Proving the existence of fair paths in infinite-state systems

Alessandro Cimatti, Alberto Griggio, Enrico Magnago

Fondazione Bruno Kessler

FONDAZIONE
BRUNO KESSLER

Context

Problem
Does a transition system admit at least one fair path?
(Counterexample to liveness property).

- Undecidable in infinite-state systems.
- Techniques to prove the language empty (property holds) and techniques to prove the existence of a fair path (witness).
- Witnesses are often limited to lasso-shaped paths.
- Not sufficient in infinite-state, need to look for witnesses with different shapes.

How can we represent them?

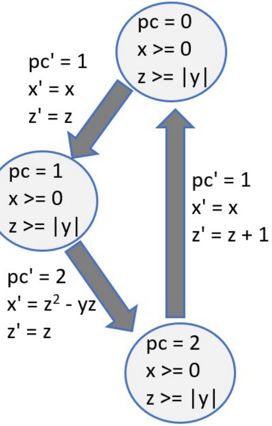
$\mathcal{R}$-abstraction

Assume we want to prove the existence of a non-terminating run for the code below.

0:	**while** $x \geq 0$ **do**
1:	$\quad x = z^2 - z * y$
2:	$\quad z = z + 1$
3:	**end while**

```
TRANS
(pc = -1 -> next(pc) = -1) &
(pc = 0 & x < 0 -> next(pc) = -1) &
(pc = 0 & x >= 0 ->
          next(pc) = 1 & next(x) = x &
          next(y) = y & next(z) = z) &
(pc = 1 -> next(pc) = 2 &
           next(x) = z*z - y*z &
           next(y) = y & next(z) = z) &
(pc = 2 -> next(pc) = 0 & next(x) = x &
           next(y) = y & next(z) = z + 1);
FAIRNESS pc != -1;
```

$\mathcal{R}$-abstraction: reachable, non-empty underapproximation with only fair paths.



$\mathcal{R}$-abstraction

Assume we want to prove the existence of a non-terminating run for the code below.

$\mathcal{R}$-abstraction: reachable, non-empty underapproximation with only fair paths.

0: **while** $x \geq 0$

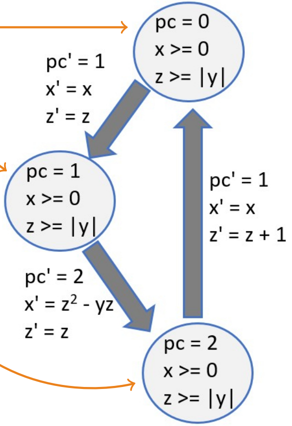
1: $x = z^2 -$

2: $z = z + 1$

3: **end while**

$\mathcal{R}$-abstraction has a sequence of regions one of which contains only fair states.

```
TRANS
(pc = -1 -> next(pc) = -1) &
(pc = 0 & x < 0 -> next(pc) = -1) &
(pc = 0 & x >= 0 ->
          next(pc) = 1 & next(x) = x &
          next(y) = y & next(z) = z) &
(pc = 1 -> next(pc) = 2 &
         next(x) = z*z - y*z &
         next(y) = y & next(z) = z) &
(pc = 2 -> next(pc) = 0 & next(x) = x &
         next(y) = y & next(z) = z + 1);
FAIRNESS pc != -1;
```



$\mathcal{R}$-abstraction

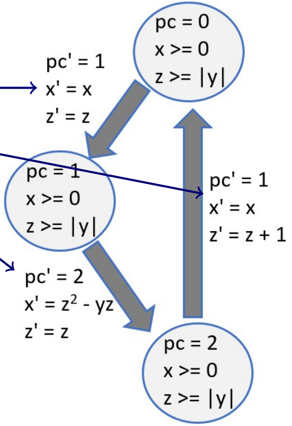
Assume we want to prove the existence of a non-terminating run for the code below.

0: **while** $x \geq 0$ **do**
1: $\quad x = z^2 -$
2: $\quad z = z + 1$
3: **end while**

Transition relation can allow bounded dwell on a region and must eventually lead to the fair one.

```
TRANS
(pc = −1 −> next(pc
(pc = 0 & x < 0 −> next(pc) = −1) &
(pc = 0 & x >= 0 −>
          next(pc) = 1 & next(x) = x &
          next(y) = y & next(z) = z) &
(pc = 1 −> next(pc) = 2 &
           next(x) = z∗z − y∗z &
           next(y) = y & next(z) = z) &
(pc = 2 −> next(pc) = 0 & next(x) = x &
           next(y) = y & next(z) = z + 1);
FAIRNESS pc != −1;
```

$\mathcal{R}$-abstraction: reachable, non-empty underapproximation with only fair paths.



$\mathcal{R}$-abstraction

Assume we want to prove the existence of a non-terminating run for the code below.

0: **while** $x \geq 0$ **do**

1: $\quad x = z^2 - z * y$

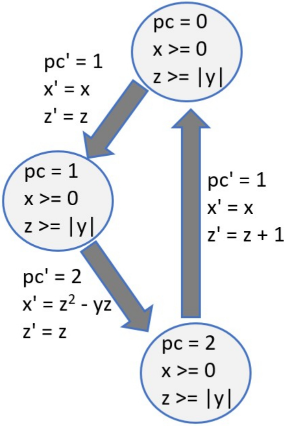
2: $\quad z = z +$

3: **end while**

Can be seen as a generalisation of closed recurrence sets to deals with fairness.

```
TRANS
(pc = -1 -> next(pc) = -1) &
(pc = 0 & x < 0 -> next(pc) = -1) &
(pc = 0 & x >= 0 ->
          next(pc) = 1 & next(x) = x &
          next(y) = y & next(z) = z) &
(pc = 1 -> next(pc) = 2 &
           next(x) = z*z - y*z &
           next(y) = y & next(z) = z) &
(pc = 2 -> next(pc) = 0 & next(x) = x &
           next(y) = y & next(z) = z + 1);
FAIRNESS pc != -1;
```

$\mathcal{R}$-abstraction: reachable, non-empty underapproximation with only fair paths.



Compositional approach

Identify $\mathcal{R}$-abstraction compositionally

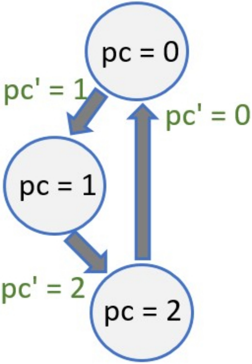
Look for $\mathcal{R}$-abstraction that can be obtained as composition of $\mathcal{AG}$-skeletons, each $\mathcal{AG}$-skeleton is responsible for a set of symbols.

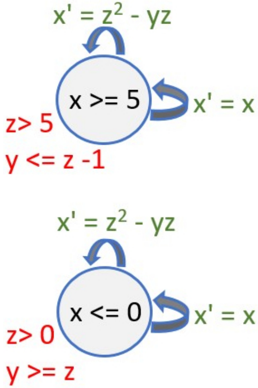
We propose a procedure that given a set of $\mathcal{AG}$-skeletons searches for a composition of a subset of them that is an $\mathcal{R}$-abstraction for the system.

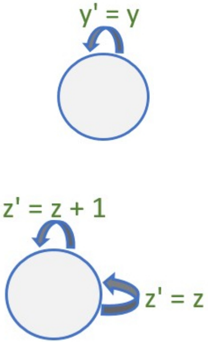
$\mathcal{AG}$-skeletons

```
while x ≥ 0 do
    x = z^2 − z ∗ y
    z = z + 1
end while
```

Each $\mathcal{AG}$-skeleton has a set of regions.
Each region has an invariant and an assumption.
The transition relation must ensure the invariants hold and provides the next assignment constraints for a subset of the symbols.



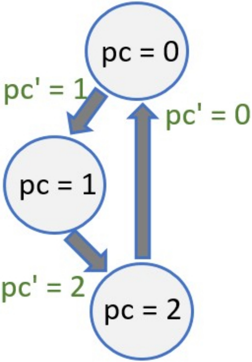


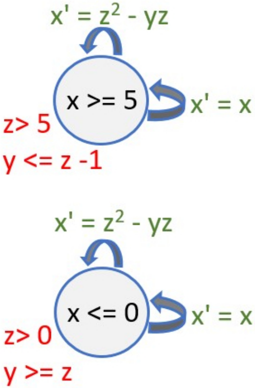


$\mathcal{AG}$-skeletons

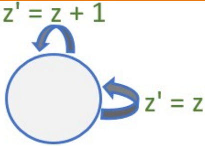
while $x \geq 0$ **do**
    $x = z^2 - z * y$
    $z = z + 1$
end while

Each $\mathcal{AG}$-skeleton has a set of regions.
Each region has an invariant and an assumption.
The transition relation must ensure the invariants hold and provides the next assignment constraints for a subset of the symbols.



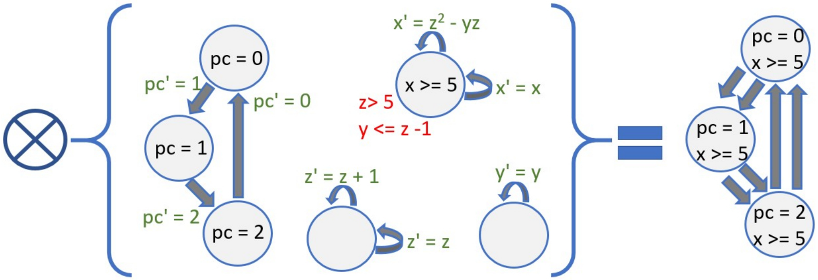


If the transition relation maps a state in region i into a state in region j, then every state in region i must have a successor in region j.



$\mathcal{AG}$-skeleton composition

Composition operator: synchronous product between $\mathcal{AG}$-skeletons such that the assumptions are met.
Composition of $\mathcal{AG}$-skeletons is still a $\mathcal{AG}$-skeleton.



Composition search

Objective
Find a reachable composition ($\mathcal{AG}$-skeleton) with a loop over the regions, one of which is fair, such that each transition underapproximates the transition relation of the original system. Such loop over the region is our $\mathcal{R}$-abstraction.

$\mathcal{R}$-abstraction search

Algorithm FIND-COMPOSITION(M, $\mathcal{H}$)

```
H ← FILTER-INCORRECT-HINTS(H)
constr ← ⊤
bad ← ⊥
while true do
    constr ← constr ∧ ¬bad
    prob ← GET-REACHABILITY-PROBLEM(H, M, constr)
    trace ← CHECK-REACHABILTY(prob)
    if trace = ∅ then
        return ∅
    end if
    comp ← COMPOSITION-FROM-TRACE(trace, H)
    bad ← CHECK-ASSUMPTIONS(comp)
    if bad = ⊥ then
        return comp
    end if
end while
```

$\mathcal{R}$-abstraction search

Algorithm $\textsc{find-composition}(M, \mathcal{H})$

$\mathcal{H} \leftarrow \textsc{filter-incorrect-hints}(\mathcal{H})$
$constr \leftarrow \top$
$bad \leftarrow \bot$
while true **do**
  $constr \leftarrow constr \land \neg bad$
  $prob \leftarrow \textsc{get-reachability-problem}(\mathcal{H}, M, constr)$
  $trace \leftarrow \textsc{check-reachabilty}(prob)$
  if $trace = \emptyset$ **then**
    return $\emptyset$
  end if
  $comp \leftarrow \textsc{composition-from-trace}(trace, \mathcal{H})$
  $bad \leftarrow \textsc{check-assumptions}(comp)$
  if $bad = \bot$ **then**
    return $comp$
  end if
end while

User provides the set of $\mathcal{AG}$-skeletons $\mathcal{H}$ and the fair transition system M

$\mathcal{R}$-abstraction search

Algorithm $\textsc{find-composition}(M, \mathcal{H})$

$\mathcal{H} \leftarrow \textsc{filter-incorrect-hints}(\mathcal{H})$

Check correctness of the input

$constr \leftarrow \top$
$bad \leftarrow \bot$
while true **do**
  $constr \leftarrow constr \wedge \neg bad$
  $prob \leftarrow \textsc{get-reachability-problem}(\mathcal{H}, M, constr)$
  $trace \leftarrow \textsc{check-reachabilty}(prob)$
  if $trace = \emptyset$ **then**
    return $\emptyset$
  end if
  $comp \leftarrow \textsc{composition-from-trace}(trace, \mathcal{H})$
  $bad \leftarrow \textsc{check-assumptions}(comp)$
  if $bad = \bot$ **then**
    return $comp$
  end if
end while

Algorithm FIND-COMPOSITION$(M, \mathcal{H})$

$\mathcal{H} \leftarrow$ FILTER-INCORRECT-HINTS$(\mathcal{H})$
$constr \leftarrow \top$
$bad \leftarrow \bot$
while true **do**
  $constr \leftarrow constr \wedge \neg bad$
  $prob \leftarrow$ GET-REACHABILITY-PROBLEM$(\mathcal{H}, M, constr)$
  $trace \leftarrow$ CHECK-REACHABILTY$(prob)$
  if $trace = \emptyset$ **then**
    return $\emptyset$
  end if
  $comp \leftarrow$ COMPOSITION-FROM-TRACE$(trac$
  $bad \leftarrow$ CHECK-ASSUMPTIONS$(comp)$
  if $bad = \bot$ **then**
    return $comp$
  end if
end while

Encode search problem into reachability: find candidate reachable composition with a fair region and that under-approximates M.

$\mathcal{R}$-abstraction search

Algorithm FIND-COMPOSITION(M, $\mathcal{H}$)

$\mathcal{H} \leftarrow$ FILTER-INCORRECT-HINTS($\mathcal{H}$)
$constr \leftarrow \top$
$bad \leftarrow \bot$
while true **do**
    $constr \leftarrow constr \wedge \neg bad$
    $prob \leftarrow$ GET-REACHABILITY-PROBLEM($\mathcal{H}, M, constr$)
    $trace \leftarrow$ CHECK-REACHABILTY($prob$)
    if $trace = \emptyset$ **then**
        return $\emptyset$
    end if
    $comp \leftarrow$ COMPOSITION-FROM-TRACE($trace, \mathcal{H}$)
    $bad \leftarrow$ CHECK-ASSUMPTIONS($comp$)
    if $bad = \bot$ **then**
        return $comp$
    end if
end while

Find path using a model checker (NUXMV)

$\mathcal{R}$-abstraction search

Algorithm $\textsc{find-composition}(M, \mathcal{H})$

$\mathcal{H} \leftarrow \textsc{filter-incorrect-hints}(\mathcal{H})$
$constr \leftarrow \top$
$bad \leftarrow \bot$
while true **do**
    $constr \leftarrow constr \wedge \neg bad$
    $prob \leftarrow \textsc{get-reachability-problem}(\mathcal{H}, M, constr)$
    $trace \leftarrow \textsc{check-reachabilty}(prob)$
    if $trace = \emptyset$ **then**
        return $\emptyset$
    end if
    $comp \leftarrow \textsc{composition-from-trace}(trace, \mathcal{H})$
    $bad \leftarrow \textsc{check-assumptions}(comp)$
    if $bad = \bot$ **then**
        return $comp$
    end if
end while

Check if all assumptions are met. If not learn incompatible set of regions and transitions.

Timed/Hybrid systems: diverging ‘time’

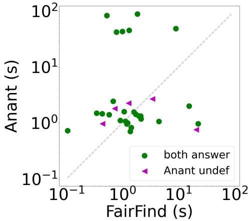
In timed and hybrid systems we want to consider only the infinite paths in which ‘time’ diverges.

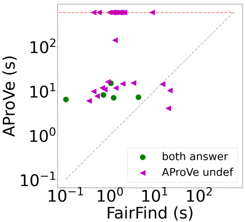
We show two ways to ensure that an $\mathcal{R}$-abstractions has only non-zeno paths:

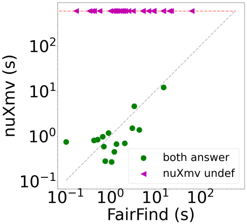
1. if a symbol diverges in some $\mathcal{AG}$-skeleton, then it diverges also in all its compositions; proof local to the $\mathcal{AG}$-skeleton;
2. provide a set of sufficient conditions under which it is possible to shrink the language of an $\mathcal{AG}$-skeleton or $\mathcal{R}$-abstraction to rule out all zeno paths without making its language empty.

Experimental evaluation

Comparison with automated tools.
Anant and AProVE on 31 non-linear software benchmarks[1], NUXMV on the software benchmarks, 3 infinite-state systems and 9 hybrid systems.



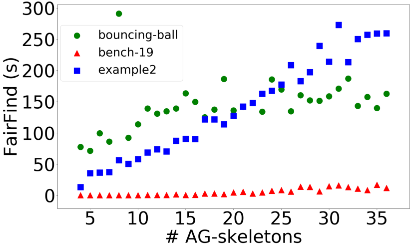




[1] non-linear software benchmarks taken from "Disproving termination with overapproximation", Byron Cook, Carsten Fuhs, Kaustubh Nimkar, Peter W. O'Hearn, FMCAD 2014

Experimental evaluation: increasing number of $\mathcal{AG}$-skeletons

How does the number of $\mathcal{AG}$-skeletons affect the time required to identify a $\mathcal{R}$-abstraction?



where *bench-19* is a non-linear software benchmark, *example2* is a non-linear infinite-state system and *bouncing-ball* is an hybrid system.

Conclusions

Summary

- representation of fair paths via $\mathcal{R}$-abstraction;
- automated search of $\mathcal{R}$-abstraction as composition of $\mathcal{AG}$-skeletons via reduction to reachability;
- prove non-zenoness locally to the $\mathcal{AG}$-skeleton when possible, otherwise shrink language of resulting $\mathcal{R}$-abstraction.

Future work

- automated synthesis of $\mathcal{AG}$-skeletons;
- allow synthesis of $\mathcal{R}$-abstractions without fixed bound on dwell time;
- automated proof of non-zenoness.

The End

Thank you for your attention, questions?

$\mathcal{R}$-abstraction [1/3]

Let $M \doteq \langle S_M, I_M(S_M), T_M(S_M, S_M{}'), F_M(S_M) \rangle$ be a fair transition system. A transition system $A \doteq \langle S_A, I_A(S_A), T_A(S_A, S'_A) \rangle$ is an $\mathcal{R}$-abstraction of M with respect to a list of formulae $\mathcal{R}(S_A) \doteq [R_0(S_A), \ldots, R_{m-1}(S_A)]$, also called regions, iff the following hold:

0. $S_M \subseteq S_A$,
1. There exists some initial state in M from which it is possible to reach an initial state of A, for some assignment to the $S_A \setminus S_M$:
$$M \not\models \mathbf{AG} \neg I_A(S_A)$$
2. The set of initial states of A is a subset of the union of the regions:
$$A \models \mathcal{R}(S_A)$$

$\mathcal{R}$-abstraction [2/3]

2. The transition relation of A underapproximates the transition relation of M:

$$\mathcal{R}(S_A) \wedge T_A(S_A, S'_A) \models T_M(S_M, S'_M)$$

3. Every state in R_0, projected over the symbols in S_M corresponds to a fair state of M:

$$A \models \mathbf{AG}(R_0(S_A) \to F_M(S_M))$$

4. Every reachable state in A has at least one successor via its transition relation T_A:

$$A \models \mathbf{AGEX}\top$$

$\mathcal{R}$-abstraction [3/3]

5. For each region $R_i \in \mathcal{R}$, with $i > 0$, every state in R_i can remain in such region at most a finite number of steps and must eventually reach a region with a lower index $j < i$:

$$A \models \bigwedge_{i=1}^{m-1} \mathbf{AG}(R_i \rightarrow \mathbf{A}[R_i \mathbf{U} \bigvee_{j=0}^{i-1} R_j])$$

6. All states reachable in one step from R_0 are in $\mathcal{R}$:

$$A \models \mathbf{AG}(R_0 \rightarrow \mathbf{AX} \bigvee_{i=0}^{m-1} R_i)$$

$\mathcal{AG}$-skeleton

- Describe evolution of a subset of the symbols S^j over a sequence of regions;
- each region R_i^j has some assumption A_i^j on the other symbols;
- there is a transition between two regions iff every state in the first one has at least one successor in the second one.

$$\begin{array}{l} \forall i, i' : 0 \leq i < m^j \wedge 0 \leq i' < m^j \rightarrow \\ \quad \exists S, S' : (R_i^j(S) \wedge A_i^j(S^{\neq j}) \wedge T^j(S, S') \wedge R_{i'}^j(S') \wedge A_{i'}^j(S^{\neq j'})) \quad \models \\ \quad \forall S \exists S^{j'} \forall S^{\neq j'} : R_i^j(S) \wedge A_i^j(S^{\neq j}) \wedge A_{i'}^j(S^{\neq j'}) \rightarrow R_{i'}^j(S') \wedge T^j(S, S') \end{array}$$

where m^j is the number of regions and $S^{\neq j} \doteq S \setminus S^j$.

SMV encoding

We define, for each regions $R_i^j, R_{i'}^j \in \mathcal{R}^j$,

$$eval(\textsc{isT}(f_k^H, i)) := \begin{cases} \top \text{ if } R_i^j \wedge A_i^j \models p_k^F \\ \bot \text{ if } R_i^j \wedge A_i^j \models \neg p_k^F \\ ? \text{ otherwise} \end{cases}$$

$$eval(\textsc{isT}(t_k^H, i, i')) := \begin{cases} \top \text{ if } R_i^j \wedge A_i^j \wedge T^j \wedge \\ \qquad R_{i'}^j \wedge A_{i'}^j \models p_k^F \\ \bot \text{ if } R_i^j \wedge A_i^j \wedge T^j \wedge \\ \qquad R_{i'}^j \wedge A_{i'}^j \models \neg p_k^F \\ ? \text{ otherwise} \end{cases}$$