Debugging Formal Specifications Using Simple Counterstrategies*

Robert Könighofer, Georg Hofferek, and Roderick Bloem

IAIK – Graz University of Technology

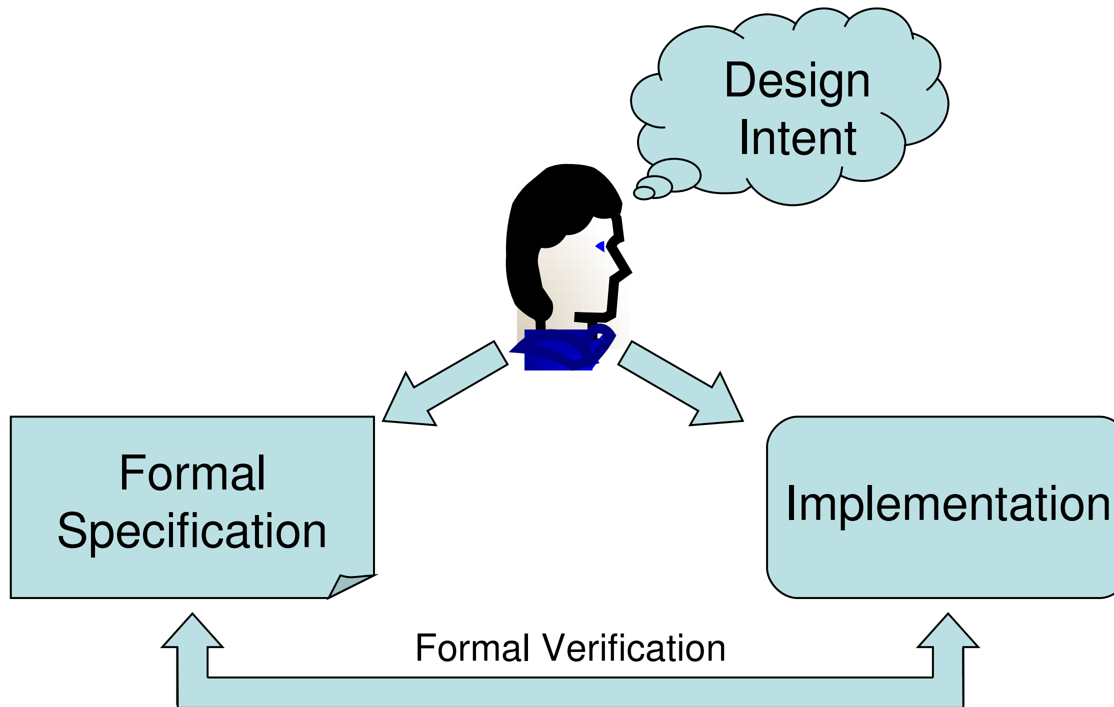
robert.koenighofer@student.tugraz.at

www.iaik.tugraz.at

* This work was supported in part by the European Commission through project COCONUT (FP7-2007-IST-1-217069).

Motivation

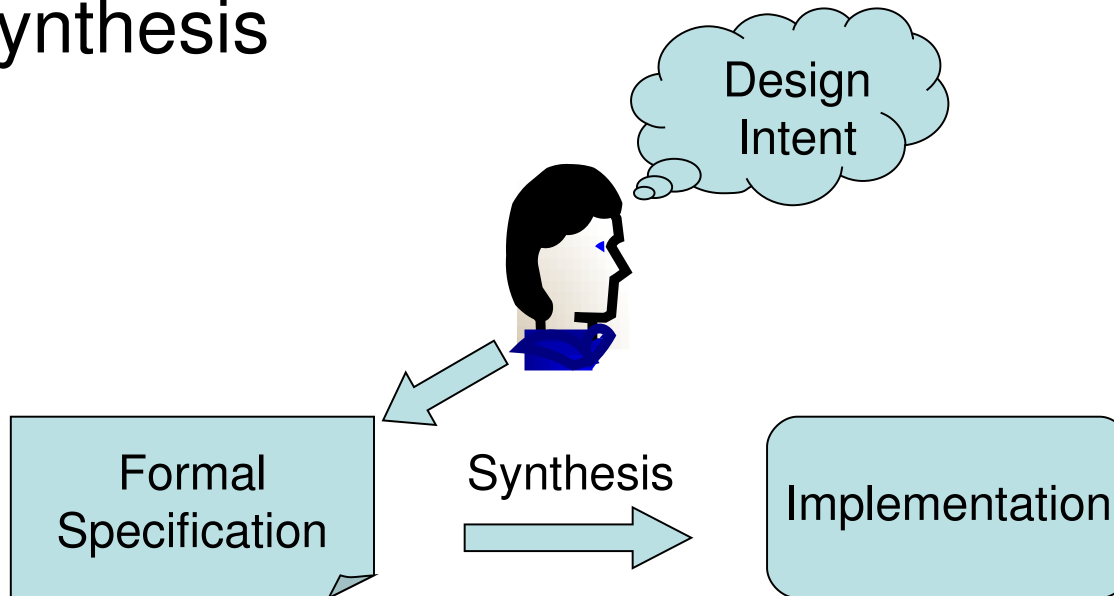
- Typical application of formal methods:



- Specification has to be correct!

Motivation

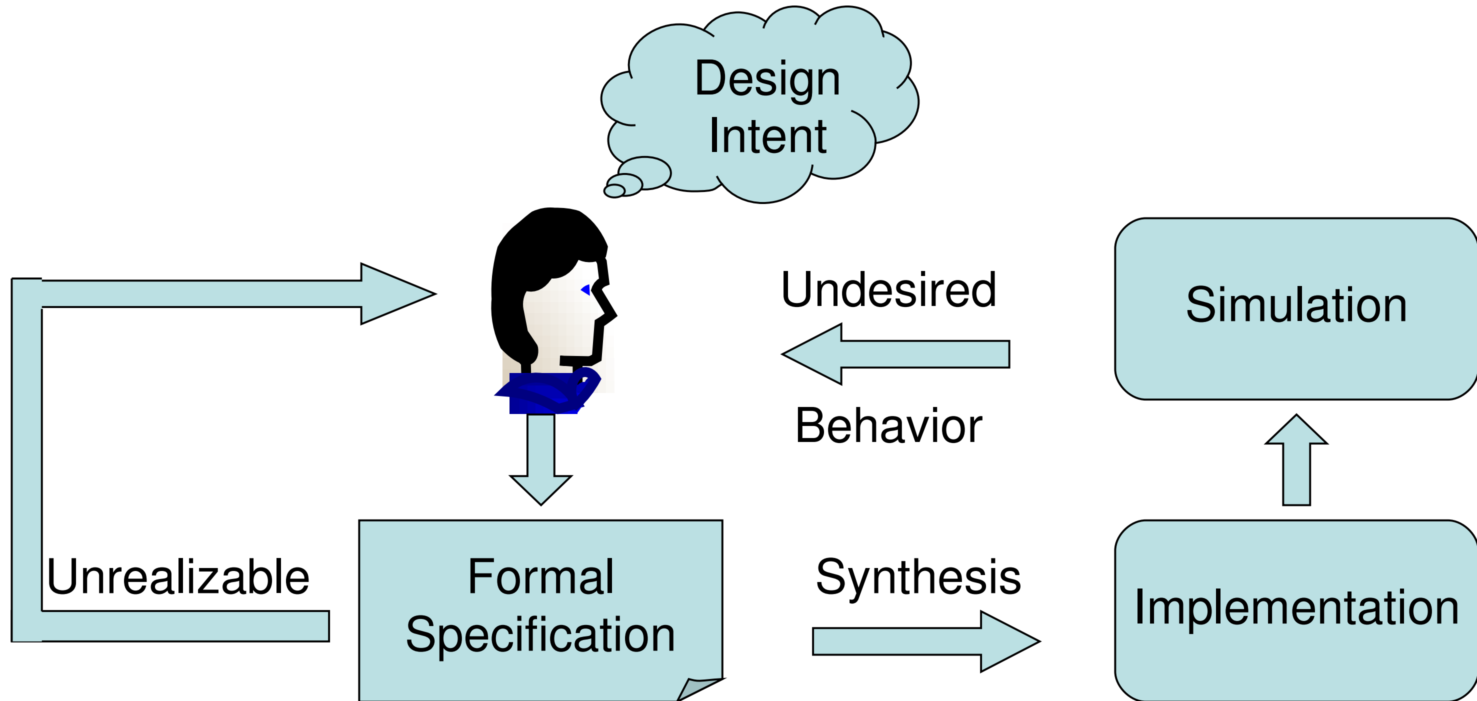
- Even more urgent: property based design + synthesis



- But: writing a correct specification is hard
- Bugs in specifications are difficult to fix

Motivation

- Specifying as an iterative process:



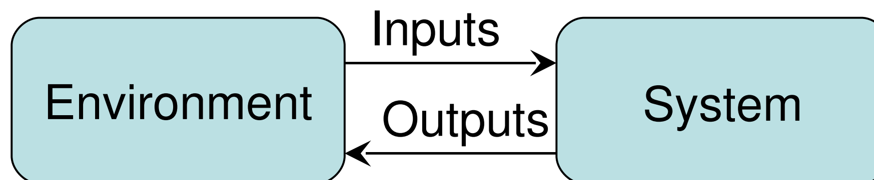
- We need techniques to debug incorrect specs

Objectives

- Goal: debug incorrect specifications
  - Incomplete: allows undesired behavior
  - Not sound: disallows desired behavior
  - Unrealizable
- Result:
  - Generic debugging approach
  - Elaboration, implementation, and evaluation for GR(1)

Setting

- Reactive Systems:



- Temporal specifications of the form A $\rightarrow$ G
- Satisfiability $\neq$ realizability
- Satisfiable: $\exists \overrightarrow{in} : \exists \overrightarrow{out} : (\overrightarrow{in} \,||\, \overrightarrow{out}) \models Spec$
- Realizable:
  - $\forall \overrightarrow{in} : \exists \overrightarrow{out} : (\overrightarrow{in} \,||\, \overrightarrow{out}) \models Spec$
  - + outputs depend on past and present inputs only

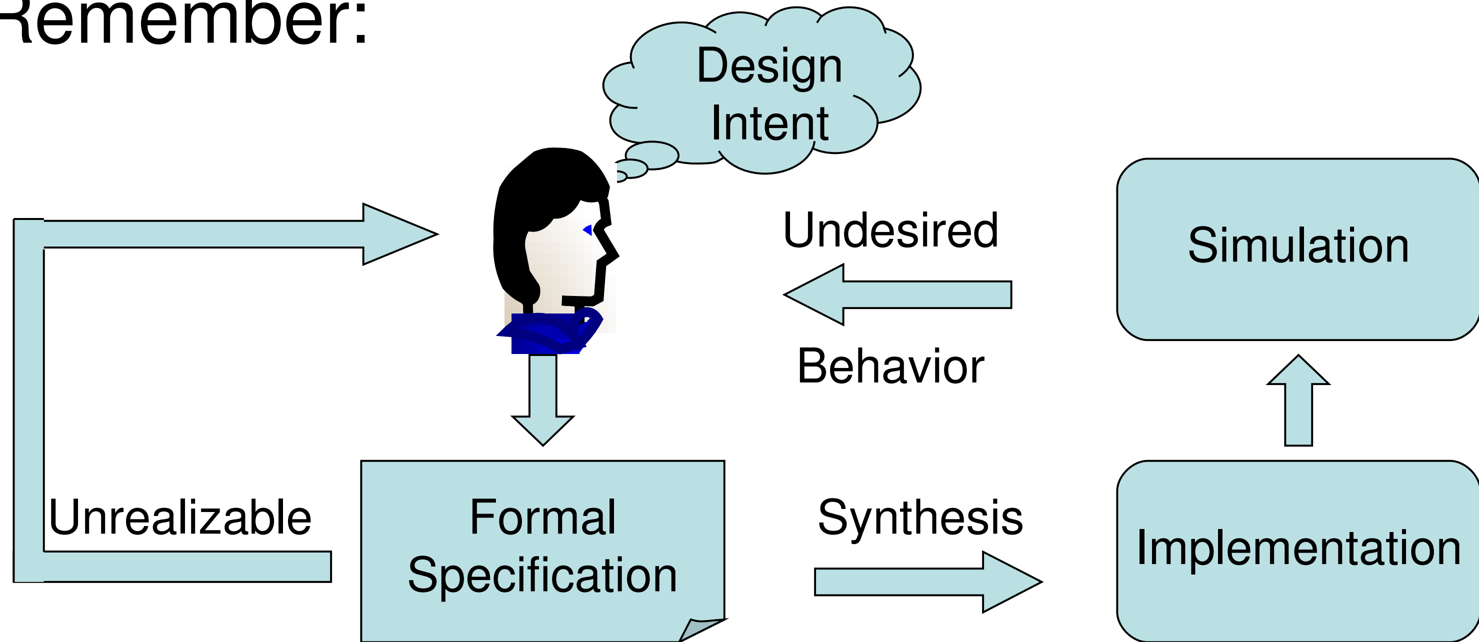
Setting - Realizability

Examples:

- ***always***(OUT=1) ∧ ***always***(OUT=0)
  - unsatisfiable, unrealizable
- ***always***(IN=1 ⇒ OUT=1) ∧ ***always***(IN=1 ⇒ OUT=0)
  - satisfiable, unrealizable
- ***always***(OUT ⇔ ***next***(IN))
  - satisfiable, unrealizable

Outline

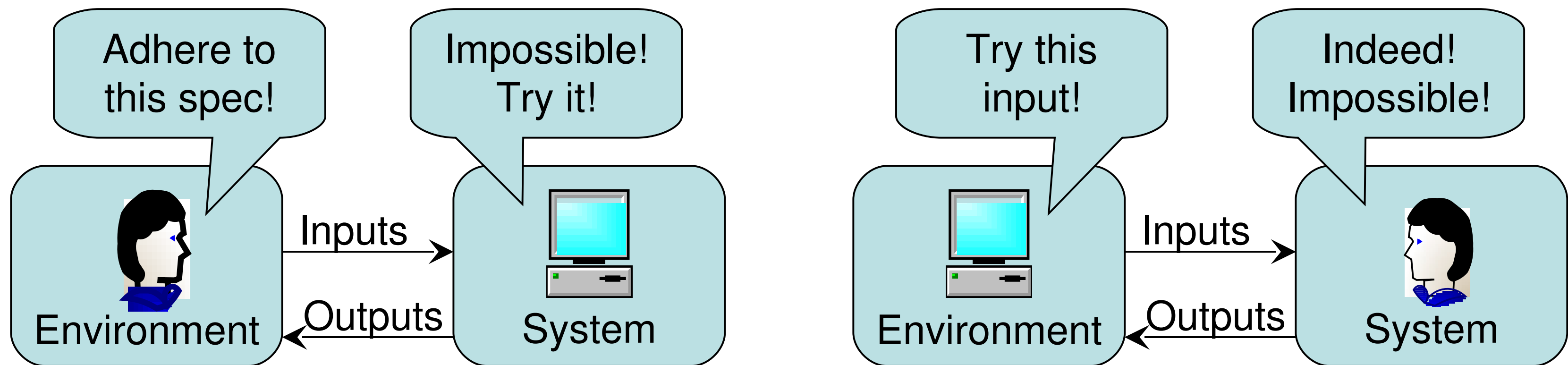
- Remember:



8. Debugging unrealizable specifications
9. Debugging undesired behavior
   - Reduction to a realizability problem

Debugging Unrealizability: Idea

- User has to understand the problem
- Reactive Systems: satisfiability ≠ realizability
- Illustration with counterstrategies
- Swapping the roles:

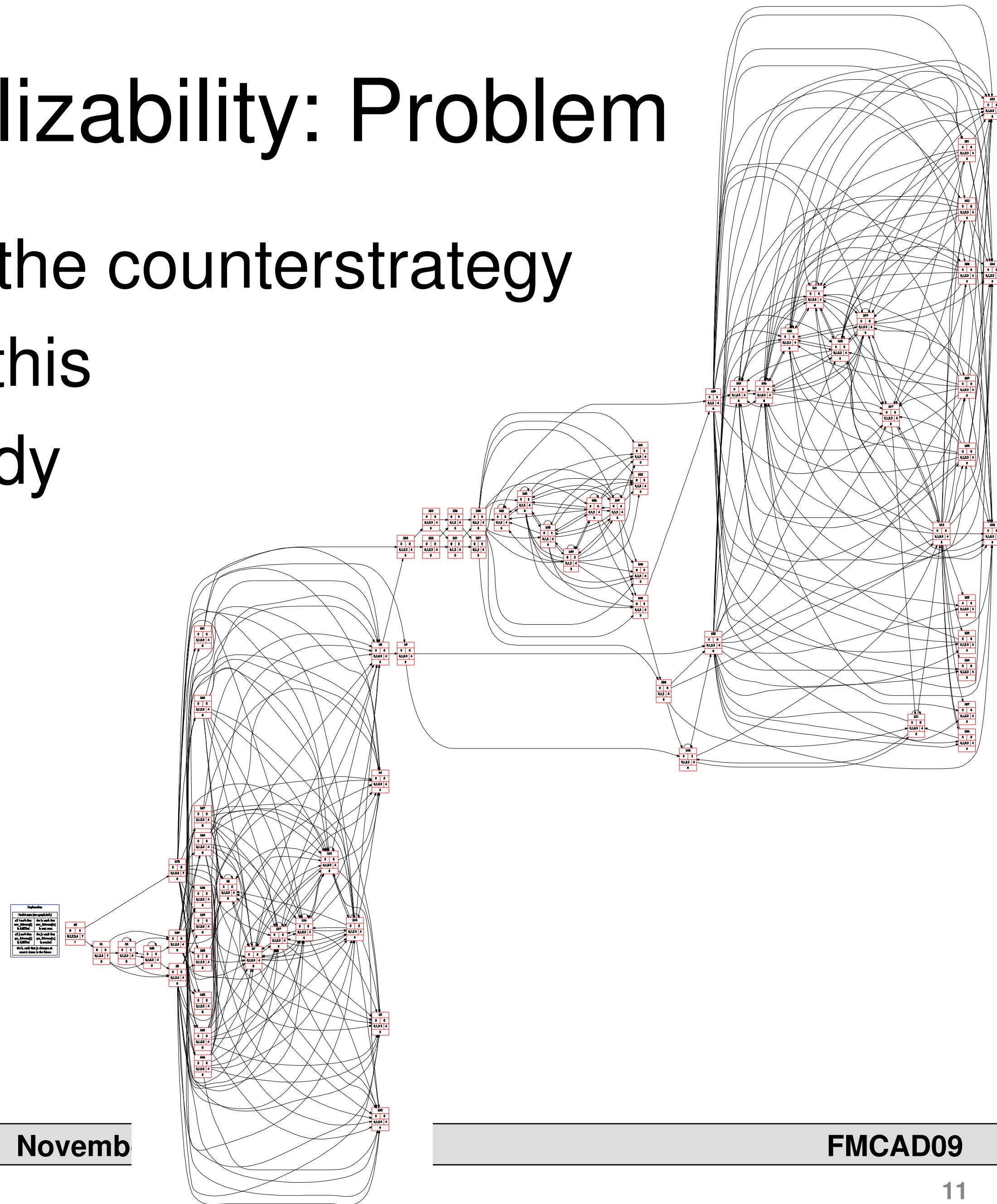


Debugging Unrealizability: Problem

- Counterstrategy can become complex
- Example:
  - ARM AMBA bus arbiter
  - 2 masters
  - 22 signals
  - 90 properties
- Input hready indicates that bus is released
  - Assumption: hready=1 again and again
  - Removed to make the specification unrealizable
  - The arbiter can no longer guarantee that requests are answered

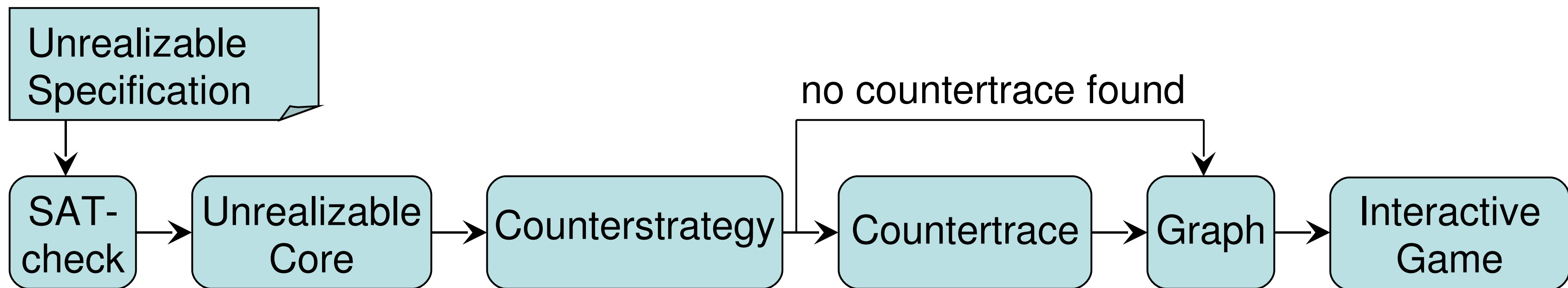
Debugging Unrealizability: Problem

- Graph illustrating the counterstrategy
- Very complex for this simple spec already

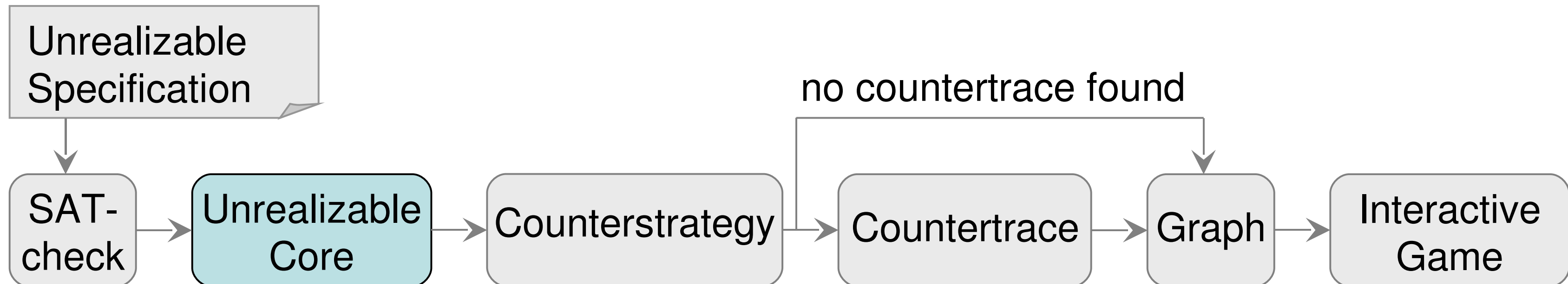


Debugging Unrealizability: Solution

- Debugging procedure:



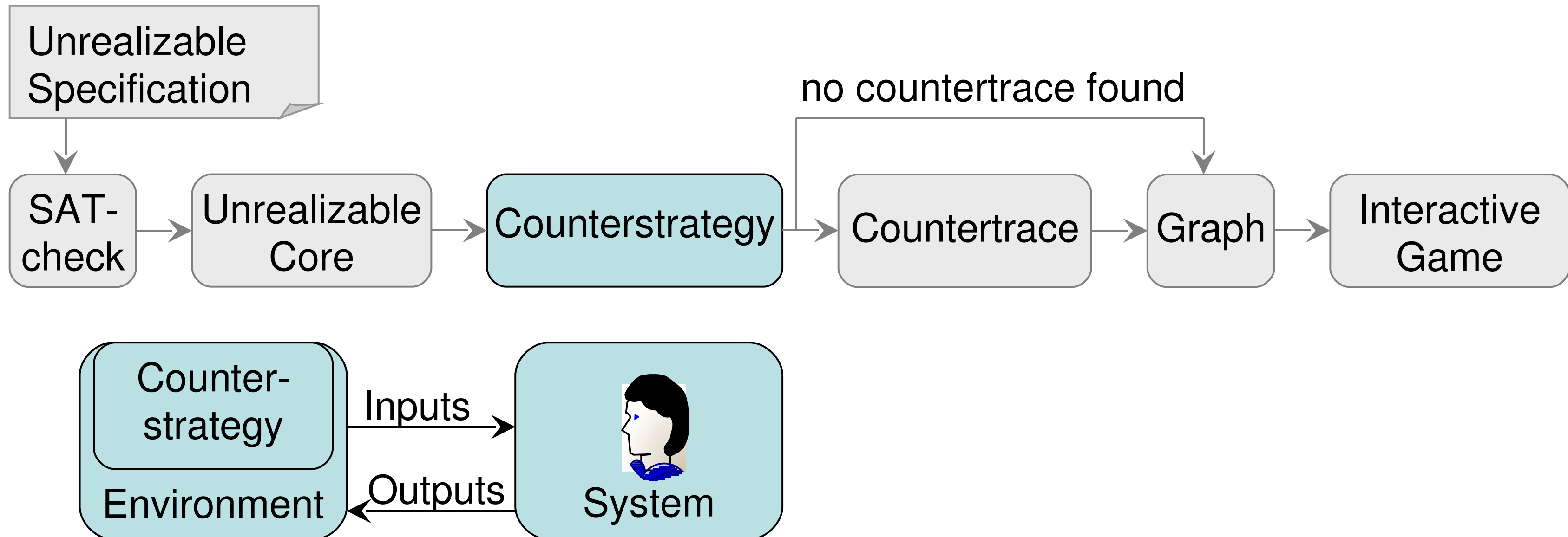
Debugging Unrealizability: Solution



- Idea [Cimatti08]: find a simpler spec that is still unrealizable
- Improvements:
  - Remove not only properties but also signals
  - Delta Debugging as a faster minimization algorithm

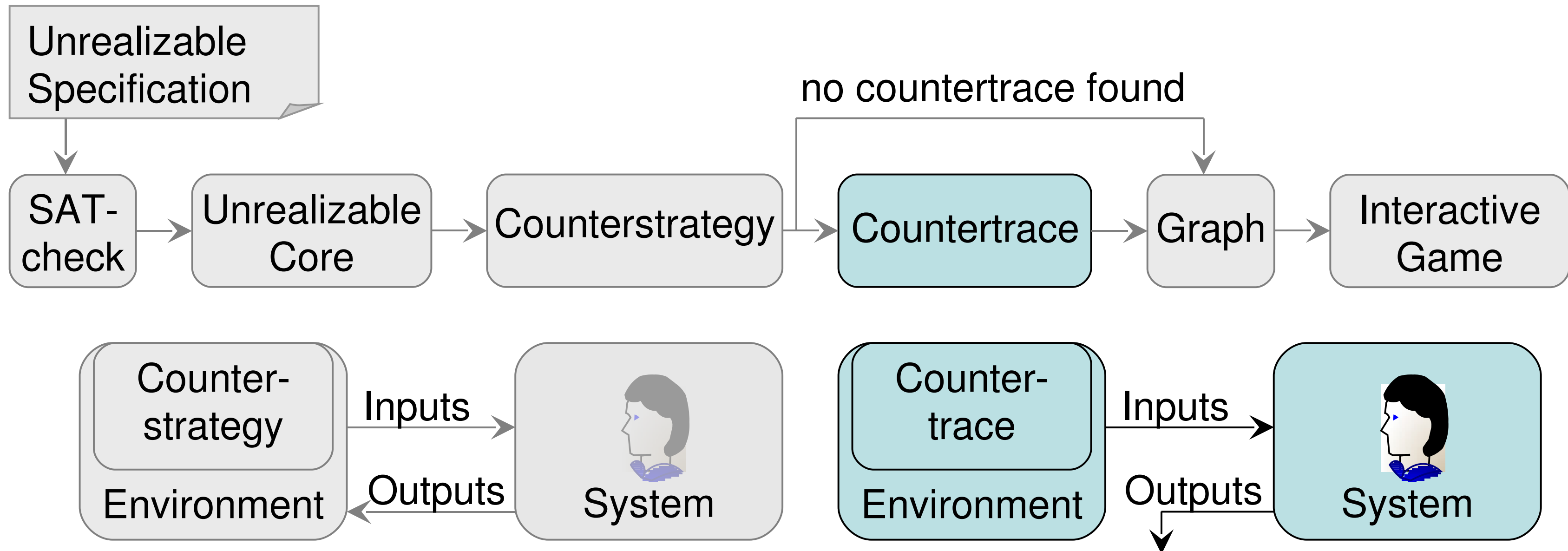
[Cimatti08] Alessandro Cimatti, Marco Roveri, Viktor Schuppan, and Andrei Tchaltsev. Diagnostic information for realizability. VMCAI, LNCS 4905, 2008.

Debugging Unrealizability: Solution



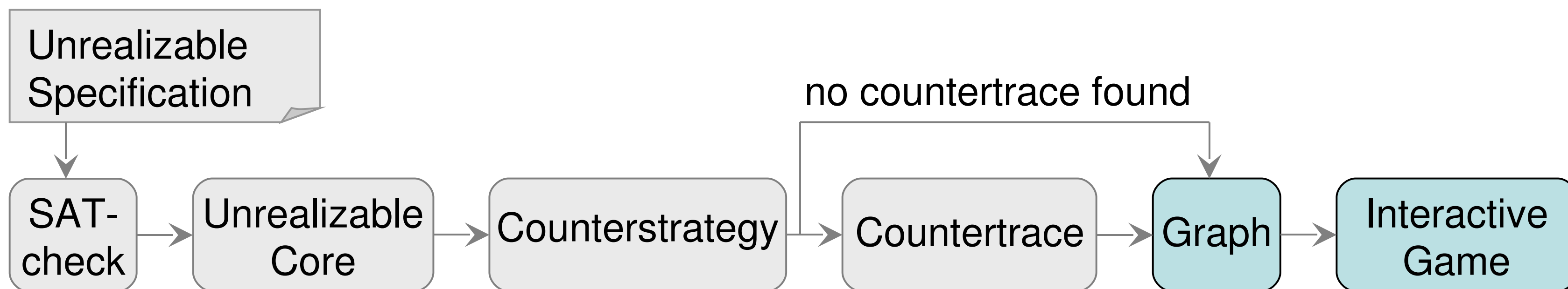
- Finds “problematic” inputs
  - No system behavior can fulfill the spec
  - Interactive nature: inputs depend on previous outputs

Debugging Unrealizability: Solution

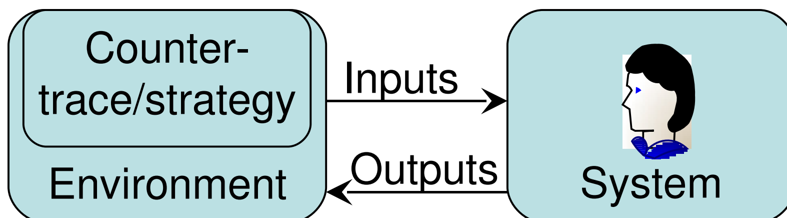


- A **single** input trace such that no system behavior fulfills the specification
  - Does not always exist
  - Computation is expensive → Heuristic

Debugging Unrealizability: Solution



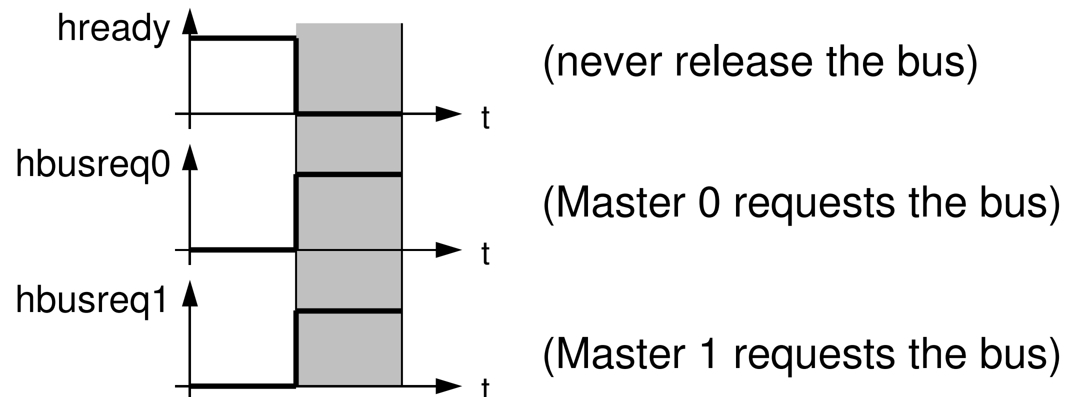
- Interactive game:



- Graph: summarizes all possible plays

Debugging Unrealizability: Example

- Remember our ARM AMBA bus arbiter example
  - Input hready: indicates that bus is released again
  - Environment assumption GF(hready=1) removed
  - System can no longer guarantee that requests are answered
- Unrealizable core [Cimatti08]
  - Removed: 70 % of the outputs, 95 % of the guarantees
- Countertrace:



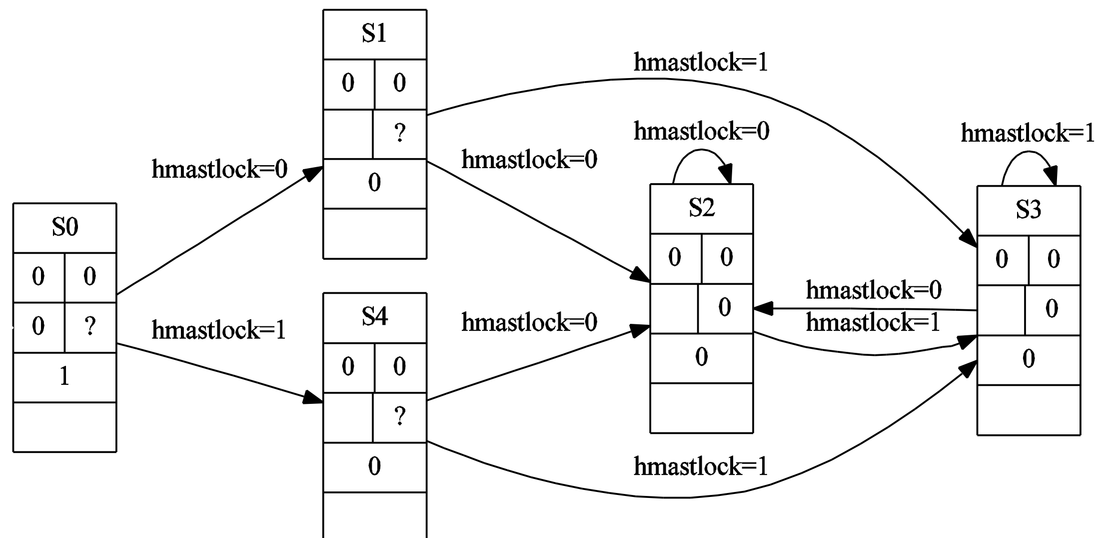
Debugging Unrealizability: Example

- Graph:

Constant next input values:
hready=0
hbusreq0=1
hlock0=1
hbusreq1=1
hlock1=1
hburst0=1
hburst1=1

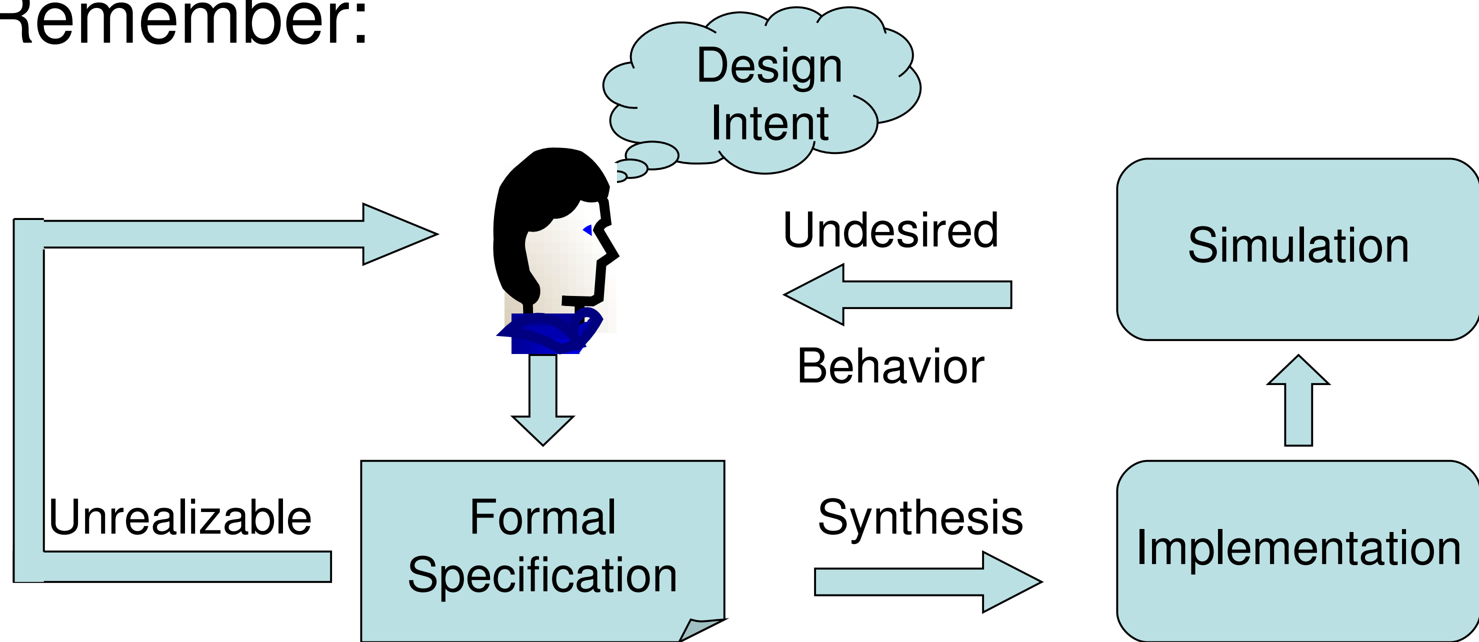
Explanation

NodeName (see graph.info)	
all i such that env_fairness[i] is fulfilled	the ix such that env_fairness[ix] is met next
all j such that sys_fairness[j] is fulfilled	the jx such that sys_fairness[jx] is evaded
the k, such that jx changes at most k times in the future	
changing next input values	



Pit Stop

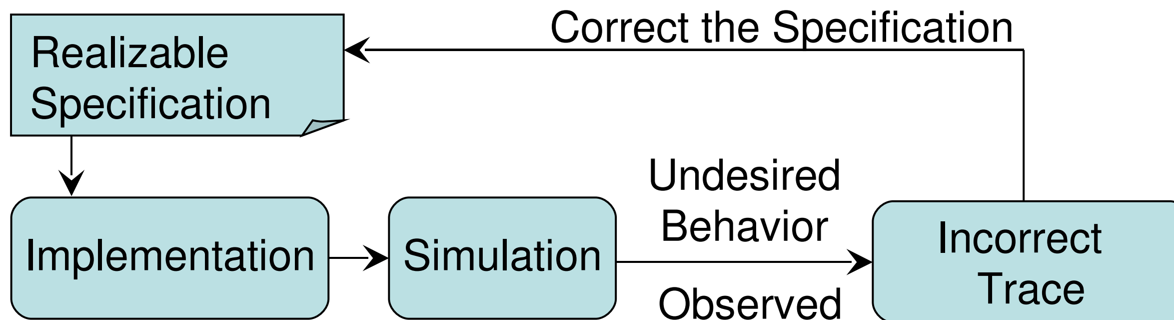
- Remember:



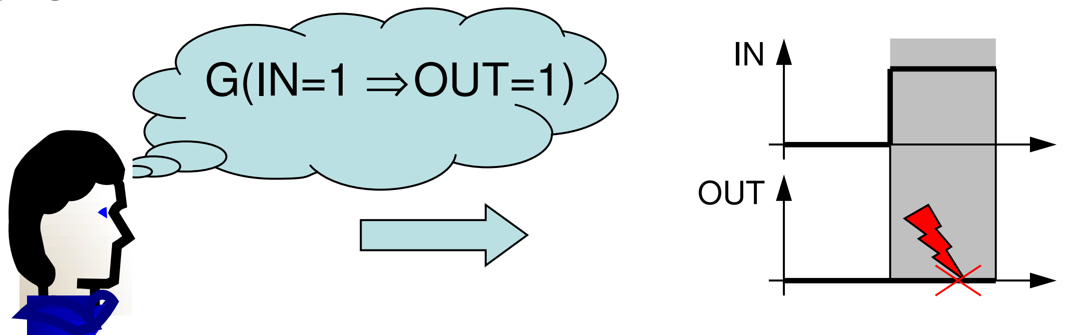
- Debugging unrealizable specifications Done
- Debugging undesired behavior Now
  - Reduction to a realizability problem

Debugging Undesired Behavior

- Scenario: undesired behavior observed

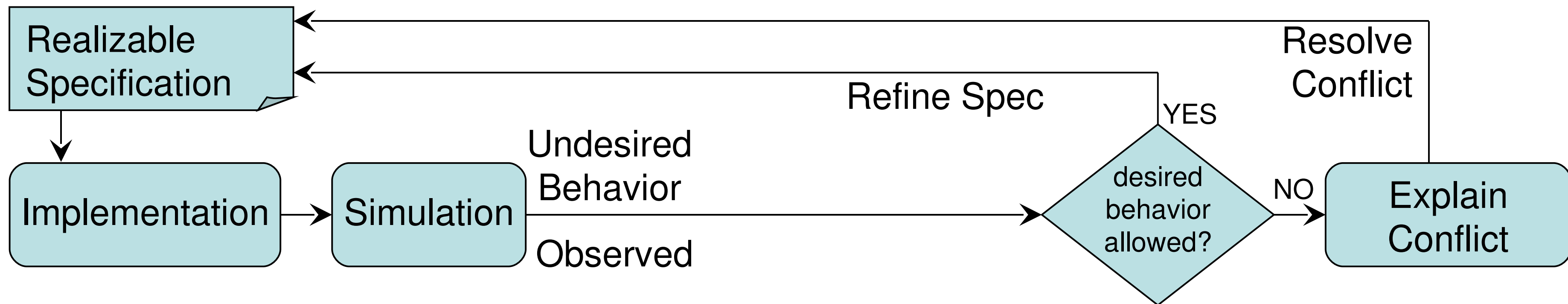


- Example:



Debugging Undesired Behavior

- Two cases:

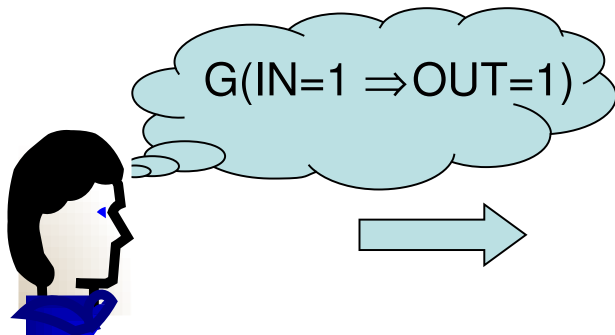


  - Spec allows observed and desired behavior
    → Incomplete
  - Spec disallows desired behavior
    → Not sound

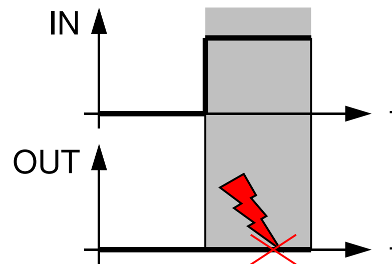
Debugging Undesired Behavior

- How can we distinguish between incompleteness and unsoundness?
  - The user specifies the desired behavior
  - Modifies the obtained simulation trace

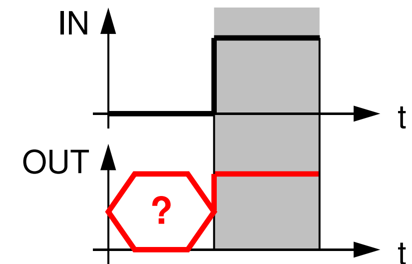
- Example:



Simulation Trace:

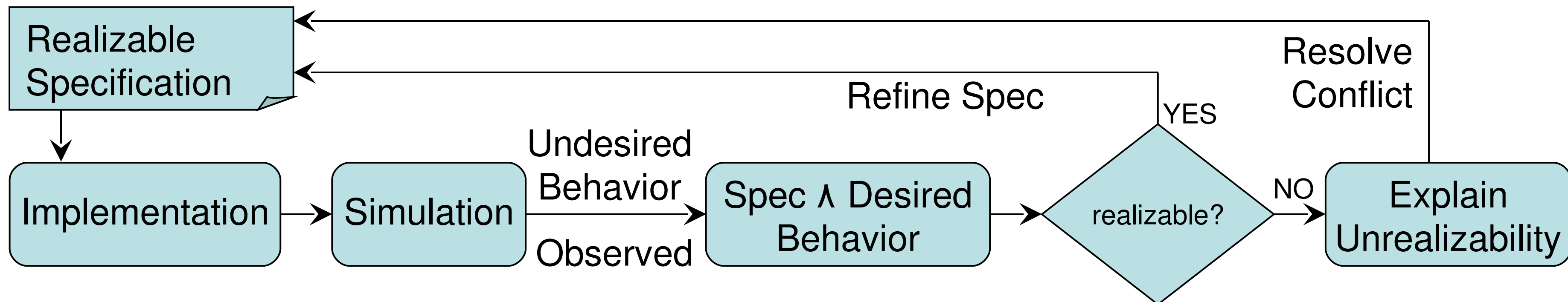


Desired Behavior:



Debugging Undesired Behavior

- Reduction to a realizability problem:



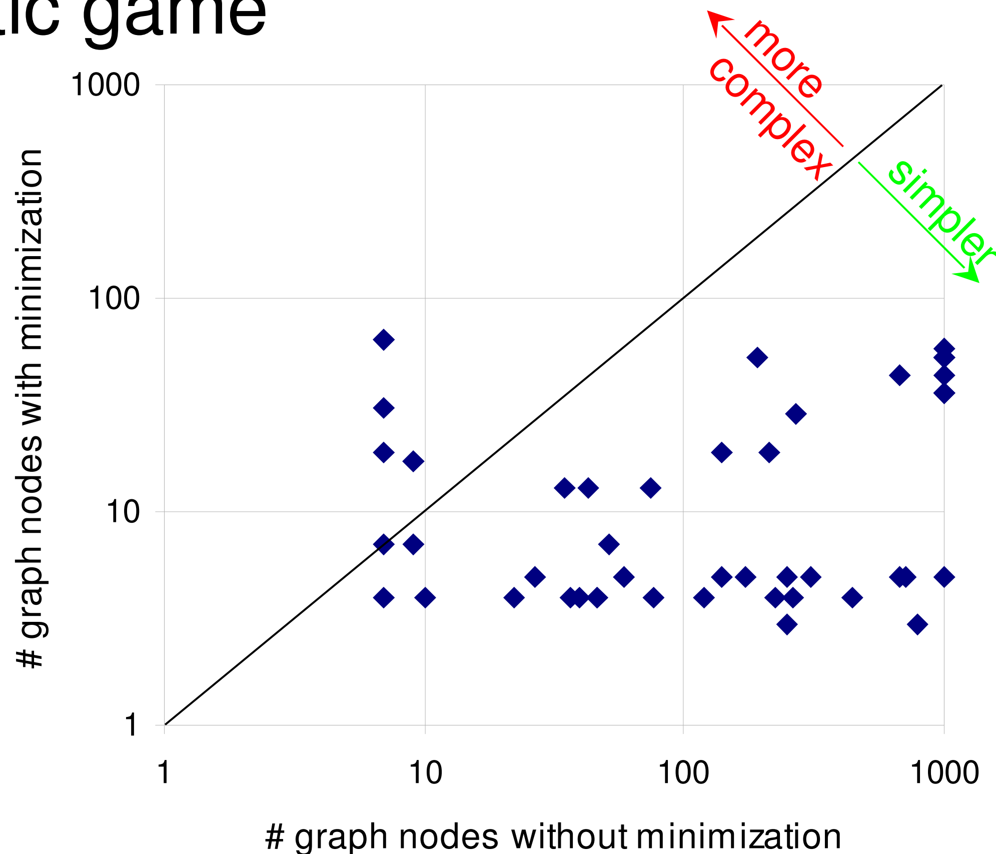
- Realizable:
  - Augmented specification eliminates incompleteness
- Unrealizable:
  - Conflict can be explained by explaining unrealizability

Experimental Results

- For GR(1) specifications
  - 22 to 218 signals
  - 90 to 6004 properties
- Countertraces are much easier to understand than counterstrategies
- Graph is helpful if no countertrace was found
- Our heuristic for countertrace computation:
  - Fast
  - Good success rate (80 %)

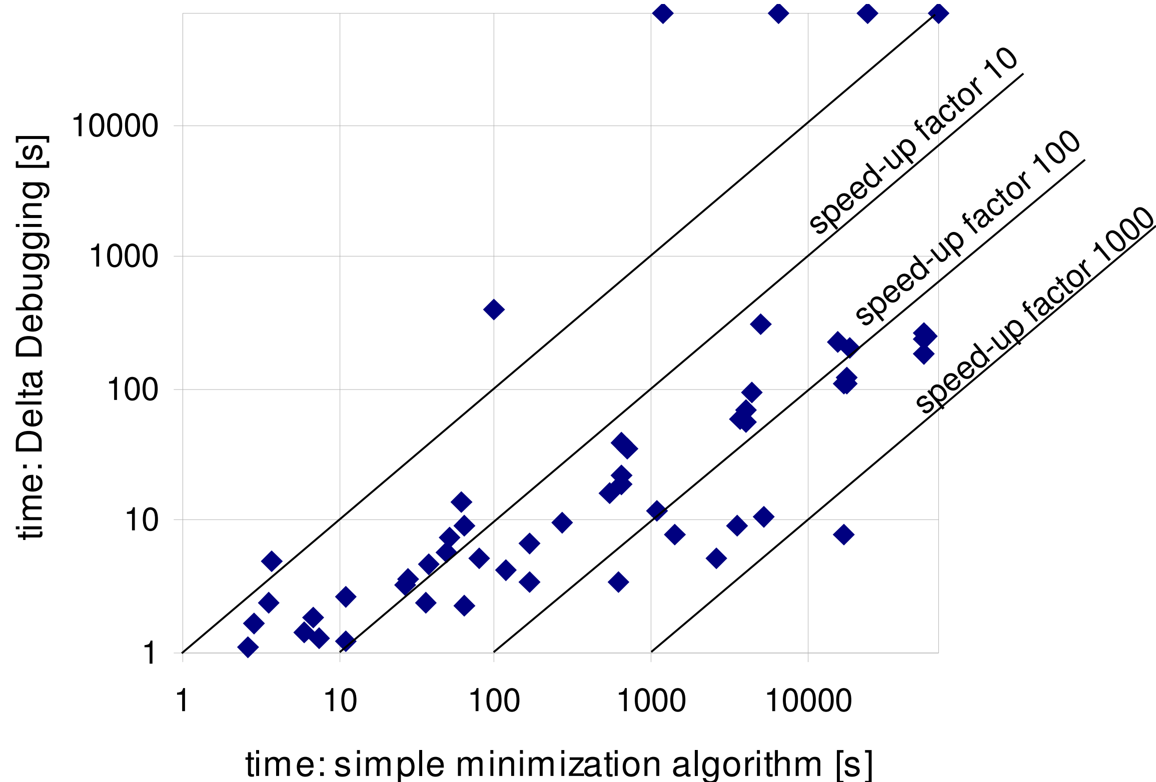
Experimental Results

- Minimization reduces the complexity of the diagnostic game



Experimental Results

- Delta Debugging is faster than the simple minimization algorithm



Implementation

- For GR(1) specifications
- In Anzu[1] and Ratsy[2]: Download it! Try it!

Game

Game Log

Dumping the specification into tmp.xml ...
Starting Marduk with the file ...
Checking for realizability (may take some time) ...
The spec is unrealizable.
I.e., no system can be constructed that implements the specification. Given an arbitrary system, the environment can provide inputs so that the system violates the specification. A specification can only be realizable if for each input trace, an output trace can be found so that the combination of the traces fulfills the specification. Additionally, the outputs in a certain time step may only depend on past and present inputs. So either your guarantees are too restrictive or the assumptions about the environment are too weak. The problem will be pointed out by the following steps.
Checking satisfiability (won't take that long) ...
The spec IS satisfiable.
That means: There exists a trace of inputs and outputs so that the specification is fulfilled.

Show Results | Show Operations | Show Help

Inputs: r0, r1, startup_failed, ix, jx, jx changes, state in graph

Outputs: g0, error, di_state0, di_state1

Step1 Step2 Step3 Step4 Step5

Start | Stop | Next Step | Clear | Prev. Step | Done | Export | Show Subviews | Hide Subviews | Play Game

Options for unrealizable specs: Sat Check | Minimize | Compute Graph | Specify Design Intent

[1] http://www.iaik.tugraz.at/content/research/design verification/anzu/

[2] http://rat.fbk.eu/ratsy

Conclusion

- Debugging formal specifications is hard
- Counterstrategies to illustrate problems
  - Unrealizability
  - Conflicts with the design intent
- Simplification is important
  - Unrealizable Core
  - Countertraces
- More details in my Master's Thesis
  https://online.tu-graz.ac.at/tug_online/edit.getVollText?pDocumentNr=114859

Questions/Discussion

... thank you for your attention!

Future Work: Model Based Diagnoses

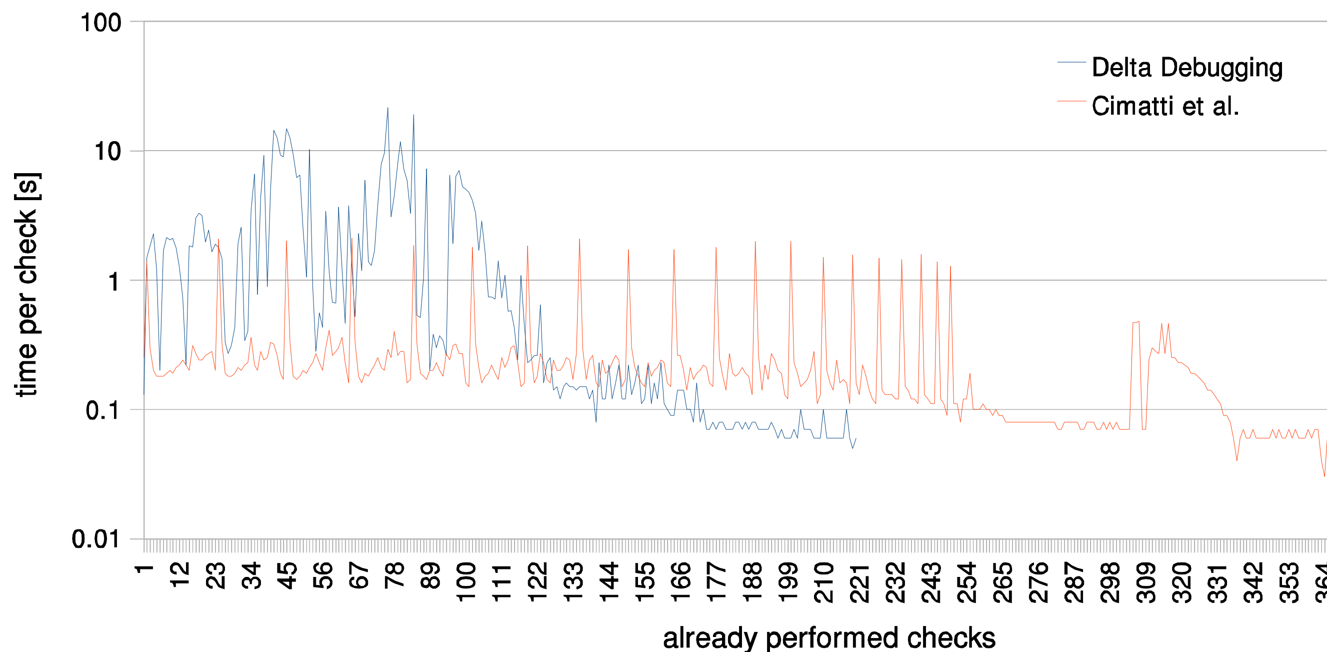
- Raymond Reiter. A theory of diagnosis from first principles. 1987.
- Conflict:
  - Set of components that cannot all be correct
  - → Set of guarantees/outputs that cannot all be correct
  - → Unrealizable Core = Minimal Conflict
- Diagnosis:
  - Set of components which, if assumed to be incorrect, explain **ALL** conflicts
  - → Points to guarantees/outputs which are likely to be incorrect
- Objections: computational effort

Sometimes: Bad performance of DD

- Compared to simple algorithm of Cimatti et al.
  - Removes one property/signal after the other
  - Linear number of checks
- Delta debugging:
  - Best case: logarithmic number of checks
  - Worst case: quadratic number of checks
- Surprising:
  - Less checks for realizability
  - More time

Sometimes: Bad performance of DD

- Details:



  - Peaks are realizability checks on realizable specifications
  - Simple algorithm needs a minimum of checks on real. specs.

Computing Countertraces