k} T(X^i, X^{i+1}) \wedge \bigwedge_{0<i\leq k} \widehat{s_i}(X^i_{\mathbb{P}})[X^i_{\mathbb{P}} \mapsto \mathbb{P}^i]$ is unsat then
        // Compute sequence interpolant (Section 3.3 [1])
        return New set of predicates
    else
        $R :=$ synth-rank-function($T, \widehat{\pi}$) // Using technique on p11 of [3]
        // How to use $R$ to block $\widehat{\pi}$ in $T$?
        // For now, assume we find some formula $R_T$ such that $\widehat{T} \wedge \widehat{R_T}$ does not admit $\widehat{\pi}$
        return $R_T$
```

References

[1] Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. Infinite-state invariant checking with ic3 and predicate abstraction. *Formal Methods in System Design*, 49:190–218, 2016.

[2] Byron Cook, Andreas Podelski, and Andrey Rybalchenko. Abstraction refinement for termination. In *Static Analysis: 12th International Symposium, SAS 2005, London, UK, September 7-9, 2005. Proceedings 12*, pages 87–101. Springer, 2005.

[3] Jakub Daniel, Alessandro Cimatti, Alberto Griggio, Stefano Tonetta, and Sergio Mover. Infinite-state liveness-to-safety via implicit abstraction and well-founded relations. In *International Conference on Computer Aided Verification*, pages 271–291. Springer, 2016.