Automated Debugging of Missing Assumptions

BRIAN KENG[1], EVEAN QIN[2],
ANDREAS VENERIS[1],
BAO LE[1]

[1]UNIVERSITY OF TORONTO

[2]VENNSA TECHNOLOGIES INC.

Outline

- Motivation
- Background
- Debugging Missing Assumptions
  - Using a Single Counter-example
  - Using Multiple Counter-examples
- Experimental Results
- Conclusion

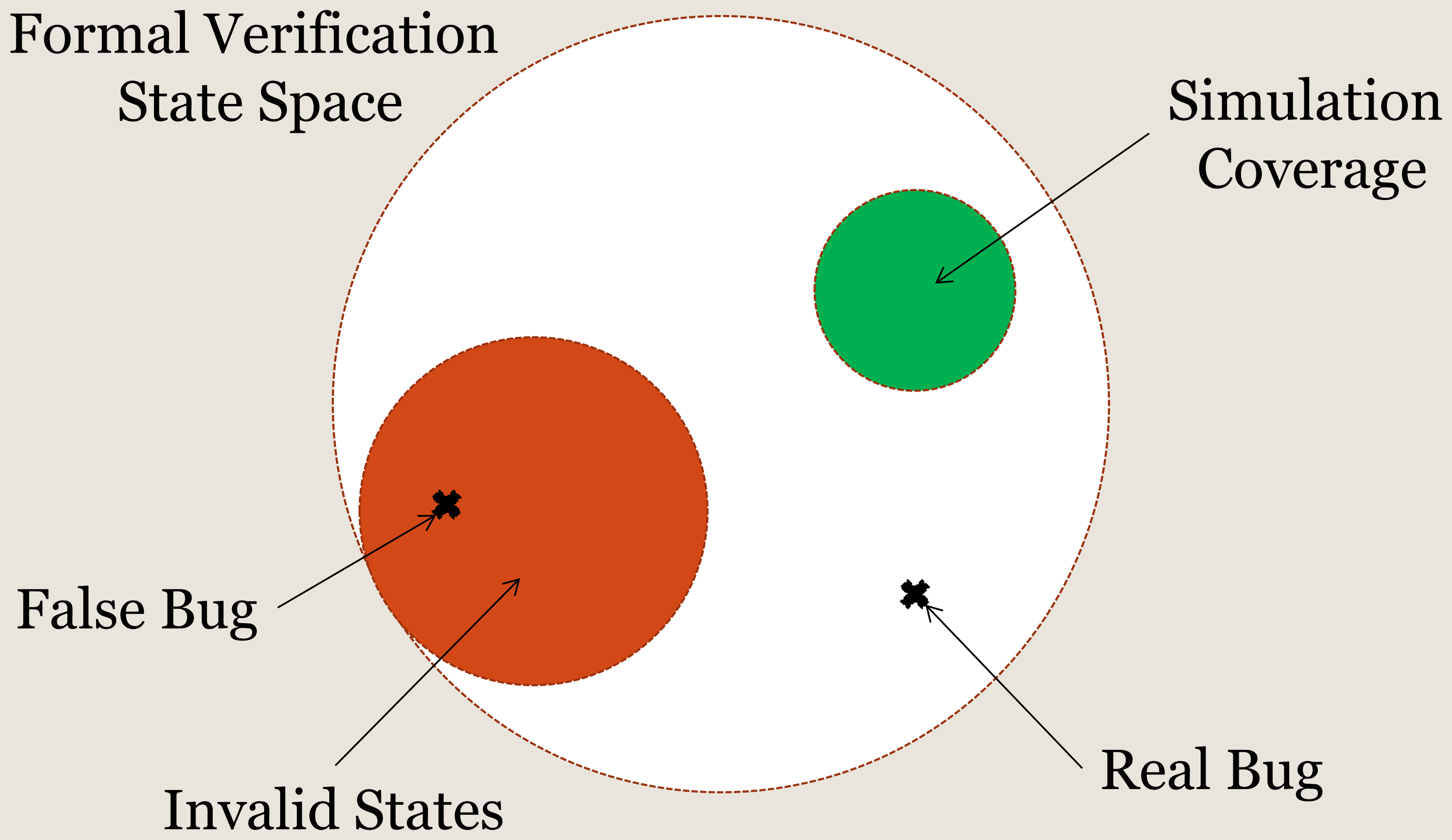
Outline

- Motivation
- Background
- Debugging Missing Assumptions
  - Using a Single Counter-example
  - Using Multiple Counter-examples
- Experimental Results
- Conclusion

Motivation

- Formal Property Checkers
  - Exhaustively verify an assertion which encodes the design intent
    - Returns counter-example that excites failure in the design
    - Can locate hard-to-find corner case failures
- Debugging formal counter-examples can be challenging, as observed failures can be due to:
  - A design bug
  - An incorrectly written assertion
  - Or a missing assumption

Motivation


Formal Verification
State Space
Simulation
Coverage
False Bug
Invalid States
Real Bug

Motivation

- Causes of Missing Assumptions
  - The design specification
  - Undocumented assertions
  - Functionality of adjacent design blocks
- The engineer needs to find the missing assumptions in order to prune the returned counter-example list
- This will expose counter-examples encoding “real” design bugs

Outline

- Motivation
- Background
- Debugging Missing Assumptions
  - Using a Single Counter-example
  - Using Multiple Counter-examples
- Experimental Results
- Conclusion

MUS and MCS

- Given a UNSAT Boolean formula Φ in CNF:
  - UNSAT Cores:
    - Subset of clauses in Φ that are UNSAT
  - Minimal Unsatisfiable Subset (MUS)
    - UNSAT core where every proper subset is SAT
  - Minimal Correct Set (MCS)
    - Minimal subset of clauses in Φ such that removing these clauses will make Φ SAT

UNSAT Core Example

$$(a) \wedge (b) \wedge (\bar{a} \vee \bar{b})$$

UNSAT Core and MUS

MCSs

MUIS and MCIS

- Minimal Unsatisfiable Input Subset (MUIS)
  - A minimal unsatisfiable set of input unit clauses that result in Φ being UNSAT
- Minimal Correction Input Set (MCIS)
  - A minimal set of input unit clauses that when removed, will result in Φ being SAT
- MUIS (MCIS) are analogous to MUS (MCS)

MUIS and MCIS Example

$$(a)\wedge(b)\wedge(c)$$

Input Clauses

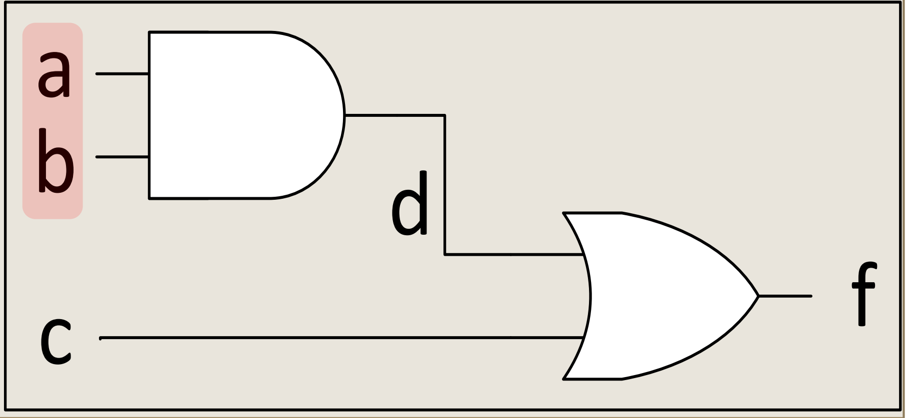
$$\wedge(a\vee\bar{d})\wedge(b\vee\bar{d})\wedge(\bar{a}\vee\bar{b}\vee d)$$

Property

$$\wedge(\bar{c}\vee f)\wedge(\bar{d}\vee f)\wedge(c\vee d\vee\bar{f})\wedge(\bar{f})$$

Transition Clauses

MCIS



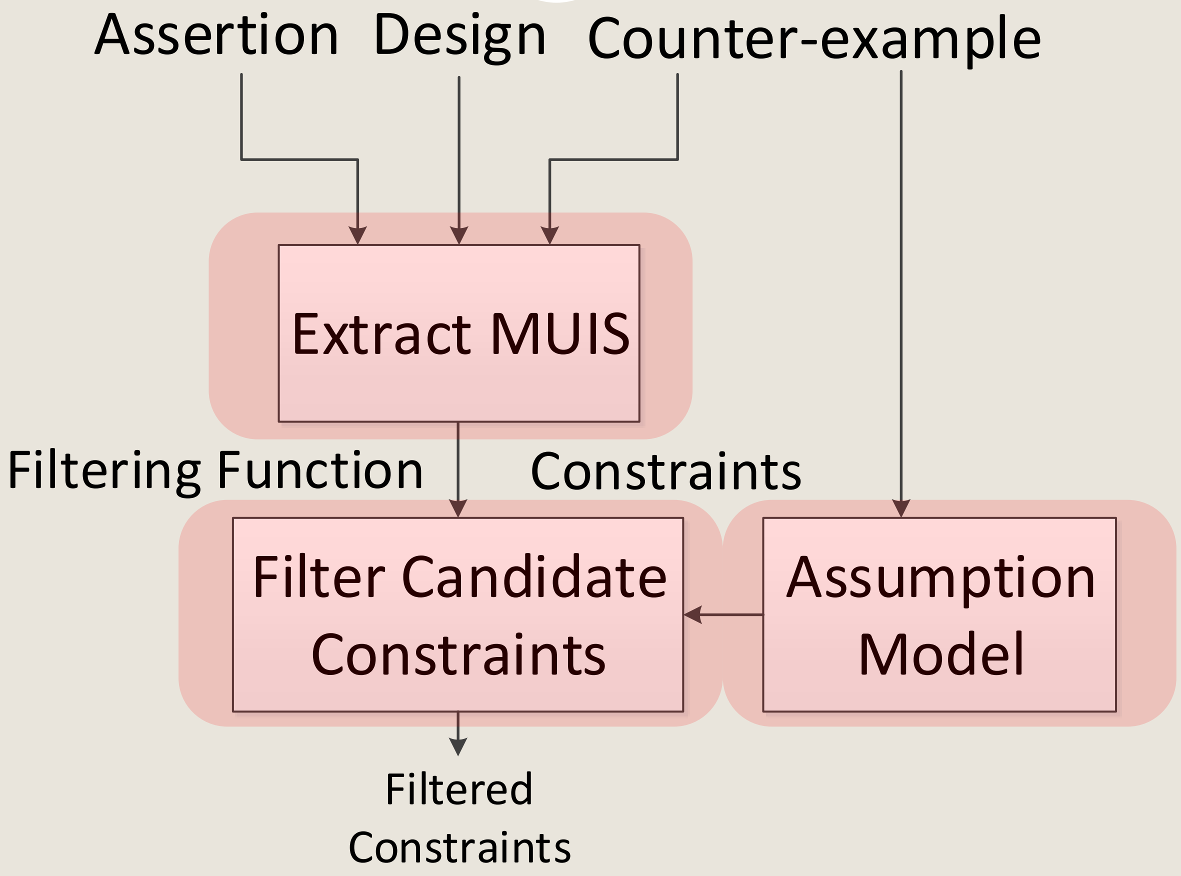
Outline

- Motivation
- Background
- Debugging Missing Assumptions
  - Using a Single Counter-example
  - Using Multiple Counter-examples
- Experimental Results
- Conclusion

Debugging Missing Assumptions

- Idea:
  - Give the engineer suggestions for the missing assumptions
- Extract all MUIS , U^i, from the design CNF to build a filtering function $F = U^o \ldots U^k$
- Given an input constraint A:
  - If F • A is SAT, the failure seen in the counter-example is not prevented
  - If F • A is UNSAT, then A will ensure that future failures will not occur in the same way as the given counter-example.
- MUISs can be computed in terms of MCISs

Debugging Missing Assumptions



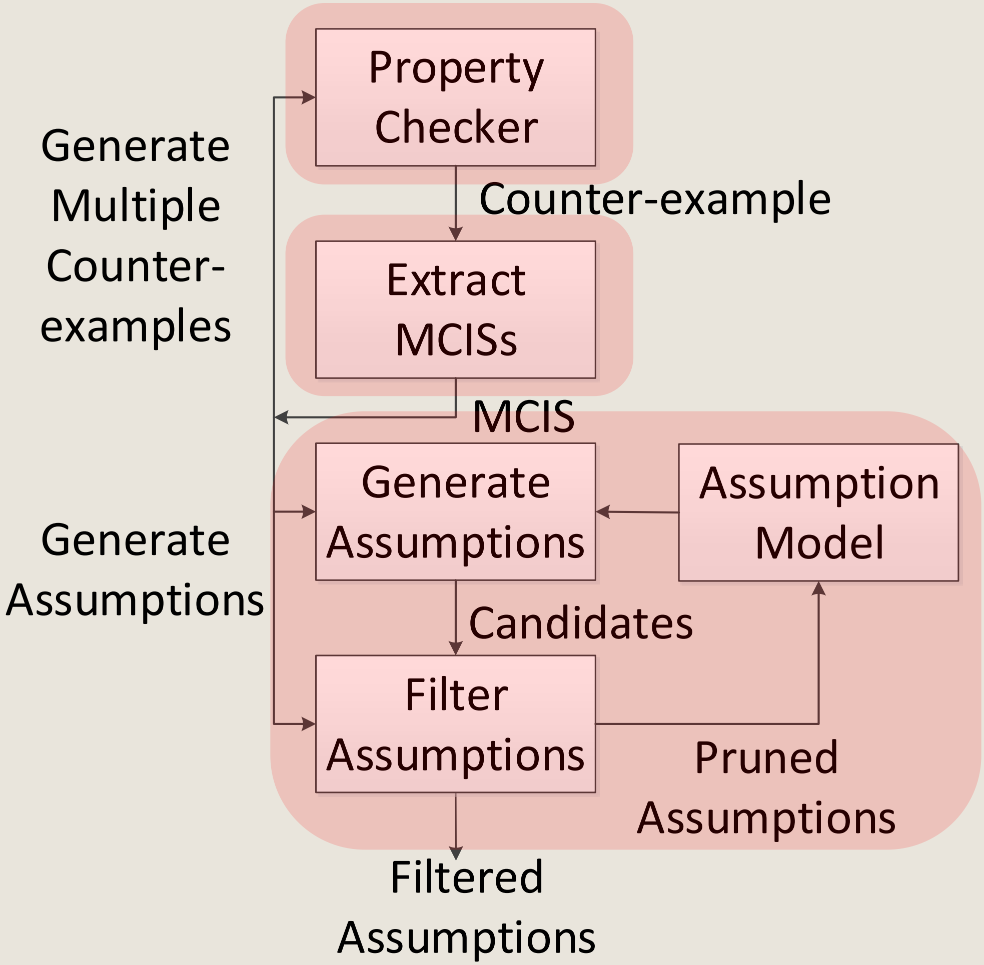
- B. Keng and A. Veneris, “Automated debugging of missing input constraints in a formal verification environment,” in Formal Methods in CAD, 2012.

Outline

- Motivation
- Background
- Debugging Missing Assumptions
  - Using a Single Counter-example
  - Using Multiple Counter-examples
- Experimental Results
- Conclusion

Using Multiple Counterexamples: Overall Flow

- Generate multiple distinct counter examples using formal tool
- Generate input assumptions that can prevent failures seen in the counter-examples
- More counter examples can aid general debugging



Generating Multiple Counter-examples

- It is difficult to generate a ‘useful’ second counter-example
  - The assertion should fail in a different manner
  - Therefore, distinct counter-examples must be found
- Two counter-examples, R and S, are distinct given their set of MUSs, M_R and M_S, such that:

$$\mathbf{M_R \cap M_S = \emptyset}$$

Generating Multiple Counter-examples

- To generate distinct counter-examples, we must prevent previously seen MUSs from occurring again
  - The MUS can be prevented if at least one of its clauses is not present
  - Functionality of the design must not be changed
  - Only input clauses can be modified to retain functionality

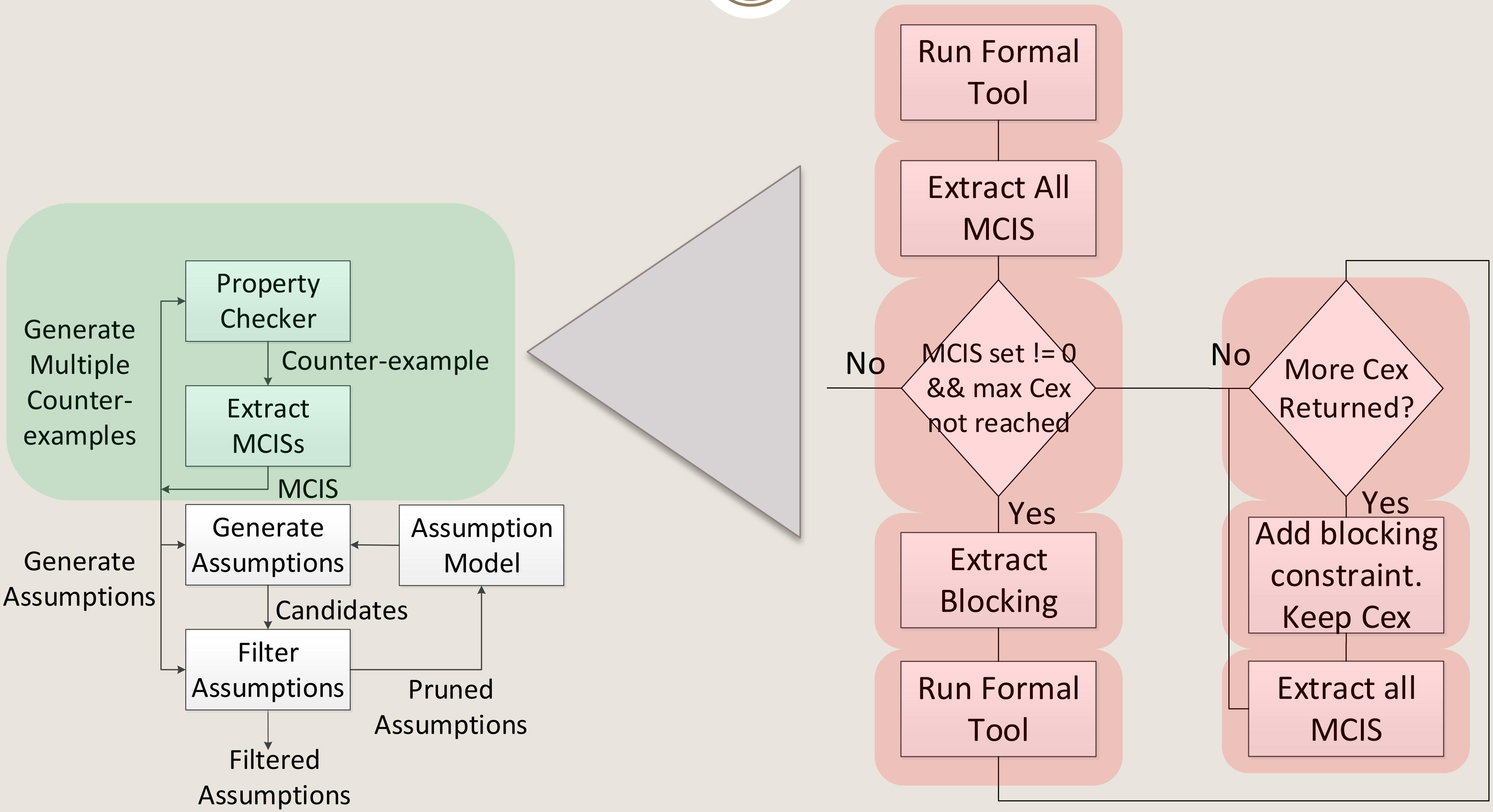
Generating Multiple Counter-examples

- As a result, previously found MUISs can be blocked.
- Using the duality between MUISs and MCISs, the blocking constraint can be computed from a single MCIS.

$$B^k = \overline{c_0^k} \wedge \overline{c_1^k} \wedge \ldots \wedge \overline{c_{|C^k|}^k}$$

$$B = B^1 \wedge \cdots \wedge B^k$$

A Practical Algorithm



Outline

- Motivation
- Background
- Debugging Missing Assumptions
  - Using a Single Counter-example
  - Using Multiple Counter-examples
- Experimental Results
- Conclusion

Experimental Results

Crt Name	# CE	MCIS Time (s)	Formal Time (s)	Gen Time (s)	Total Candidate Assumpt	Filter Cand. Assumpt
cpu	10	255	100	5	31	3
ddr2	9	383	1395	1504	4094	333
hpdmc	10	70	60	4	90	33
mips	4	278	93	9	59	22
mrisc	8	88	1126	5	39	10
pci	8	611	761	7	25	10

- Six designs from OpenCores used for evaluation, with assertions written based upon the specification document.

Experimental Results

Crt Name	# CE	MCIS Time (s)	Form Time (s)	Total Cand. Assumptions	Filt Using n CE			
					1	5	10	15
cpu	15	653	356	154	2	2	2	2
ddr2	3	625	86	226	68	-	-	-
hpdmc	15	112	148	97	17	16	11	11
mips	4	278	93	163	36	-	-	-
mrisc	8	88	1126	92	11	5	-	-
pci	8	611	761	267	9	9	-	-

- In other 5 cases, counter-example found similar to the original one but shifted in time.
- In another case, a counter-example is found using an entirely different way to excite the design and causes the assertion to fail.
- The width of the filter is extended by 30.4% and 38.3% respectively (averages for all experiments in paper)

Outline

- Motivation
- Background
- Debugging Missing Assumptions
  - Using a Single Counter-example
  - Using Multiple Counter-examples
- Experimental Results
- Conclusion

Conclusion

- Debugging missing assumptions
  - Generate multiple formal counter-examples for the failure
  - Generate a function that encodes the input combinations that caused the assertion to fail
  - Use the function to generate a list of fixed cycle assumptions that prevent the failures
- These can be used as hints for the actual missing assumption