Debugging Specifications

Joint work with Robert Könighofer and Georg Hofferek

Roderick Bloem

IAIK – Graz University of Technology
Roderick.Bloem@iaik.tugraz.at

Setting

- Two views of specification
  - *Post facto* – verification
  - *Ante facto* – design description followed by manual or automatic synthesis
- Input: a temporal specification $A \rightarrow G$
  - Set of Büchi automata A for assumptions
  - Set of Büchi automata G for guarantees
- Output: a system that fulfills the specification
- Property Synthesis
  - Tools: Lily (LTL), Anzu (GR1)
  - We can synthesize real circuits (AMBA bus arbiter, GenBuf)

Motivation

- Property Synthesis – Correct by Construction?
  - Getting spec right is hard!
  - Specifying is an iterative process
  - Mistakes are easy to make, easy to detect, but:
  - **Finding fault in spec is very hard (not executable)**
- Things that may go wrong
  1. System behavior unexpected: underconstrained (coverage)
  2. **System behavior unexpected: incorrectly constrained**
  3. **Unreliazable (no system exists): overconstrained**
- I will explain unrealizability first, then incorrect specs
- Techniques here are implemented in Anzu but applicable to all synthesis methods (including liveness)

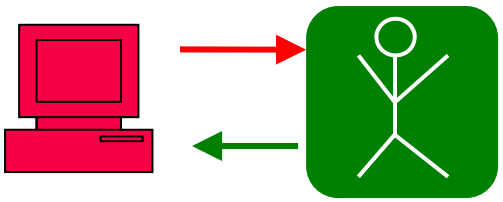
Unrealizability Examples

- input x, output y
- $G\neg y \wedge Xy$
  - unsatisfiable
- $(x \rightarrow Gy) \wedge X(y \leftrightarrow x)$
  - There is no output for some input
- $(y \leftrightarrow Xx)$
  - unrealizable
  - But for each input trace there is an output trace

Debugging Info

Prior: tools says *unrealizable* and stops

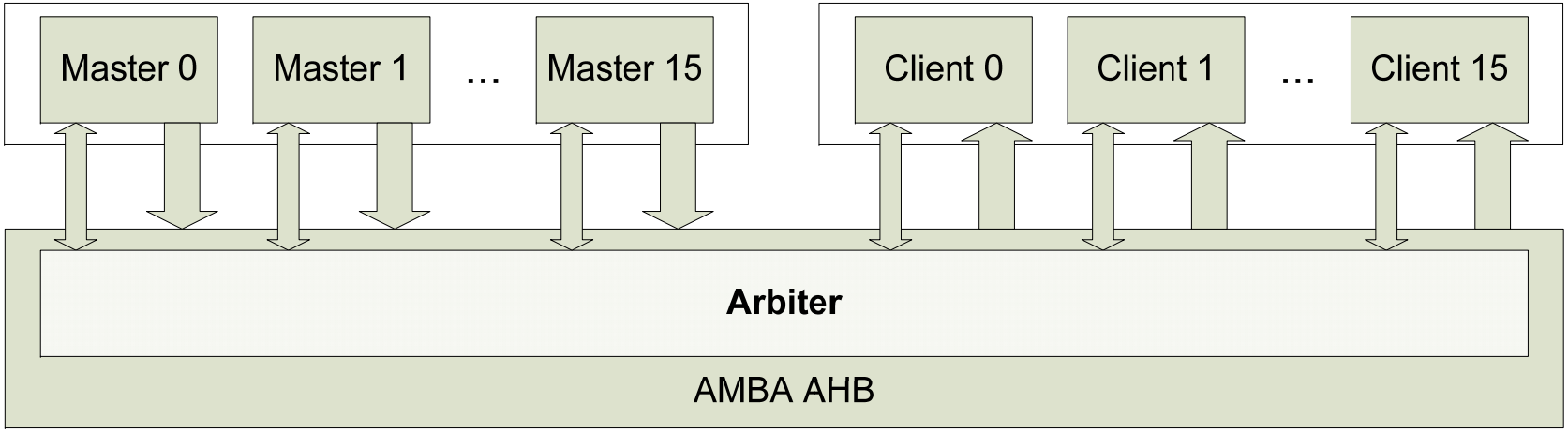
- Assumption: You can demo imagined implementation
  - If anything you try is wrong, you will understand why the spec is unrealizable
- Show a **counterstrategy** (Tripakis/Altisen99, etc.)
  - Counterstrategy: tool suggests inputs, you give outputs
  - Requires solving the co-game (co-GR1) and creating strategy
  - Let me show you that this does not suffice…



Environment (Moore) System (Mealy)

Example: AMBA Bus

- Industrial standard
- ARM's AMBA AHB bus
  - High performance on-chip bus
  - Data, address, and control signals (pipelined)
  - Arbiter part of bus (determines control signals)
  - Up to 16 masters and 16 clients



Example: AMBA Bus

- Master initiates transfer. It has the signals
  - HBUSREQi - Master i wants the bus
  - HLOCKi - Master i wants an uninterruptible access
  - HBURST - This access has length 1/4/incr
  - address & data lines
- The arbiter decides access
  - HGRANTi - Next transfer for master i
  - HMASTER[] - Currently active master
  - HMASTLOCK - Current access is uninterruptible
- The clients synchronize the transfer
  - HREADY - Ready for next transfer
- Sequence for master
  - Ask; wait for grant; wait for hready; state transfer type & start transfer

Example: AMBA Arbiter

- Specification
- 12 Guarantees, 3 Assumptions.
- Example:
  - *"When a locked unspecified length burst starts, new access does not start until current master (i) releases bus by lowering HBUSREQi."*

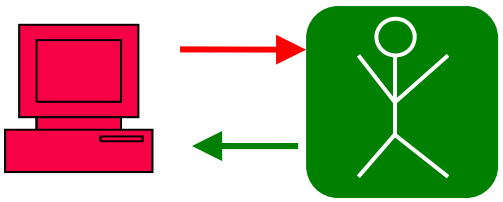
D[::]V D[::]V

  - $\bigwedge_i$ G(HMASTLOCK $\wedge$ HBURST=INCR $\wedge$ HMASTER=i $\wedge$ START $\rightarrow$ X(¬START U ¬HBUSREQi))

Example: Mistake

- Example or AHB with 4 masters
  - Spec size 113
  - 23 Signals
- Forget to require that clients release the bus by raising HREADY:
  - Remove assumption GF HREADY
  - Impossible for system to release bus → cannot assign to next client

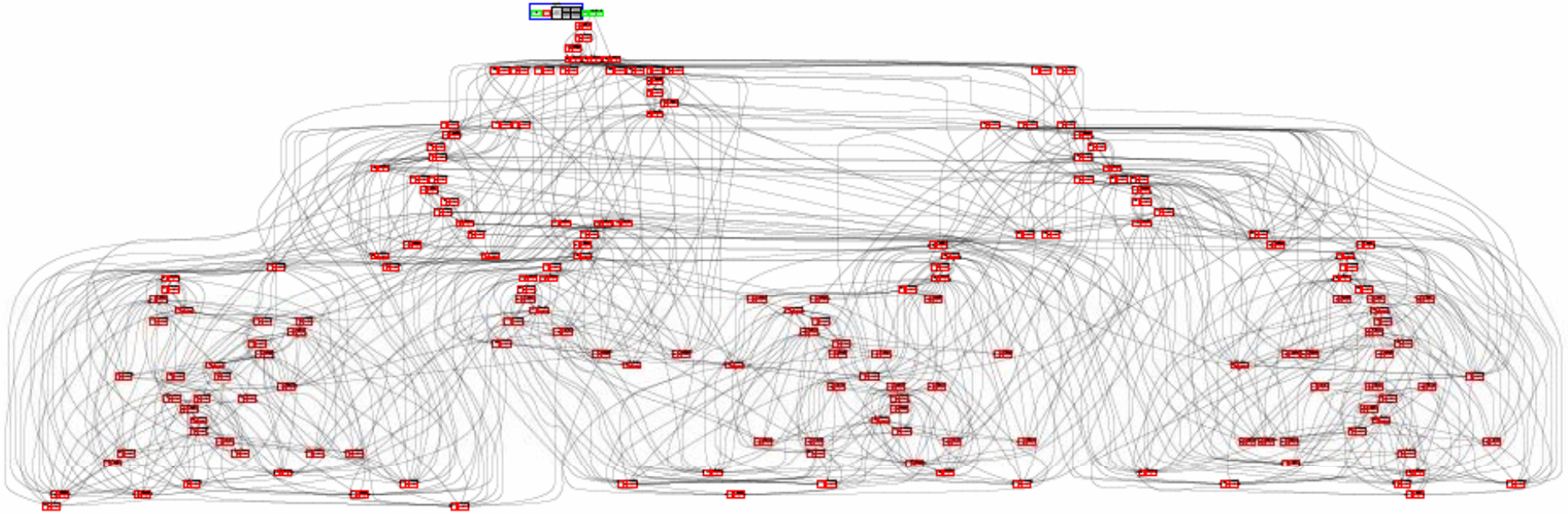
What is the counterstrategy?



Environment (Moore)

System (Mealy)

Example: Counterstrategy

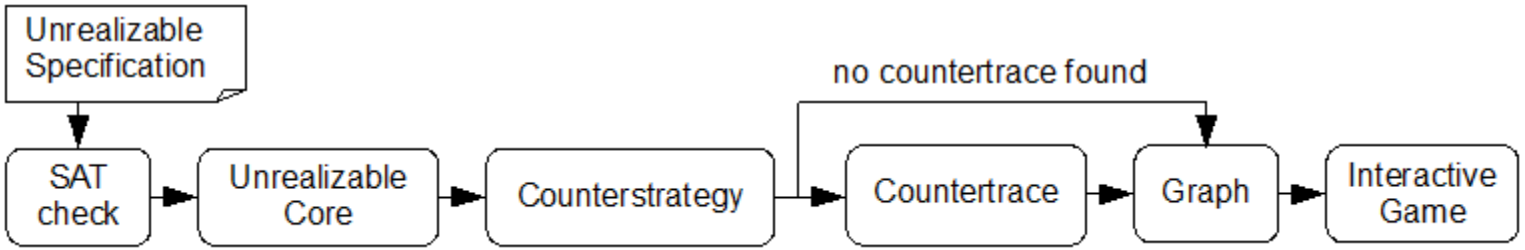


computation takes 775 s, graph has 183 nodes.

Useful Debugging Info

A goof explanation in three steps:

1. Simplify the specification (Cf. Cimatti et al 08)
   - Extend simplification: delta debug, remove variables
2. Simplify strategy to trace (heuristic)
3. Present strategy



Simplify Spec

- Let A be set of assumptions, let G be set of guarantees, let V be set of variables
- **Theorem 1**. Let $G' \subseteq G$. If $A \rightarrow G$ realizable then $A \rightarrow G'$ realizable.
  - Use theorem to minimize G
- **Previous approach** (Cimatti): Remove one guarantee at a time
- **Problem**: Many calls, and calls with large specs (long runtime)
- **Our Solution**: Use Delta Debugging (Zeller)
  - Attempts to remove half the guarantees at each step
- **Problem**: Smaller G = *larger* counterstrategy!
- **Our Solution:**
- **Theorem 2**. If $A \rightarrow G$ realizable then $(\forall V.A) \rightarrow (\exists V.G)$ realizable
  - Use Theorem 1 & 2 to minimize G and set of variables

Example: Minimize Spec

Result:

- Spec size reduces from 113 to 4
- 56 checks v. 136 for Cimatti
- Swift!

D[::]V

D[::]V

Resulting spec:

$G(\neg hready \rightarrow X \neg start)$

$G(\neg start \rightarrow (hmaster=3 \leftrightarrow X\ hmaster=3))$

$GF(hmaster=3 \vee \neg hbusreq3)$

Minimize Spec: Counterstrategy

Result:

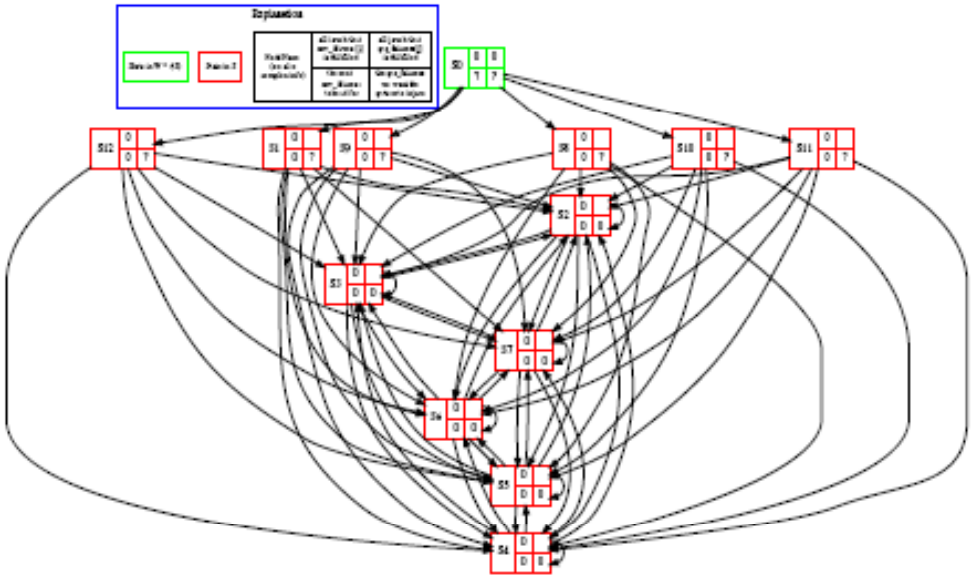
13 instead of 183 nodes

7 instead of 23 signals

14s. versus 775s.

Example: Variables that communicate bus status have been removed

Counterstrategy after simplification:



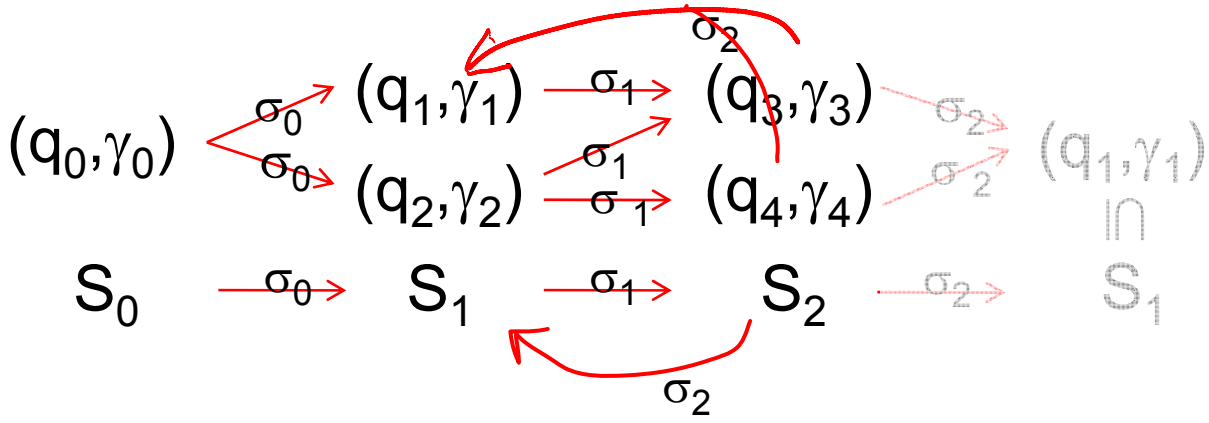
But we are still not happy…

Countertrace

- Would like to find a *countertrace* $x \in X^\omega$ such that $\neg\exists y \in Y^\omega: x||y \models \varphi$
- Such an input trace may not exist
  - $(y \leftrightarrow X\, x)$
- Computing it is hard:
  - Using automata, $\neg\exists y \in Y^\omega: x||y \models \varphi$ involves projection, complementation
- Use a heuristic
  - Existing trace may be not be found

Countertraces

- Given: game plus winning finite state strategy
- Strategy is map: $(q,\gamma) \rightarrow (\sigma_{out}, \gamma')$ new strategy state
  state in game, state in strategy, input
- How do I get a trace?
  - Produce a walk through product of game & strategy



Countertraces

- Guaranteed to be winning: sticks to strategy (including liveness constraints)
- Worst-case complexity horrible, very fast in practice.
- Finds very short countertraces
- Example: minimized spec: countertrace of length 1. (Raise 2 requests, never release bus.)
  - Overall reduction: from 183 states to trace of length 1

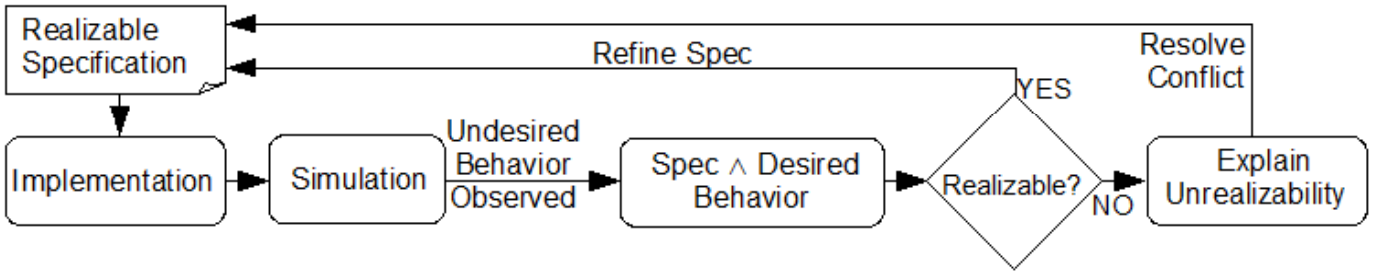
Incorrect Specs

- Specification/Synthesis flow
  1. Write spec
  2. Synthesize circuit
  3. Simulate result
  4. Change spec and go to 2

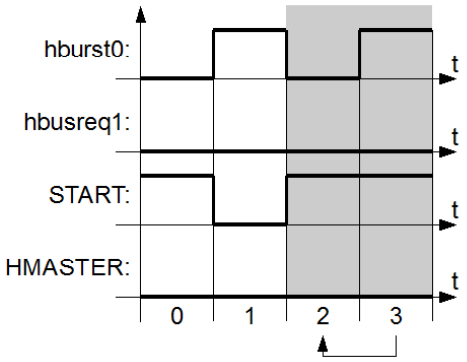
D[::]V
D[::]V

- What if the system behaves differently from expectation? Spec is wrong! It may be
  1. underconstrained – add a property (see coverage papers)
  2. incorrectly constrained

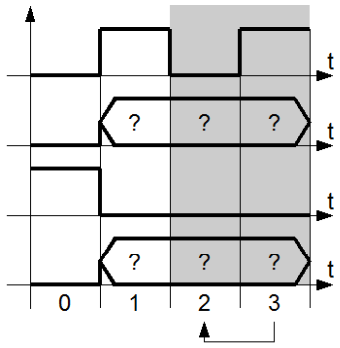
Incorrect Specs



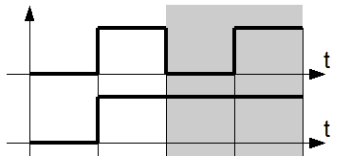
- Case 2: Incorrect constraints
  - Add expected behavior to spec – spec is now unrealizable
  - Compute reason for unrealizability.
  - Result: minimal set of contradictory requirements plus explanation



Undesirable trace



Desired behavior



Countertrace

Experimental Results

- Implemented in Anzu, a GR(1) synthesis tool
- Tried with AMBA, Genbuf
- All incorrect specs are satisfiable
- Minimizing spec
  - makes spec much shorter (removes 74-93%)
  - makes counterstrategy much simpler (removes 18-98% of nodes)
  - Crucial: remove variables
- Delta debugging is much faster than Cimatti (Saves up to 66% of calls)
- Finds short countertraces for all examples
- Useful:
  - Countertraces are great,
  - graphs are good,
  - games are not so useful.
- Threat to validity: tested using artificial specification errors.

Conclusions

- Specs are usually wrong
  - When doing synthesis: no design against which to check spec
  - Synthesis is wonderful method to debug spec
- Detecting fault is easy, finding it is had
- We promise help using mix of existing & new techniques
  - Counterstrategy
  - Spec simplification
  - countertraces