Generalized Counterexamples to Liveness Properties

Gadi Aleksandrowicz
Jason Baumgartner
Alexander Ivrii
Ziv Nevo

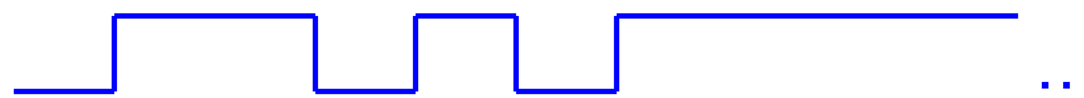
IBM

Outline

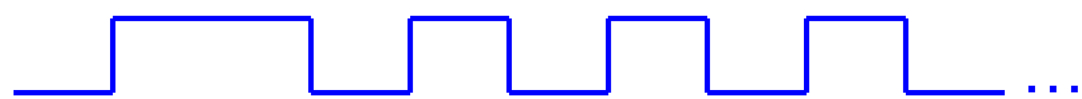
- Generalized counterexamples to liveness
  - and why they are especially interesting
- How to detect that a trace exhibits a liveness CEX
  - and how to manipulate traces to increase this likelihood
- k-LIVENESS with failure detection
- Conclusions

Liveness Properties

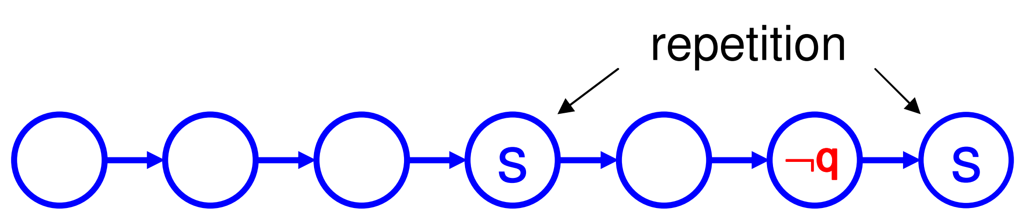
- Reduce to the form FGq (with q a state variable)
- FGq passes:
  - on every trace q eventually becomes true forever



- FGq fails:
  - there is a trace on which ¬q holds infinitely often



  - equivalently, there is a finite trace with a repeating state, and ¬q in-between



Example

- (q, x, y) – state variables
  - initially: $q = 1, x = 0, y = 0$
  - next-state: $q' = (q \wedge x) \vee (\neg q \wedge y)$, $x' = q \wedge y$, $y' = \neg x$

- There is a concrete counterexample to FGq of length 4:

  repetition

  - $(1, 0, 0) \rightarrow (0, 0, 1) \rightarrow (1, 0, 1) \rightarrow (0, 1, 1) \rightarrow (1, 0, 0)$

- There is a "generalized" counterexample to FGq of length 2:

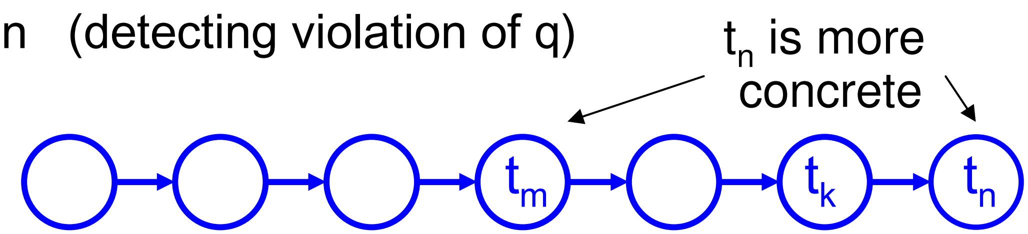
  repetition

  - $(1, 0, \cdot) \rightarrow (0, \cdot, 1) \rightarrow (1, 0, \cdot)$

Generalized CEXes

- generalized state: a partial assignment to state variables
- s is a generalized predecessor of t:

  for every state in s, there is a transition to some state t
- $t_0, t_1, \ldots, t_n$ generalized trace:
  - t_0 contains a state in Init
  - t_i is a generalized predecessor of t_{i+1} for every i, $0 \leq i < n$
- generalized counterexample to FGq:
  - a generalized trace $t_0, t_1, \ldots, t_n$
  - $t_m \Rightarrow t_n$ for some $0 \leq m < n$ ("closing" the generalized loop)
  - $t_k \Rightarrow \neg q$ for some $m \leq k \leq n$ (detecting violation of q)



Observations

- The existence of a generalized liveness CEX always implies the existence of a concrete CEX
- A generalized liveness CEX can be exponentially shorter than a concrete CEX
- Makes sense to develop algorithms that search for generalized counterexamples
  - In the paper we suggest a BMC-like algorithm based on 3-valued netlist encoding

k-LIVENESS

- Reference: “A Liveness Checking Algorithm that Counts”, FMCAD’12 [Claessen-Sörensson]

- A safety query of the form “is there a trace on which $\neg q$ occurs at least k times” is passed to a model checker

- If there is no such trace for some k, FGq passes

- Does not detect whether FGq fails

Extending k-LIVENESS

- Analyze counterexample traces
  - $\neg q$ occurs at least k times
  - somewhat generalized - if implemented on top of PDR

- If there are states t_m, t_n, t_k with $m < k \leq n$ so that $t_m \Rightarrow t_n$ and $t_k \Rightarrow \neg q$ then FGq fails. Both checks are purely syntactic (very fast).
- Detects failure of FGq on 44 HWMCC'12 liveness benchmarks (with small values of k)
- On 2 benchmarks performs significantly better than BMC

Design	k-LIVENESS	BMC
cubak	295s	12084s
cuhanoi10	5s	3492s

Example

- (q, x, y) – state variables
  - initially: $q = 1, x = 0, y = 0$
  - next-state: $q' = q \wedge x, \quad x' = x, \quad y' = \neg y$

- Consider traces of length 2:
  - concrete: $(1, 0, 0) \rightarrow (0, 0, 1) \rightarrow (0, 0, 0)$ not a CEX
  - generalized: $(1, 0, \cdot) \rightarrow (0, 0, \cdot) \rightarrow (0, 0, \cdot)$ CEX
  - generalized more: $(1, 0, \cdot) \rightarrow (0, 0, \cdot) \rightarrow (0, \cdot, \cdot)$ not a CEX

Generalizing traces may create or destroy liveness CEXes

Manipulating Traces

- Generalization ("backwards")
  - If s is a predecessor of t, sometimes can remove variables from s
- Concretization ("forward")
  - If s is a predecessor of t, sometimes can add variables to t
- ConcretizeTentative ("try to close the loop")
  - If t_i and t_j have no variables in opposite polarities ($i<j$), concretize from t_i towards t_j

Design	k generalized	k concrete	k modified
cubak	20	20	20
cujc128f	5	1	1
cutf2	9	12	5
cutq2	16	16	12
lmcs06dme2p0	4	5	4

Concluding remarks

- Generalized counterexamples to liveness can be significantly shorter than concrete counterexamples
- It makes sense to search for generalized counterexamples directly
- k-LIVENESS can be easily extended with failure detection
- Traces may be manipulated to increase the chance of detecting a counterexample

Thank You!