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!

Diagram:
- Design Intent
- Formal Specification
- Implementation
- Formal Verification
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

Diagram:
- Design Intent
- Formal Specification
- Simulation
- Implementation
- Undesired Behavior
- Unrealizable
- Synthesis
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: $\exists \overrightarrow{in} : \exists \overrightarrow{out} : (\overrightarrow{in} \parallel \overrightarrow{out}) \models Spec$
- Realizable:
  - $\forall \overrightarrow{in} : \exists \overrightarrow{out} : (\overrightarrow{in} \parallel \overrightarrow{out}) \models Spec$
  - + outputs depend on past and present inputs only
Setting - Realizability

Examples:

- \textit{always}(\text{OUT}=1) \land \textit{always}(\text{OUT}=0)
  - unsatisfiable, unrealizable

- \textit{always}(\text{IN}=1 \Rightarrow \text{OUT}=1) \land \textit{always}(\text{IN}=1 \Rightarrow \text{OUT}=0)
  - satisfiable, unrealizable

- \textit{always}(\text{OUT} \Leftrightarrow \text{next}(\text{IN}))
  - satisfiable, unrealizable
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 $\neq$ realizability
- Illustration with counterstrategies
- Swapping the roles:

![Diagram](image_url)

Adhere to this spec!

Impossible! Try it!

Try this input!

Indeed! Impossible!
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:

  Unrealizable Specification → SAT-check → Unrealizable Core → Counterstrategy → Countertrace → Graph → Interactive Game

  no countertrace found
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

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:
  - hready (never release the bus)
  - hbusreq0 (Master 0 requests the bus)
  - hbusreq1 (Master 1 requests the bus)
Debugging Unrealizability: Example

- Graph:
Pit Stop

- Remember:
  - Debugging unrealizable specifications
  - Debugging undesired behavior
    - Reduction to a realizability problem
Debugging Undesired Behavior

- Scenario: undesired behavior observed

- Example:

```plaintext
G(IN=1 ⇒ OUT=1)
```

Diagram:
- Realizable Specification
- Implementation
- Simulation
- Incorrect Trace
- Undesired Behavior: Observed
- Correct the Specification
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:
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

![Graph showing speed-up factor comparison between Delta Debugging and the simple minimization algorithm.]
Implementation

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

---

1. [http://www.iaik.tugraz.at/content/research/design_verification/anzu/](http://www.iaik.tugraz.at/content/research/design_verification/anzu/)
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


- 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