Causality in Models of Computation

Stefan Leue

University of Konstanz
Chair for Software and Systems Engineering

Stefan.Leue@uni-konstanz.de
http://sen.uni-konstanz.de/

Shonan, June 25, 2019

Joint Work with

Georgiana Caltais, University of Konstanz

Florian Leitner-Fischer, ZF

Martin Kölbl, University of Konstanz

Thomas Wies, New York University

Causal Analysis

- **Pearl‘s Causal Hierarchy (CACM, Vol. 62(3), March 2019)**
  - association
    - how does seeing X change my believe in Y?
  - intervention
    - what if I do X?
  - **counterfactuals**
    - was it X that caused Y?
    - what if I had acted differently?
- **On the "Rightness" of Causal Models**
  - comply with temporal order of cause and effect
  - be well-defined and well-motivated
  - be useful

Causality Analysis

- **Actual Causes**
  - safety evidence at design time
    - e.g., ISO 26262, DO-178C
  - fault localization and debugging at development time
    - e.g., delta debugging
  - failure forensics
    - e.g., FTA, WB-analysis
- **Blame / Responsibility**
  - not a primary concern here
- **Causality in Software-based Systems**
  - assume given System / Software Model
  - need to relate to models of Computation

Outline

1. **Models of Computation**
2. **Causation**
3. **Causality Checking (Dynamic Causal Analysis)**
4. **Analysis and Repair of Timed Systems (Static Causal Analysis)**
5. **Conclusion and Further Thoughts**

Outline

1. **Models of Computation**

2. **Causation**

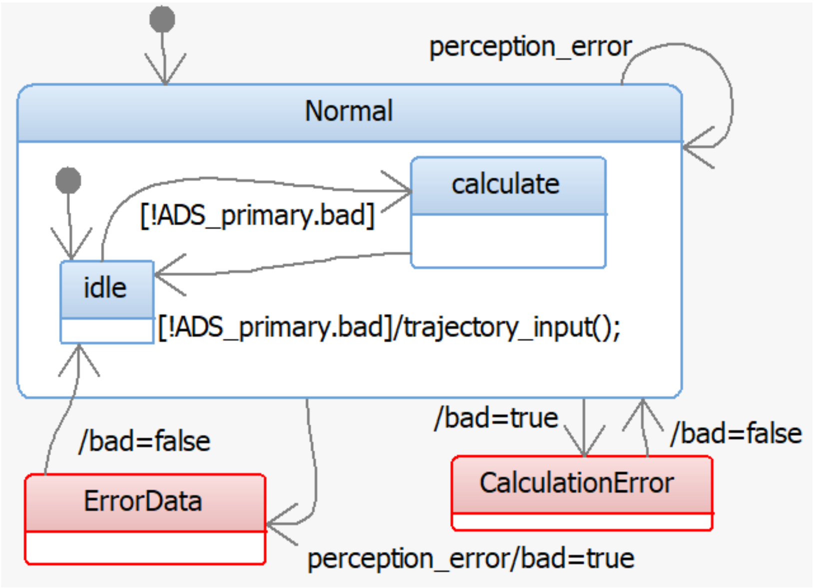
3. **Causality Checking (Dynamic Causal Analysis)**

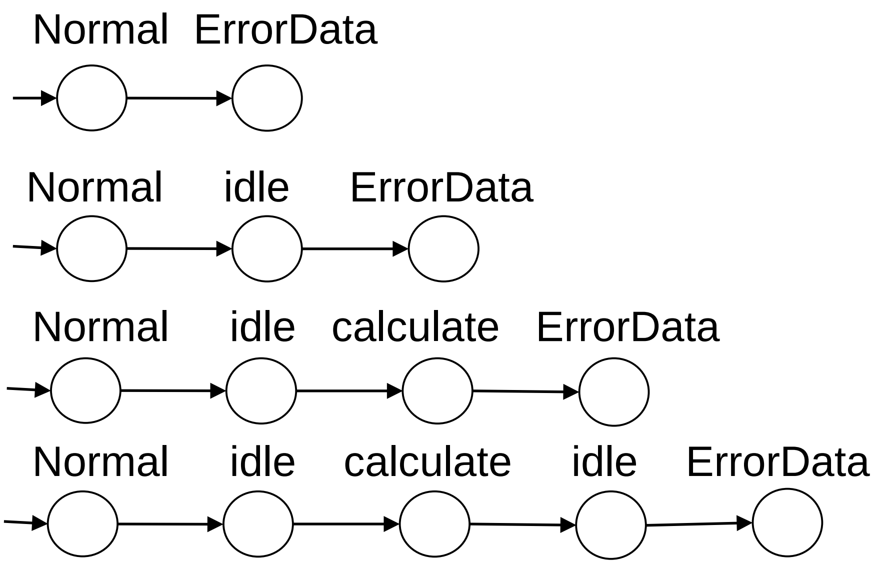
4. **Analysis and Repair of Timed Systems (Static Causal Analysis)**

5. **Conclusion and Further Thoughts**

Models of Computation

- **State Machine Models**
  - semantics as traces (for many models...)





  - – event orders matter

- **Other Models**
  - process algebras (transition systems, traces)
  - timed automata (timed transition systems, timed traces)
  - stochastic models (Markov chains, cylinder sets)

software engineering systems

Models of Computation

- **Characteristics**
  - closed world
    - syntactically closed
    - semantically closed
  - may contain nondeterminism
    - due to abstraction, environment modeling or concurrency
  - finite / infinite

Outline

1. Models of Computation

2. Causation

3. Causality Checking (Dynamic Causal Analysis)

4. Analysis and Repair of Timed Systems (Static Causal Analysis)

5. Conclusion and Further Thoughts

Causation

- **D. Hume, An Enquiry Concerning Human Understanding, 1748:**
  - "Yet so imperfect are the ideas which we form concerning it, that it is impossible to give any just definition of cause, except that it is drawn from something extraneous and foreign to it."
  - "Similar objects are always conjoined with similar."
  - "…therefore, we may define a cause to be *an object followed by another, and where all the objects, similar to the first, are followed by objects similar to the second*.
  - Or, in other words, *where, if the first object had not been, the second never had existed*."

quoted from:
D. Hume, An Enquiry Concerning Human Understanding, edited by E. Steinberg, 2nd ed., Hackett, 1993

Counterfactual Reasoning

- **D. Lewis, "Causation", *Journal of Philosophy*, 70: 556–67 (1973)**
  - causal dependence:
    - *"Where **c** (= cause) and **e** (= effect) are two distinct possible events, **e** causally depends on **c** if and only if,*
      - *if **c** were to occur **e** would occur;*
      - *and if **c** were not to occur **e** would not occur."*
  - often simplified as:
    - ***c** is causal for **e** if were **c** not to occur, then **e** would not occur either*
  - foundation of common software debugging techniques
    - e.g., delta debugging

software engineering systems

Consequences

- **Need for Alternate Worlds**
  - what-if analysis
    - had there been another course of action (= "world") in which the gate had been closed before the car entered the crossing, there would not have been an accident (= a "good" world)
  - "good" world
    - the effect (property violation) does not occur
  - "bad" world
    - the effect (property violation) occurs

Counterfactual Reasoning

- **"Naïve" Counterfactual Reasoning**
  - ***c*** *is causal for* ***e*** *if were* ***c*** *not to occur, then* ***e*** *would not occur either* (Hume, Lewis)
  - limitations
    - not suitable for complex logical causal relationships
      - conjunction
      - disjunction (over-determination)
      - preemption
    - no adaptation to models of computation (e.g., transition systems, traces)
    - missing explicit causality of orderings of multiple events
    - causality of non-occurrence of events not explicit

Halpern / Pearl Structural Equation Model (SEM)

- **Key Ideas**
  - exogenous and endogenous variables over arbitrary domains
  - events (variable changes) are represented by boolean variables
    - specified using structural equations
  - computes minimal boolean disjunction and conjunction of causal events
  - causal dependency of events represented by causal networks
  - reference

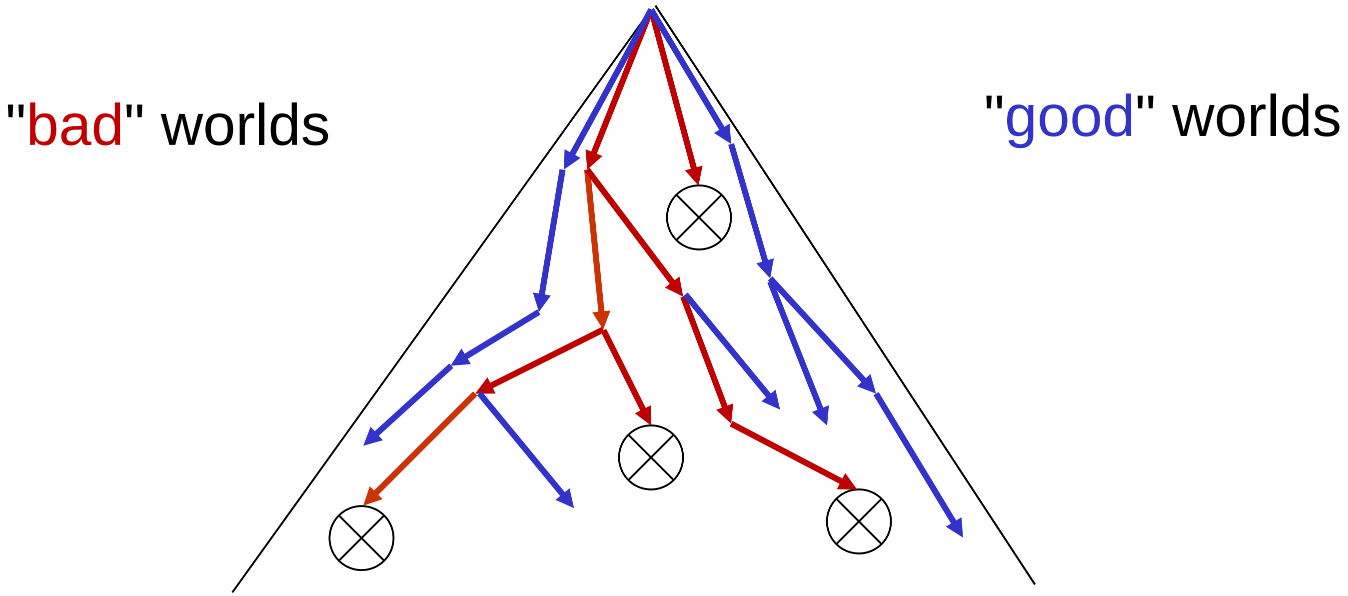
    J. Halpern and J. Pearl, "Causes and explanations: A structural-model approach. Part I: Causes," *The British Journal for the Philosophy of Science*, 2005.

Halpern / Pearl Structural Equation Model (SEM)

- **Actual Causality Conditions**
  - AC1: ensures that there exists a world where the boolean combination of causal events **c** and the effect **e** occur
  - AC2:
    1. if at least one of the causal events does not happen, the effect **e** does not happen
    2. if the causal events occur, the occurrence of other events can not prevent the effect
  - AC3: no subset of the causal events satisfies AC1 and AC2 (minimality)

Causality Analysis

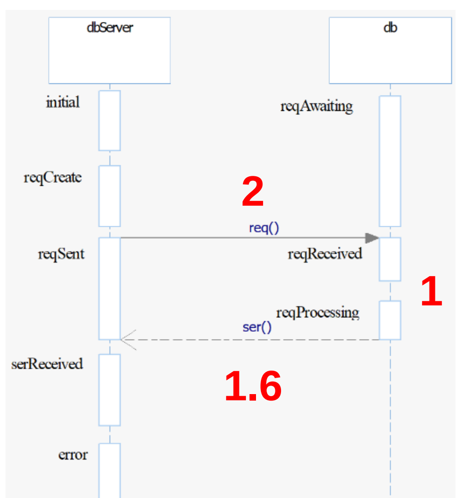
- **Causality Analysis In Models of Computation**
  - inspired by Lewis, Halpern-Pearl
  - derive actual causes from models
  - characteristics
    - preemption implicit
    - no (explicit) contingencies
    - non-determinism
  - algorithmic computation of alternate worlds

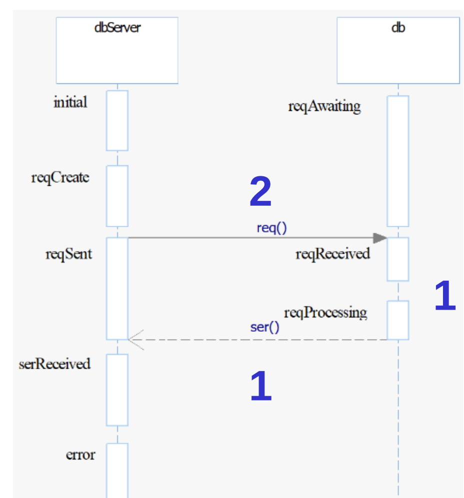


software systems engineering

Causality Analysis

- **Causality Analysis In Models of Computation**
  - counterfactual analysis based on (Finite) Models of Computation
    - dynamic (semantic) causes
      - non-deterministic branch choices during execution
        - "bad": train_approaching, car_crossing, gate_closing, train_crossing
        - "good": train_approaching, car_crossing, gate_closing, *car_leaving*, train_crossing
      - length of delay transitions along execution trace

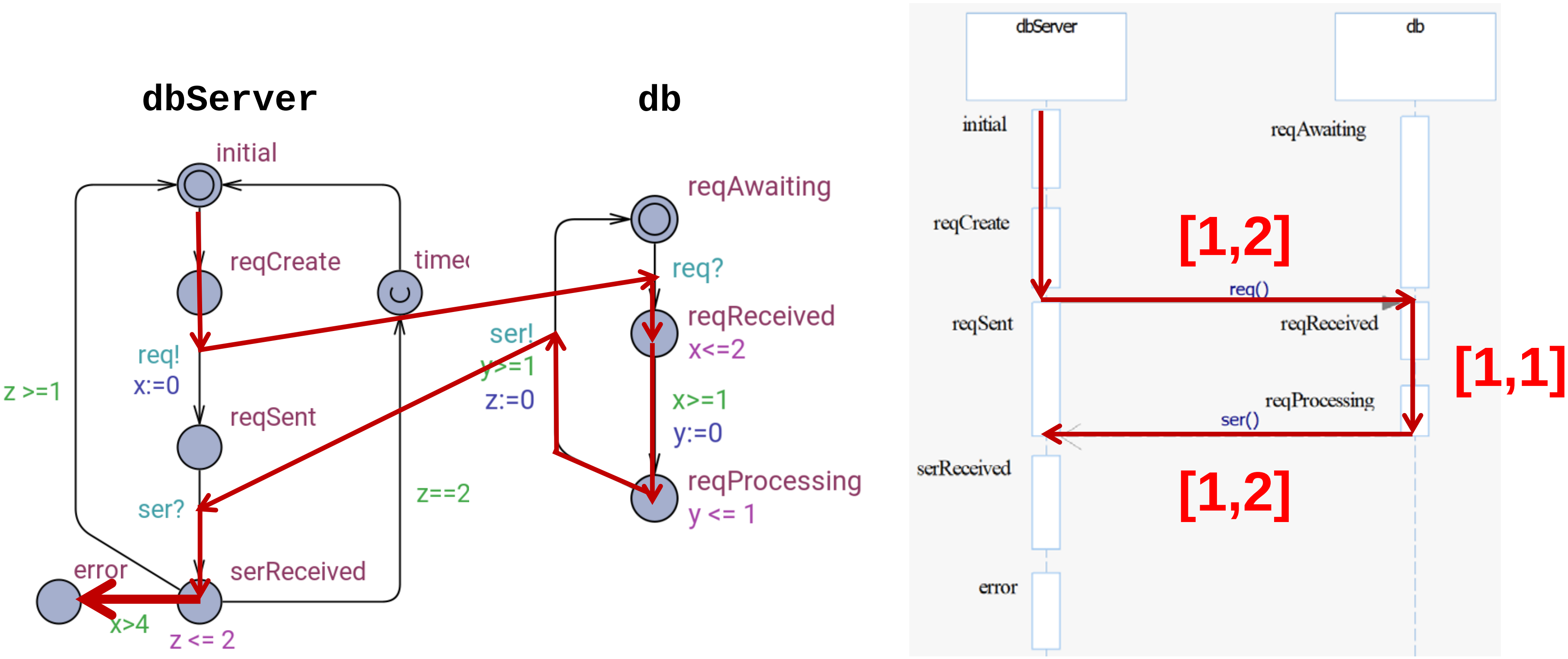




      - …

software systems engineering

- **Causality Analysis In Models of Computation**
  - counterfactual analysis based on (Finite) Models of Computation
    - static (syntactic) causes
      - synchronizations, timing bounds, ...



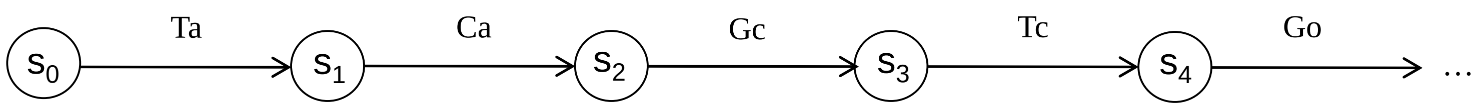
software systems engineering

Outline

1. Models of Computation
2. Causation
3. **Causality Checking (Dynamic Causal Analysis)**
4. Analysis and Repair of Timed Systems (Static Causal Analysis)
5. Conclusion and Further Thoughts

Causality Checking

- **Adaptation of Halpern/Pearl SEM [VMCAI 2013] [IJCCBS]**
  - reachability properties
    - e.g., $\Box \neg (Tc \wedge Cc)$
  - relate to concurrent computation models
    - transition systems
    - traces



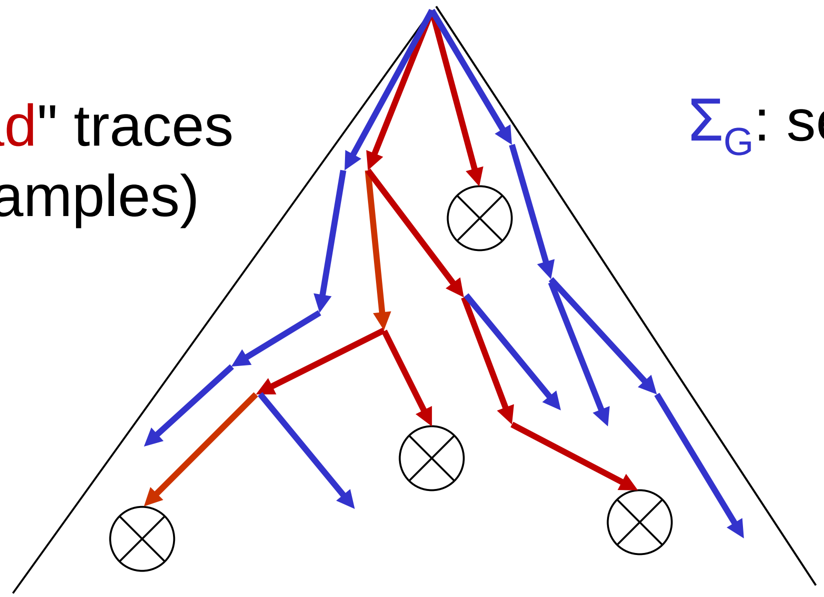
  - consider
    - ordering of events and
    - non-occurrence

    as potential actual causes
  - mechanization
    - algorithmic implementation

Causality Checking

- **Execution Traces Define (Alternate) Worlds**
  - explore state space with depth-first or breadth-first search
  - model check reachability of "bad" (hazardous) states
  - "bad traces"
    - all simple traces that lead to a state violating the property
  - "good traces"
    - all simple traces that do not reach a state violating the property

Σ_B: set of "bad" traces (counterexamples)



Σ_G: set of "good" traces

software systems engineering

Actual Cause Conditions for Causality Checking

- **AC1**
  - all events occurring along a trace that leads to a hazard
    - example candidate: σ = Ta, Ca, Gf, Cc, Tc
- **AC2**
  - **AC2.1**: if at least one of the presumed causal events does not occur, the hazard does not occur
    - σ satisfies this test since σ' = Ta, Ca, Gc, Tc does not lead to the hazard
  - **AC2.2**: if the additional occurrenc of another event prevents the hazard, then the non-occurrence of this event is to be considered causal
    - σ'' = Ta, Ca, Gf, Cc, Cl, Tc does not lead to the hazard, therefore
      - σ''' =Ta, Ca, Gf, Cc, ¬Cl, Tc is causal
      - σ = Ta, Ca, Gf, Cc, Tc is **not** causal
- **AC3**
  - no subtrace of the candidate trace satisfies AC1 and AC2

Actual Cause Conditions for Causality Checking

- **OC (Order Condition)**
  - for the same set of events
    - one order leads to the hazard
    - a different order does not lead to the hazard
  - example
    - order of events Cc, ¬Cl, Tc is important for causing hazard
    - relative order of Ta and Ca is not important, but they need to precede the above events

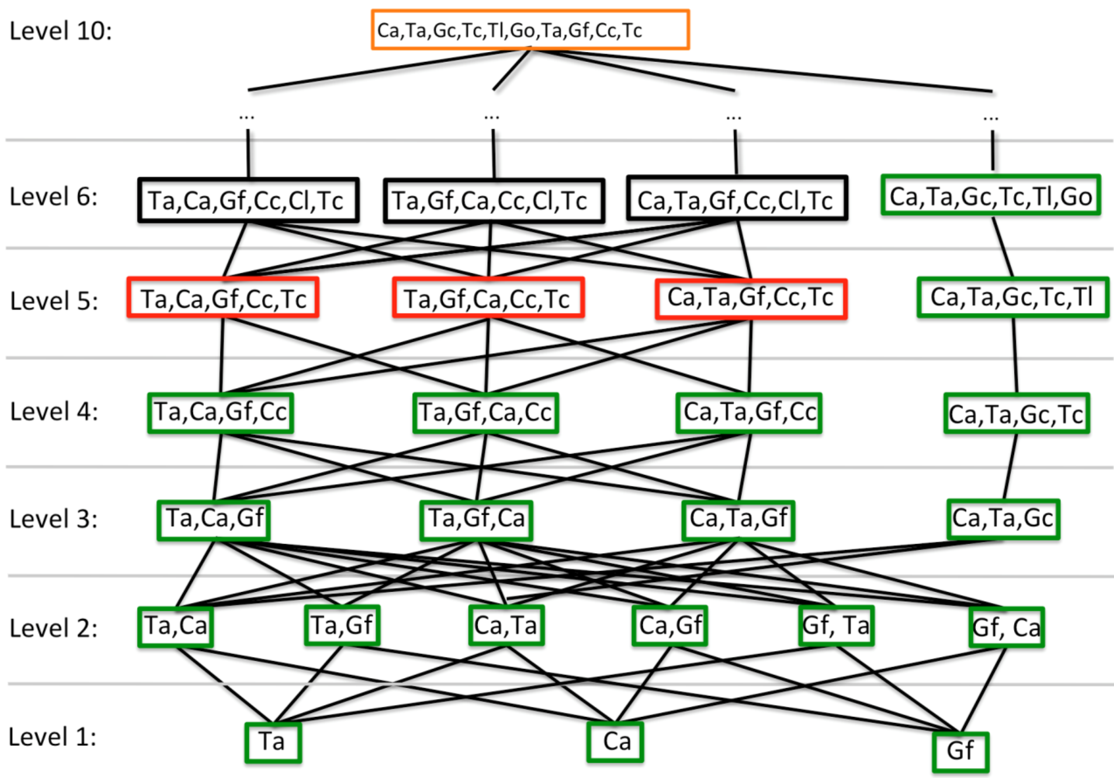
Algorithmics

- **Sub-Executions**
  - reduce checks for AC1-AC3 and OC1 to sub-execution tests
    - ordered and unordered sub-trace operators
  - (proofs in IJCCBS paper)
- **Implementation Variants**
  - on-the-fly
    - use DFS / BFS on the state space
      - store paths in an adequate data structure as you obtain them
        - subset graph

software engineering systems

Causality Checking

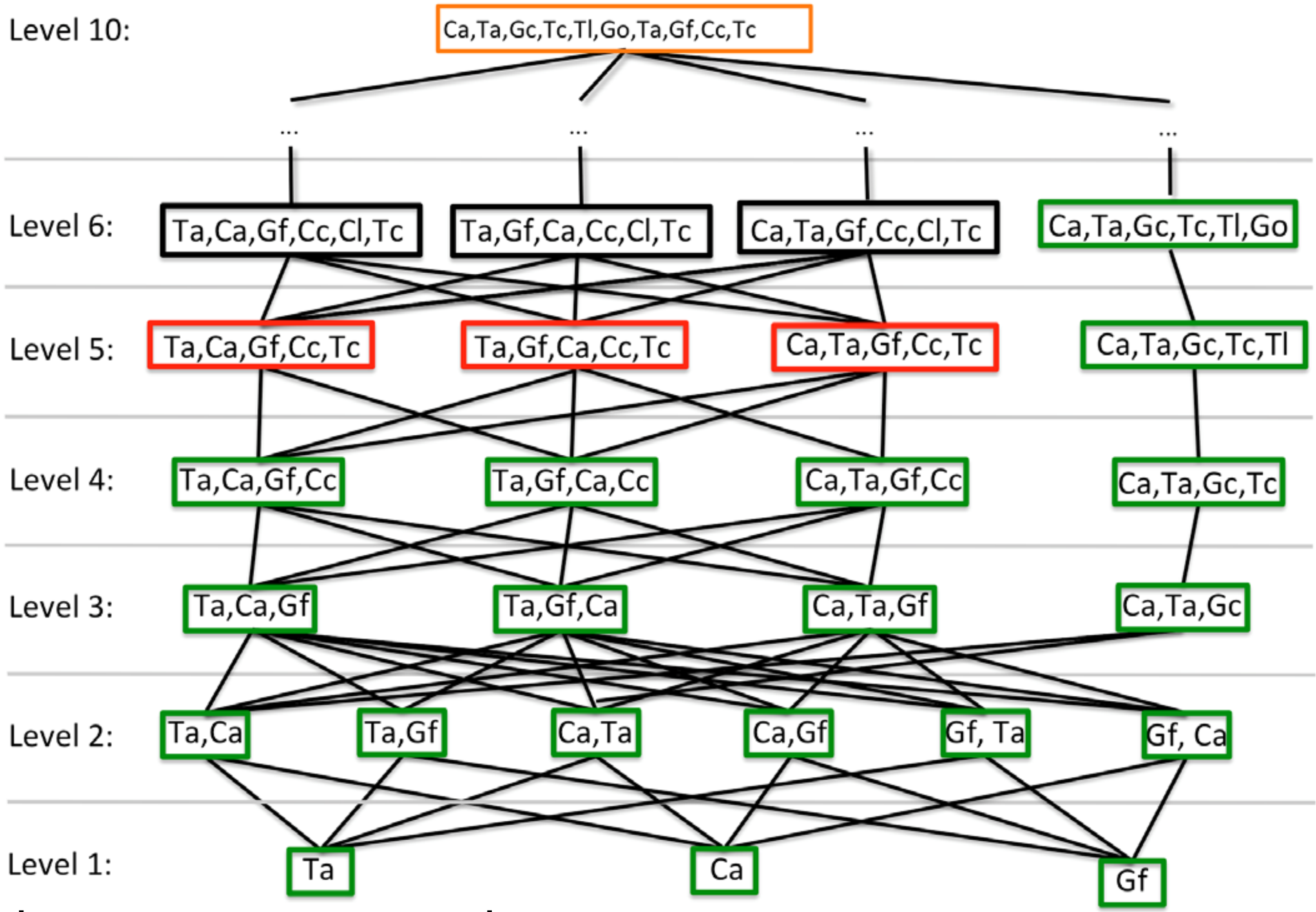
- **Algorithmic Implementation**
  - based on Breadth-First Search using SPIN model checking
  - subset graph construction



  - prefix tree data structure
  - parallelization

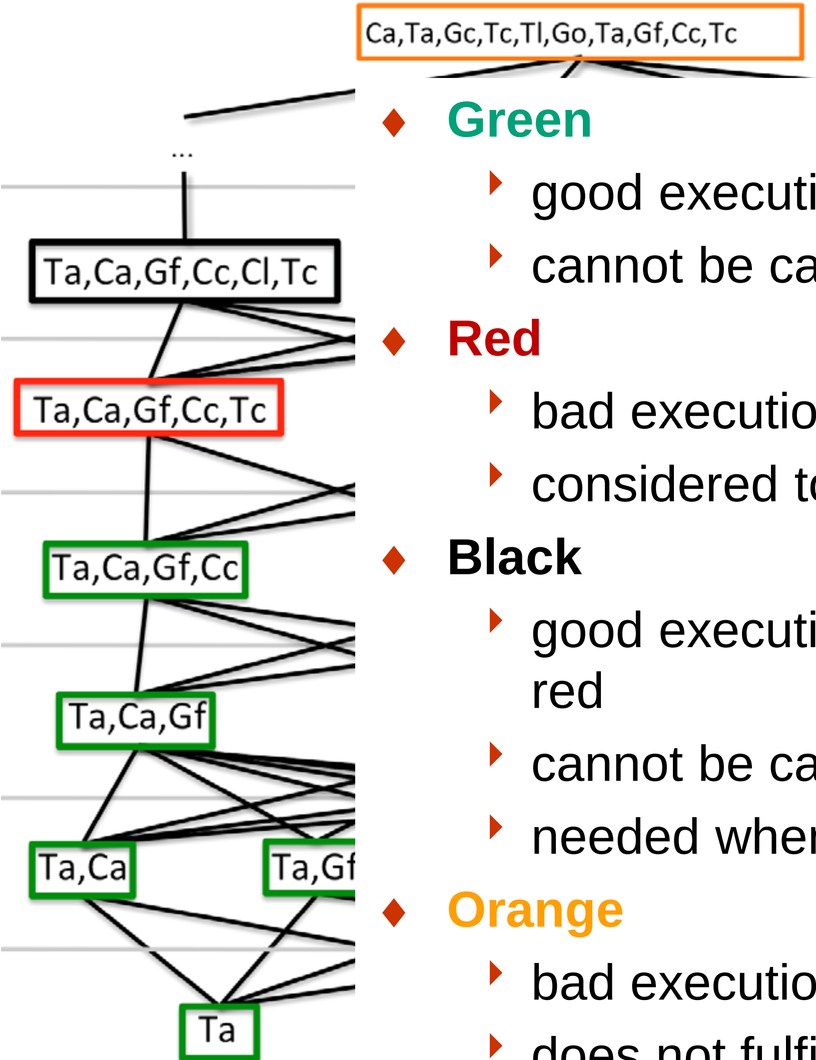
software systems engineering

Subset Graph



- nodes represent execution traces
- levels correspond to trace length
- sub- /super-traces on adjoining levels are connected
- color indicates potential causality

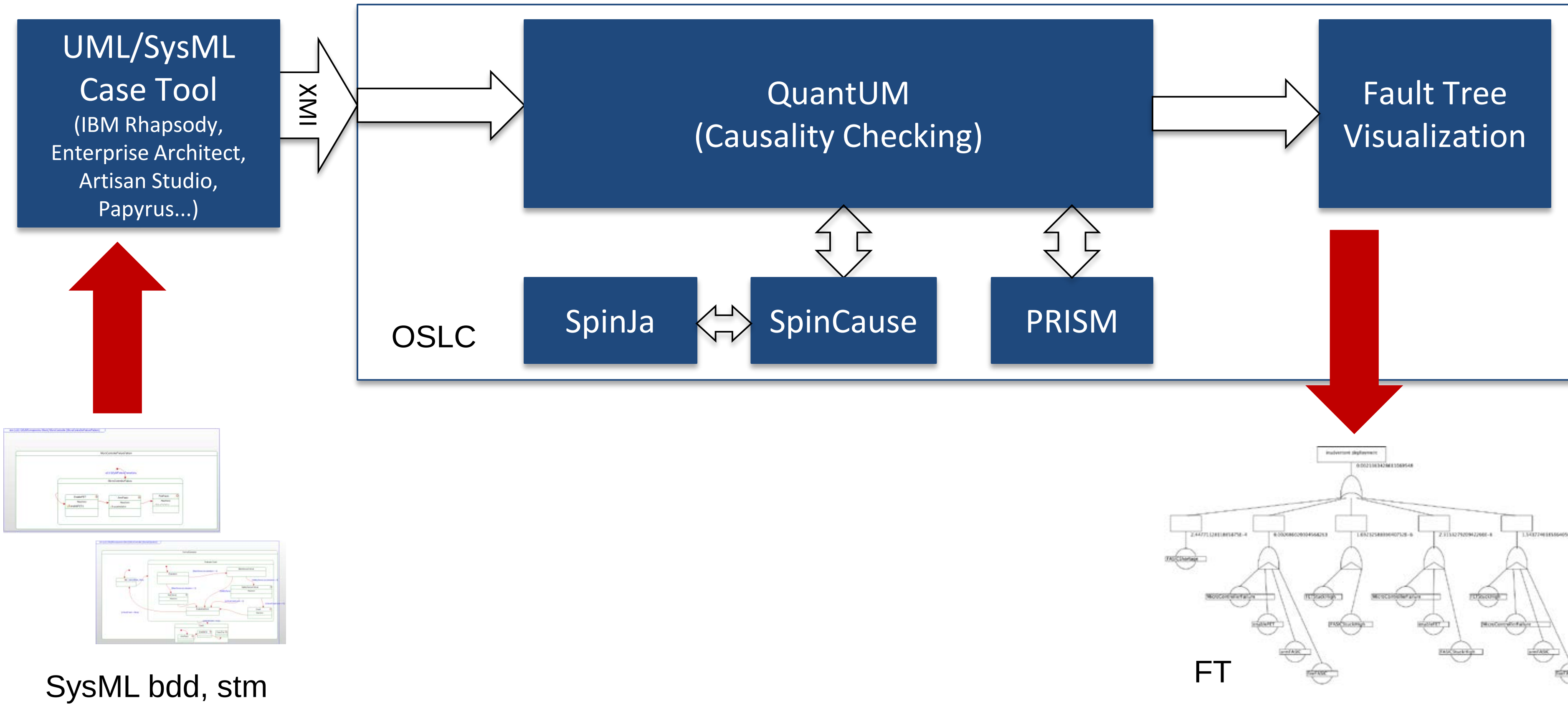
Subset Graph



- **Green**
  - good execution trace, all sub-traces are colored green
  - cannot be causal because they are good traces
- **Red**
  - bad execution trace, all sub-traces are colored green
  - considered to be causal
- **Black**
  - good execution trace, at least one sub-trace is colored red
  - cannot be causal since they are good traces
  - needed when checking condition AC2.2
- **Orange**
  - bad execution trace, at least one sub-trace colored red
  - does not fulfill the minimality constraint AC3 for being causal

Causality Checking

- **QuantUM Tool [SPIN 2014]**
  - support for automated safety analysis
    - e.g., safety evidence according to DO 178C / ISO 26262



QuantUM Case Study: Automated Driving System

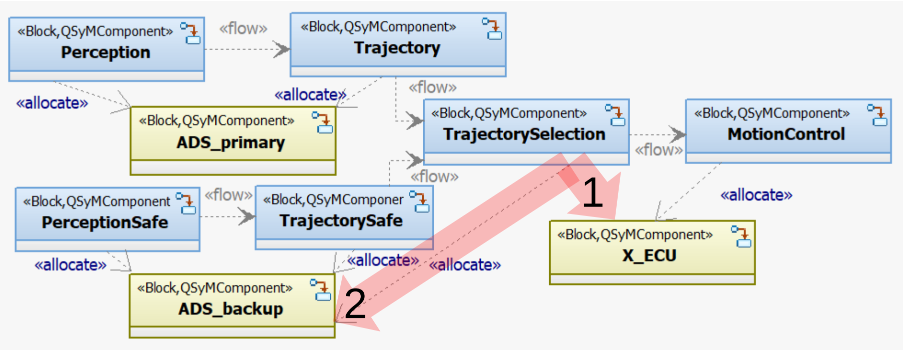
- **Definition of Safety Goals in Accordance with ISO 26262 [FMICS 2018]**
  - Safety Goal 1 (SG1)

    Ensure that the ADS provides driving information to the vehicle platform at any time.
  - Safety Goal 2 (SG2)

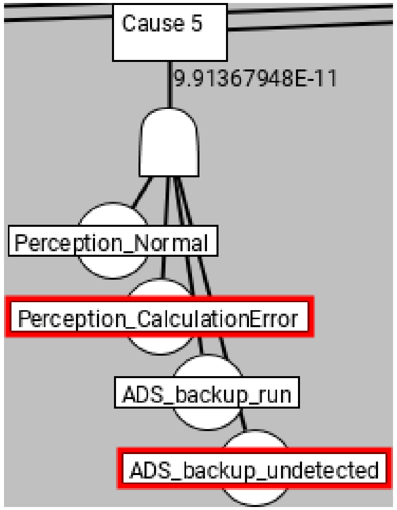
    Ensure that the emergency mode is enabled when a failure of the ADS occurs.
  - Safety Goal 3 (SG3)

    Ensure that the emergency mode of the ADS is available on demand for at least t_1 seconds.

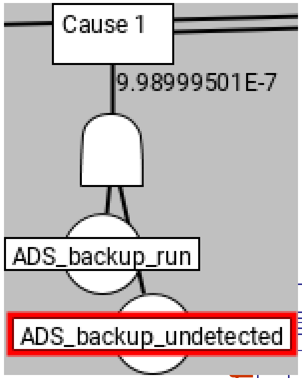
QuantUM Analysis Results



- **SG1, Architecture Variant 1, 1.320 · 10^{-8}**
  - **`ADS_backup_undetected`** co-occurs with **`Perception_CalculationError`**
  - relatively low probability
  - insignificant for SG1 violation



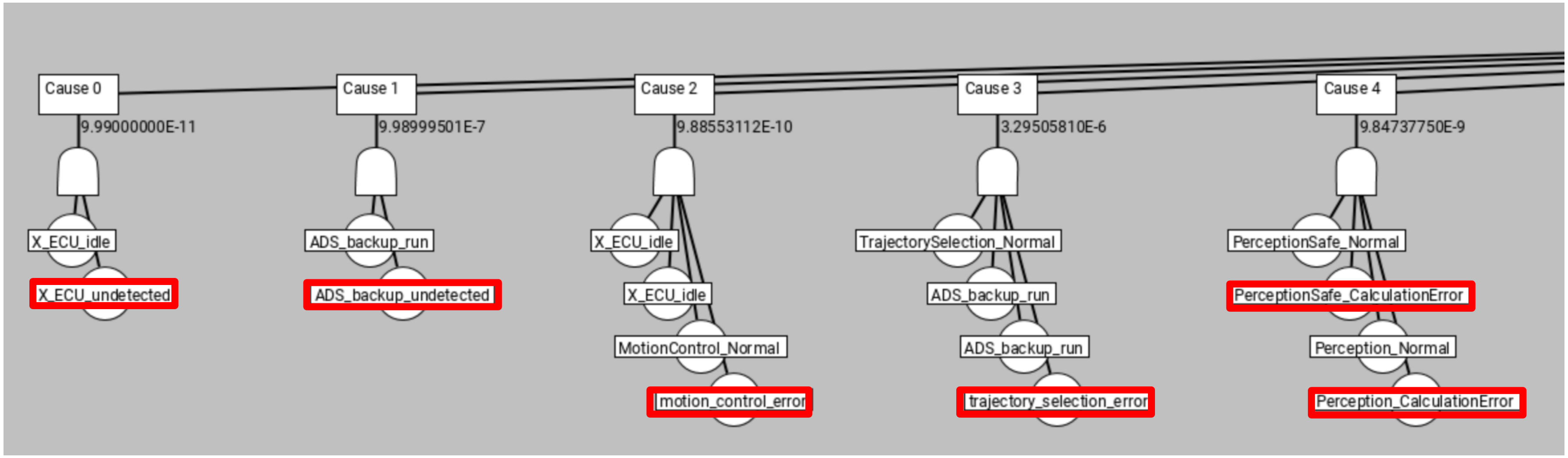
- **SG1, Architecture Variant 2, 4.306 · 10^{-6}**
  - **`ADS_backup_undetected`** occurs unconditionally
  - significant impact on SG1 violation probability
  - result of alternative software mapping



software engineering systems

QuantUM Analysis

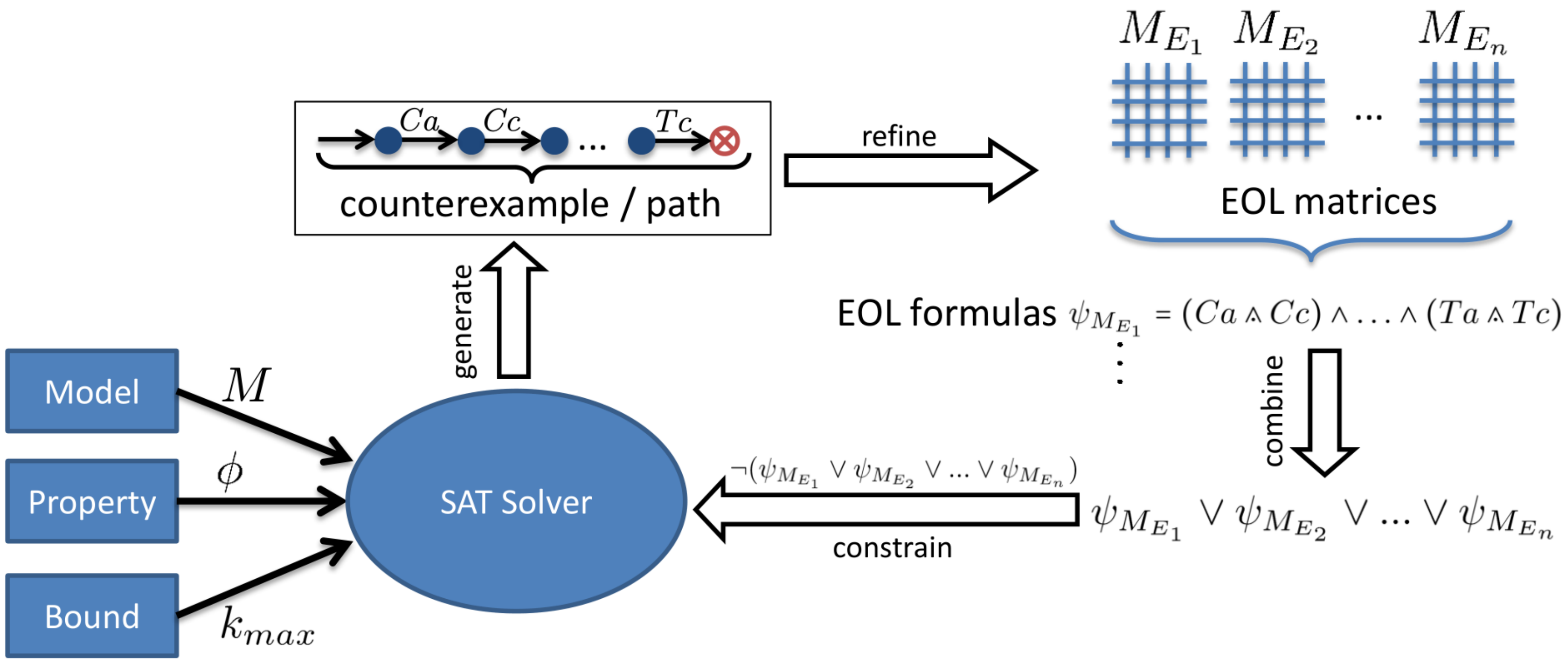
Single Point Failures



- easily recognizable in fault tree
- failure probability for each single point failure easily available

Symbolic Bounded Causality Checking

- **Algorithmic Scheme [SPIN 2015]**
  - computation of good and bad traces using bounded model checking
  - iterative model refinement and bound increase



Causality Checking

- **Further Extensions (In the Works...)**
  - backward search algorithm to compute all traces
    - deal with duplicate states during on-the-fly state space exploration
    - -> [ATVA 2019]
  - CC for general temporal properties
    - any linear time property expressible by omega-regular automata
    - requires computation of all lasso-shaped counterexamples
      - efficient algortihmic solution?
  - fully logical encoding of CC
    - enables BDD-based computation of actual causes

software systems engineering

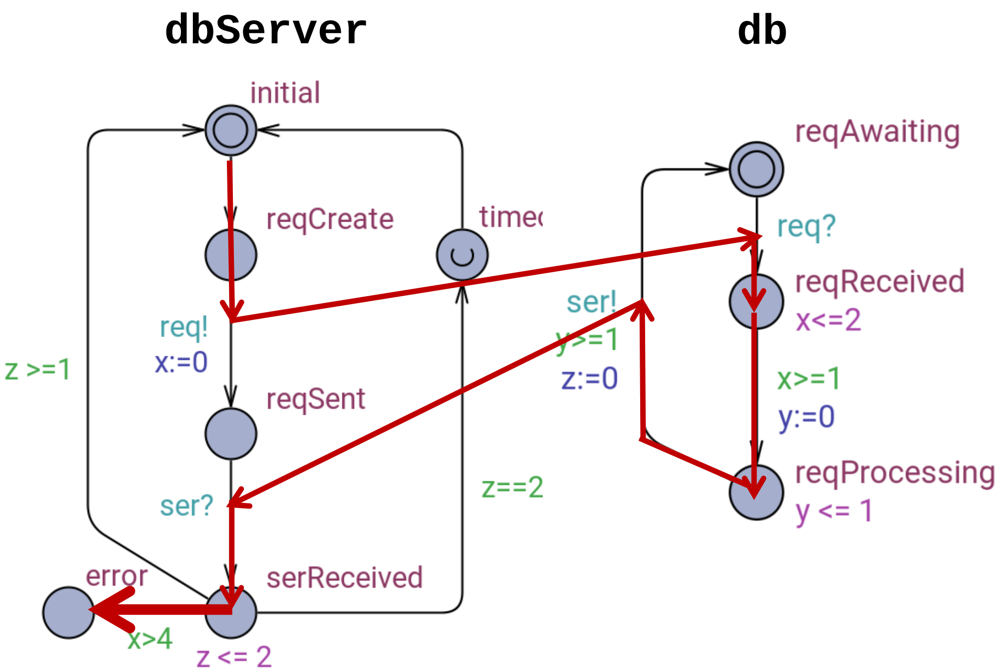
Outline

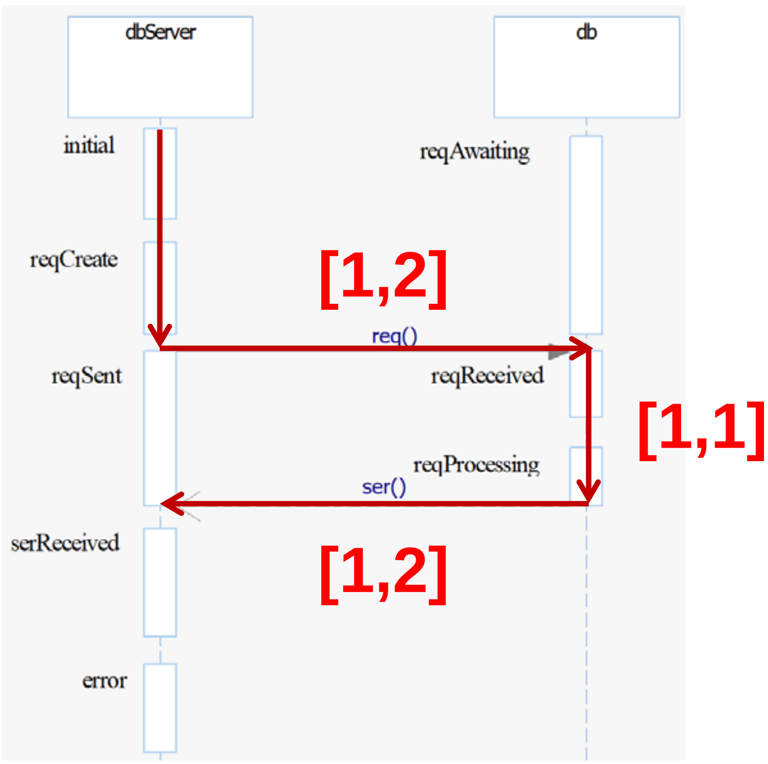
1. Models of Computation
2. Causation
3. Causality Checking (Dynamic Causal Analysis)
4. **Analysis and Repair of Timed Systems (Static Causal Analysis)**
5. Conclusion and Further Thoughts

software engineering systems

Timed Reachability Property

"maximum 4 time units between sending request and receiving service"





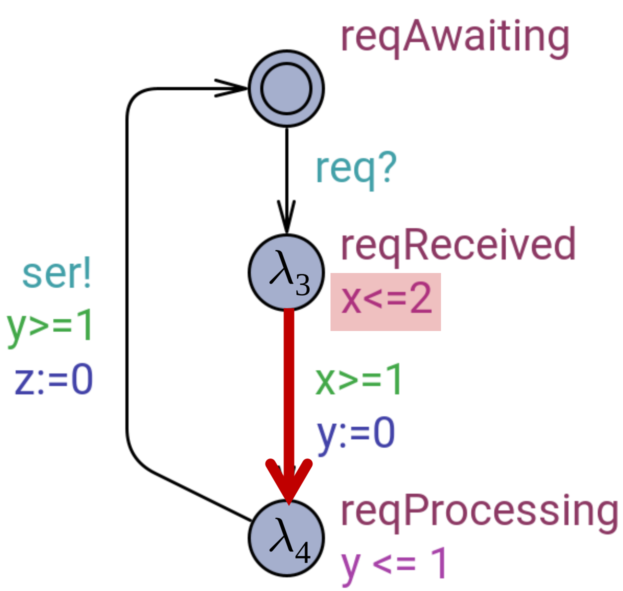
⇒ Timed Diagnostic Trace (Counterexample)

Analysis and Repair

- **Questions? [CAV 2019]**
  - **why** the property violation?
    - what are the actual causes?
  - can we **repair** the timed diagnostic trace?
    - what is the repair?
  - is the repair **admissible** in the context of the full NTA?
    - what does admissibility mean at all in this setting?

Formalization TDT

- **Logic Representation**
  - zone constraints not useful due to normalization, optimization
    - syntactic structures of constraints in NTA model need to remain visible
  - construction of a strongest postcondition symbolic semantics



$$c_{x,3} \leq 2 \land c_{x,3} + \delta_3 \leq 2$$

TDT Constraint System (TDTCS)

- **TDTCS**

$$\mathcal{C}_0 \equiv \bigwedge_{c \in C} c_0 = 0 \qquad \textit{(clock initialization)}$$

$$\mathcal{A} \equiv \bigwedge_{j \in [0,n]} \delta_j \geq 0 \qquad \textit{(time advancement)}$$

$$\mathcal{R} \equiv \bigwedge_{c \in reset_j,} c_{j+1} = 0 \qquad \textit{(clock resets)}$$

$$\mathcal{D} \equiv \bigwedge_{c \notin reset_j} c_{j+1} = c_j + \delta_j \qquad \textit{(sojourn time)}$$

$$\mathcal{I} \equiv \bigwedge_{(\beta,\sim) \in ibounds(c,l_j)} c_j \sim \beta \wedge c_j + \delta_j \sim \beta \qquad \textit{(location invariants)}$$

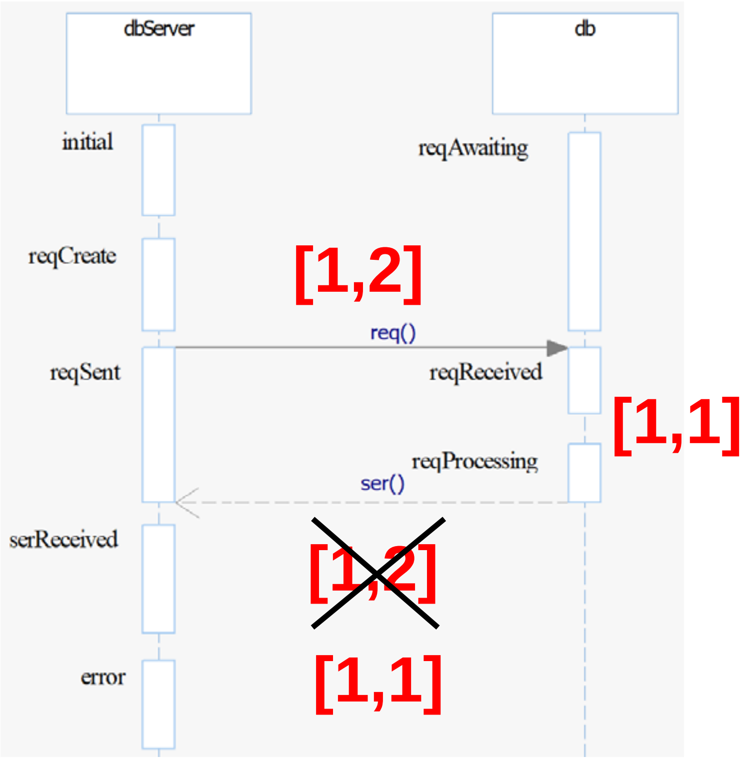
$$\mathcal{G} \equiv \bigwedge_{(\beta,\sim) \in gbounds(c,\theta_j)} c_j + \delta_j \sim \beta \qquad \textit{(transition guards)}$$

$$\mathcal{L} \equiv @l_n \wedge \bigwedge_{l \neq l_n} \neg @l \qquad \textit{(location predicates)}$$

software engineering systems

Repair

- **Non-Violation of Timed Reachability Property**
  - for one given trace
  - π_i = ({**serReceived**}, $\{x>4\}$, {**error**})



software engineering systems

- **Bound Variation**
  - Bound Variation Modified TDTCS (BVTDTCS)
    - Location invariant constraints $\mathcal{I}^{bv}$

$$c_{x,3} \leq 2 + \beta^{bv}{}_{x,3,1} \wedge c_{x,3} + \delta_3 \leq 2 + \beta^{bv}{}_{x,3,1}$$

    - Transiton guard constraints $\mathcal{G}^{bv}$

$$c_{x,3} + \delta_3 \geq 1 + \beta^{bv}{}_{x,3,2}$$

    - $\mathcal{T}^{bv} \equiv \mathcal{C}_0 \wedge \mathcal{N} \wedge \mathcal{A} \wedge \mathcal{R} \wedge \mathcal{D} \wedge \mathcal{U} \wedge \mathcal{I}^{bv} \wedge \mathcal{G}^{bv} \wedge \mathcal{Z}^{bv}$

Repair

- **Clock Bound Repair Constraint System (CBRCS)**
  - there is a model (= assignment of δ_j values) for $\mathcal{T}^{bv}$

$$(\exists c_i, \lambda_j, \beta^{bv}_{c_{ij},l})(\mathcal{T}^{bv})$$

  - for all concretizations, $\mathcal{T}^{bv}$ implies error guard violated

$$(\forall c_i, \delta_j)(\mathcal{T}^{bv} \Rightarrow \neg\Phi)$$

  - force all bound variation variables to 0

$$\mathcal{F}^{bv} : \bigwedge_{c_i,\lambda_j,(\beta_l,\sim_l)\in(ibounds(c_i,\lambda_j)\ \cup\ gbounds(c_i,\lambda_j))} \beta^{bv}_{c_{ij},l} = 0$$

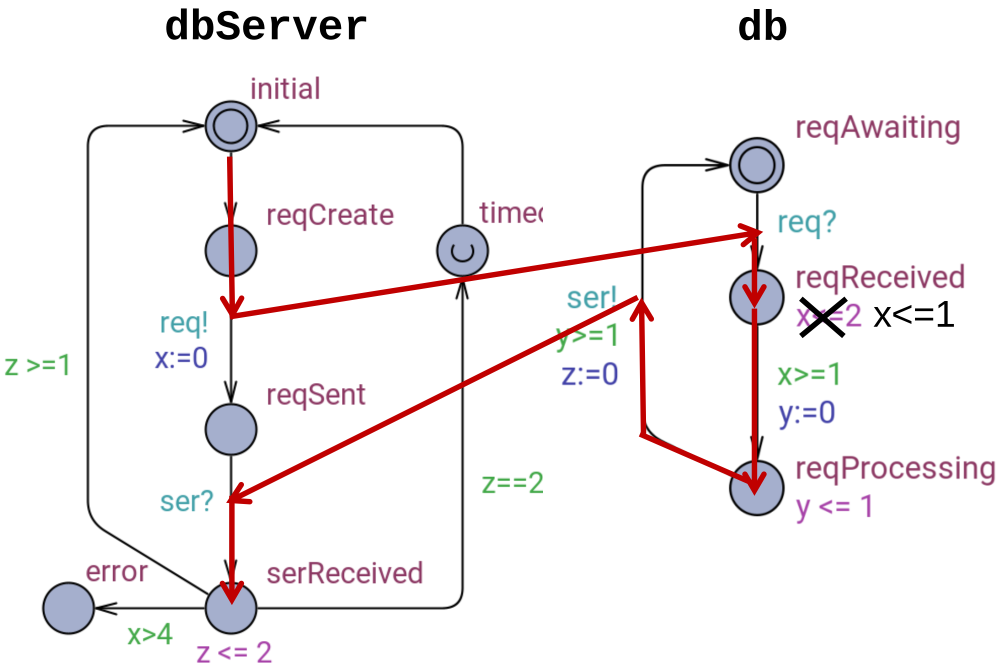
  - CBRCS

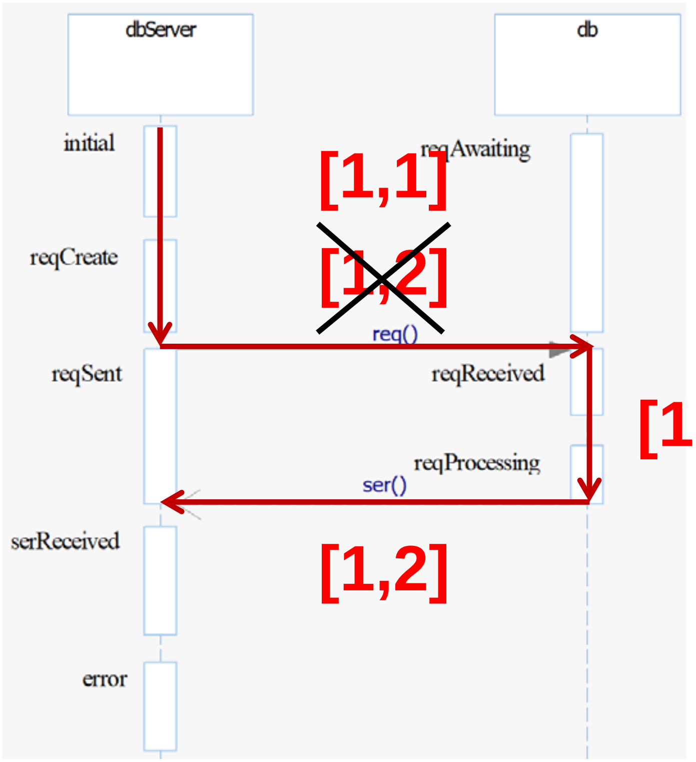
$$\mathcal{V}^{bv} \equiv (\exists c_i, \lambda_j, \beta^{bv}_{c_i,l,k})(\mathcal{T}^{bv} \wedge ((\forall c_i, \lambda_j)(\mathcal{T}^{bv} \Rightarrow \neg\Phi)) \wedge \mathcal{F}^{bv})$$

    - unsat
    - MaxSMT: soft-assert $\mathcal{F}^{bv}$
      - yields non-zero values for some β^{bv}
      - $\Rightarrow$ repair

Repair

- **Example**
  - Z3: **`(define-fun _bv_x3_1 () Int (- 1))`**
  - $\beta^{bv}{}_{x,3}$ = -1





Repair

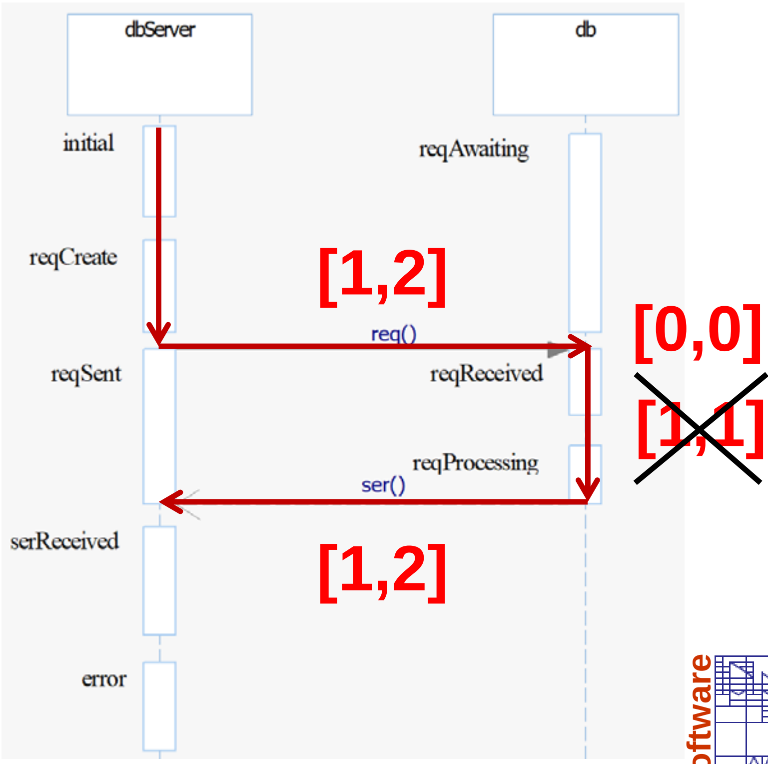
- **Computing All Minimal Repairs**
  - iterative scheme

    $\mathcal{F}_i^{bv}$: bound variation variables forced to 0 in iteration i

    $\hat{\beta}_i^{bv}$: bound variation variables set to non-zero value in iteration i

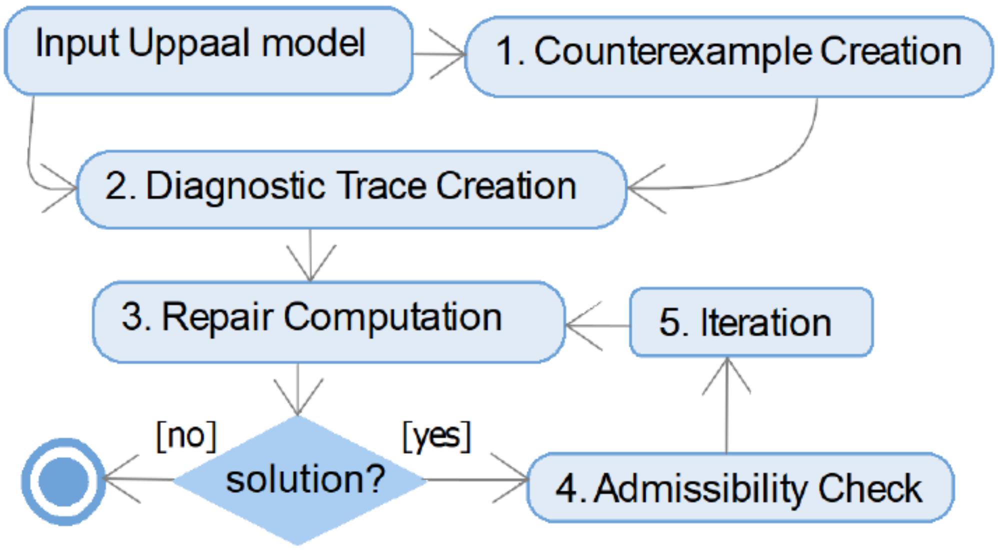
$$\mathcal{F}_{i+1}^{bv} \equiv \mathcal{F}_i^{bv} \wedge \bigwedge_{\hat{\beta}_i^{bv}} \hat{\beta}_i^{bv} = 0$$

  - example



TarTar Tool

- **Architecture**
  1. UPPAAL model checking on Π
  2. UPPAAL-TDT to smtlib2
  3. Z3: compute repair yielding N
     – quantifier elimination on $(\forall c_i, \delta_j)(\mathcal{T}^{bv} \Rightarrow \neg\Phi)$
  4. admissibility check
     1. Ltsmin / Opaal: untimed BAs for N, modified N‘
     2. LearnLib: compare $\mathcal{L}_\mu$(N) = $\mathcal{L}_\mu$(N')



Quantitative Experimental Evaluation

- **Systematic Fault Seeding**
  - input: model without property violations
  - mutate a single guard/invariant constraint by {-10, -1, +1, +0.1 Max, +Max}
  - check if mutated model violates property
    - created 60 TDT by mutation
- **Results / TarTar**

Model	# Seed	# TDT	T_{UP}	*Len.*	*# Rep.*	*# Adm.*	*# Sol.*	T_{QE}	T_R	SD_R	T_{Adm}	*# Var.*	*# Con.*
repaired db Fig. 2	35	6	0.005s	4	12	12	6	0.217s	0.024s	0.001	2.331s	25	40
CSMA/CD [17]	90	6	0.005s	2	36	16	6	0.110s	0.024s	0.000	2.369s	16	36
Elevator [28]	35	3	0.003s	1	6	6	3	0.093s	0.024s	0.000	2.211s	6	16
Viking	85	3	0.009s	18	6	6	3	0.304s	0.053s	0.000	2.809s	120	140
Bando [29]	740	12	0.250s	279	26	24	12	17.490s	6.302s	1.707	3.847s	1,156	2,441
Pacemaker [19]	240	7	0.016s	9	28	16	7	1.389s	0.174s	0.013	2.782s	114	290
SBR [23]	65	14	0.035s	81	24	16	8	11.071s	0.908s	0.219	26.834s	253	401
FDDI [29]	100	9	0.006s	5	36	30	9	0.118s	0.031s	0.000	2.367s	55	84

  - computed at least one repair for 56 (93%) out of 60 TFTs
  - computed at least one **admissible** repair for 54 (90%) of the TDTs
  - $T_R + T_{QE}$ depends on Len and $\#\text{clocks}$

software systems engineering

Outline

1. Models of Computation
2. Causation
3. Causality Checking (Dynamic Causal Analysis)
4. Analysis and Repair of Timed Systems (Static Causal Analysis)
5. **Conclusion and Further Thoughts**

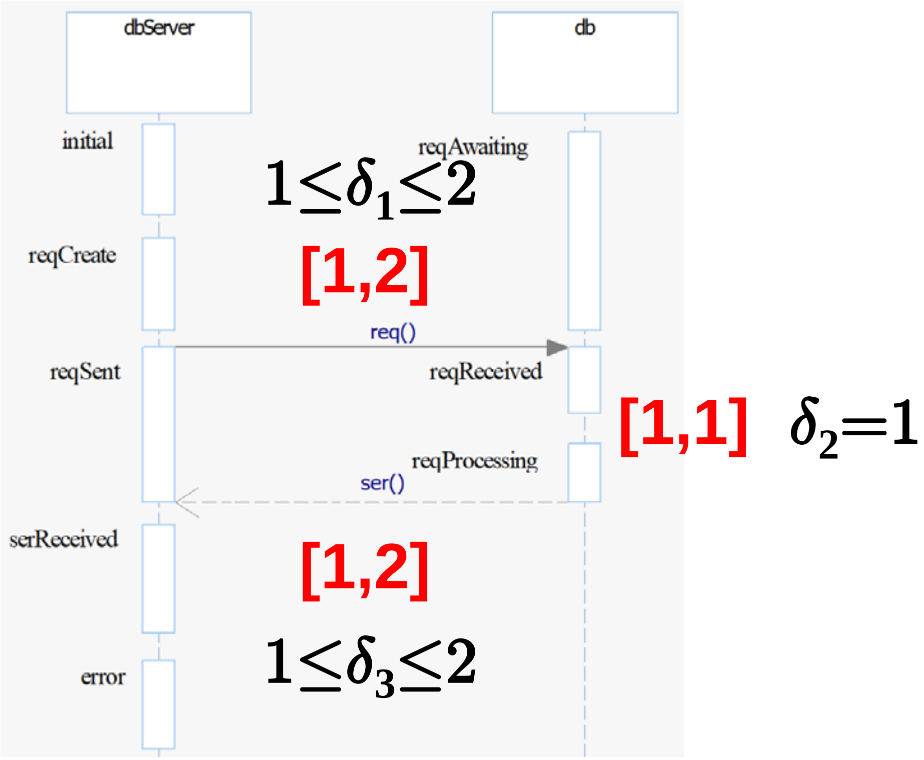
Further Thoughts

- **Causality and Real Time**
  - counterfactuals
    - what are alternate worlds?
    - what are closest alternate worlds?
  - actual causes
    - what are the a.c. for real-time property violations?
      - values in a dense domain
  - causality checking for real time
    - trace based?
    - constraint based?
  - answers from "first principles" of causality?

- **Applicability**
  - debugging
  - fault forensics
  - design space exploration

Further Thoughts

- Dynamic Causal Analysis of Timed Systems



- establishing a counterfactual trace for delay $\geq$ 4
  - $\delta_1 \neq 2 \vee \delta_3 \neq 2 \vee \delta_1 + \delta_3 > 3$
- minimality
  - delay value or number of delays to constrain?

Further Musings

- **Repair**
  - in the context of counterfactual causality
    - repair $\equiv$ elimination of all actual causes?

Selected Publications

- [SAFECOMP 2011] M. Kuntz, F. Leitner-Fischer and S. Leue: *From Probabilistic Counterexamples via Causality to Fault Trees*, Proc. SAFECOMP 2011), Springer LNCS, 2011.
- [VMCAI 2013] F. Leitner-Fischer and S.Leue: *Causality Checking for Complex System Models*, Proc. VMCAI 2013, Springer LNCS, 2013.
- [IJCCBS] F. Leitner-Fischer and S. Leue: *Probabilistic Fault Tree Synthesis using Causality Computation*, International Journal of Critical Computer-Based Systems, Vol. 4, No. 2, pp.119–143, 2013.
- [SPIN 2014] F. Leitner-Fischer and S. Leue: *SpinCause: A Tool for Causality Checking*. Proc. SPIN 2014, ACM, 2014.
- [SPIN 2015] A. Beer, S. Heidinger, U. Kühne, F. Leitner-Fischer and S. Leue: *Symbolic Causality Checking Using Bounded Model Checking*. Proc. SPIN 2015, Springer LNCS, 2015.
- [CREST 2016] G. Caltais, S. Leue, M. Mousavi*: (De-)Composing Causality in Labeled Transition Systems*. CREST@ETAPS 2016: 10-24
- [FMICS 2018] M. Kölbl, S. Leue: *Automated Functional Safety Analysis of Automated Driving Systems*. FMICS 2018: 35-51.
- [ATVA 2019] M. Kölbl and S. Leue: *An Efficient Algorithm for Computing Causal Trace Sets in Causality Checking*. In: Proc. ATVA 2019, LNCS, Springer. 2019. To appear.
- [CAV 2019] M. Kölbl, S. Leue, T. Wies: *Clock Bound Repair for Timed Traces*, Proc. CAV 2019, LNCS, Springer. 2019. To appear.

Universität
Konstanz

PhD / PostDoc
Position available!
full-time, TVL-13

Thank you
for your attention.
Questions?