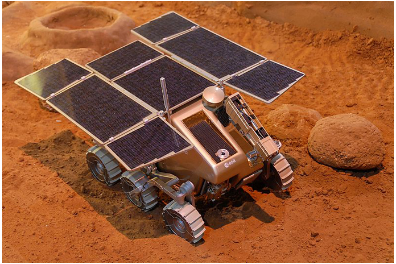
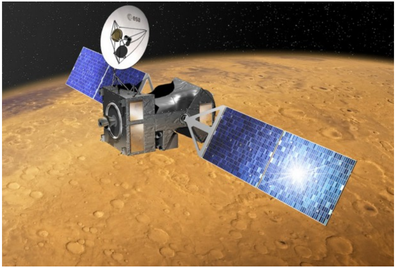
A Formal Foundation of FDI Design via Temporal Epistemic Logic

Marco Gario
gario@fbk.eu

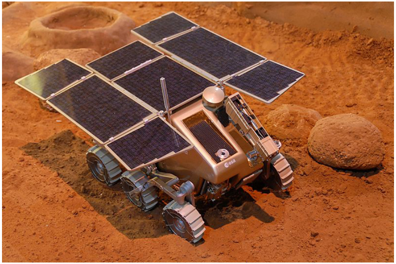
Fondazione Bruno Kessler
University of Trento

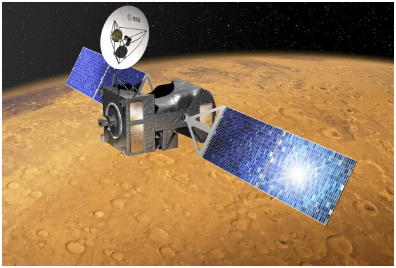
2016-03-03



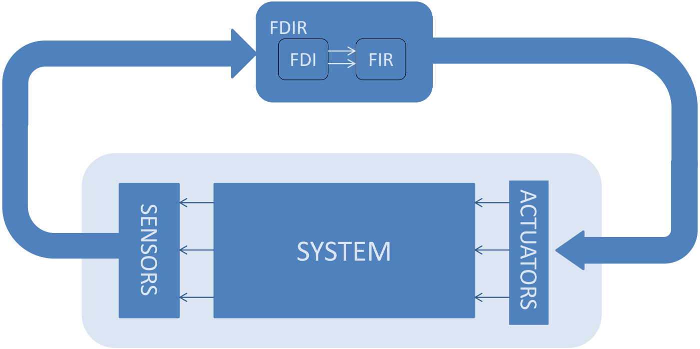


On-Board Autonomy





On-Board Autonomy: Faults?



Fault Detection, Identification and Recovery

FDI(R)

Fault Detection, Identification and Recovery

FDI(R)

FDIR development is performed **late** in the system development life-cycle when changes are **costly** or **impossible**

Conservative or **over-blown** designs:

- Monitor everything
- Disable FDI during critical situations

Certification is difficult

How to design an FDI ?

- Limited observability (Sensors)
- Interaction of multiple faults and nominal operations

How to specify an FDI?

How to specify an FDI?

There is **no standard** way of specifying an FDI!

Requirements are usually **prescriptive** (e.g. thresholding)

⇒ **Hard** to certify that objectives are met!

Ultimate Goals

- Specification of FDI:
  - **Early-validation** of the FDIR design
  - Simplification of the **certification** process
- Verification of FDI:
  - Higher **dependability** of systems
  - **Reduction of costs** in terms of design effort, implementation and **reuse** of existing FDIR components

Contributions

1. Formal FDI design: [AAAI13, DX13, TACAS14, LMCS15]
   - Alarm Specification Language (ASL_K) & Formal grounding on Temporal Epistemic Logic KL_1
   - Diagnosability testing for ASL_K
   - Pareto Optimal Sensor Placement [FMCAD14]
   - Synthesis of FDI components

Contributions

1. Formal FDI design: [AAAI13, DX13, TACAS14, LMCS15]
   - Alarm Specification Language (ASL_K) & Formal grounding on Temporal Epistemic Logic KL_1
   - Diagnosability testing for ASL_K
   - Pareto Optimal Sensor Placement [FMCAD14]
   - Synthesis of FDI components
2. Validation of Timed Failure Propagation Graphs based on Satisfiability Modulo Theory [AAAI15]

Contributions

1. Formal FDI design: [AAAI13, DX13, TACAS14, LMCS15]
   - Alarm Specification Language (ASL_K) & Formal grounding on Temporal Epistemic Logic KL_1
   - Diagnosability testing for ASL_K
   - Pareto Optimal Sensor Placement [FMCAD14]
   - Synthesis of FDI components
2. Validation of Timed Failure Propagation Graphs based on Satisfiability Modulo Theory [AAAI15]
3. First model-checking approach for KL_1 over infinite state transition systems [AAMAS16]

Contributions

1. Formal FDI design: [AAAI13, DX13, TACAS14, LMCS15]
   - Alarm Specification Language (ASL_K) & Formal grounding on Temporal Epistemic Logic KL_1
   - Diagnosability testing for ASL_K
   - Pareto Optimal Sensor Placement [FMCAD14]
   - Synthesis of FDI components
2. Validation of Timed Failure Propagation Graphs based on Satisfiability Modulo Theory [AAAI15]
3. First model-checking approach for KL_1 over infinite state transition systems [AAMAS16]
4. Implementation [IMBSA14, TACAS16] and Application within ESA projects AUTOGEF and FAME [DASIA12, DASIA14, IMBSA14b]

Setting of this Work

Model-based techniques for FDI design for **discrete-time reactive systems**. The FDI is **compiled** to run on-board, performs **passive diagnosis** and outputs **Boolean alarms**.

Setting of this Work

Model-based techniques for FDI design for **discrete-time reactive systems**. The FDI is **compiled** to run on-board, performs **passive diagnosis** and outputs **Boolean alarms**.

Model-based:

- as opposed to **Data-Driven**, and **Rule-Based**
- **Early-phase** analysis
- **Re-use** model from functional analysis and recovery planning

Setting of this Work

Model-based techniques for FDI design for **discrete-time reactive systems**. The FDI is **compiled** to run on-board, performs **passive diagnosis** and outputs **Boolean alarms**.

Discrete-Time Reactive System:

- as opposed to **continuous time**, and **combinational** system
- Focus on high-level components interaction
- Reasonable complexity **trade-off**:
  - Combinational ⇒ too limited
  - Continuous time ⇒ too expressive

Setting of this Work

Model-based techniques for FDI design for **discrete-time reactive systems**. The FDI is **compiled** to run on-board, performs **passive diagnosis** and outputs **Boolean alarms**.

Compiled:

- as opposed to **on-the-fly**
- Easier to verify and certify
- Autonomous system with **limited on-board capabilities**
- **Consistency** in behavior (functional and timed)

Setting of this Work

Model-based techniques for FDI design for **discrete-time reactive systems**. The FDI is **compiled** to run on-board, performs **passive diagnosis** and outputs **Boolean alarms**.

Passive diagnosis

- as opposed to **active** diagnosis
- Safer to integrate with the system
- Potentially more limited

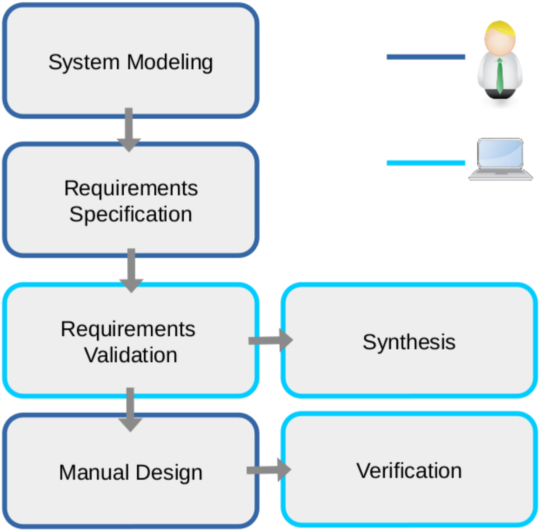
Setting of this Work

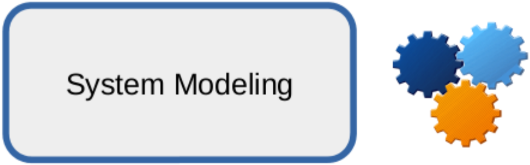
Model-based techniques for FDI design for **discrete-time reactive systems**. The FDI is **compiled** to run on-board, performs **passive diagnosis** and outputs **Boolean alarms**.

Boolean alarms:

- as opposed to **quantitative** information (e.g., probabilities), and **explanations** (e.g., diagnosis)
- Target decision making $\Rightarrow$ FR
- Simplify certification

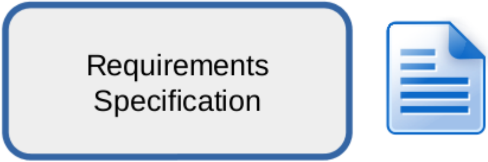
Formal Model-Based Design of FDI





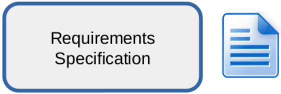
- How does the **system work**?
- What are the **faults**?
- What **sensors** are available?

$\Rightarrow$ Infinite/Finite State Discrete-Time Systems: SMV, SLIM



- What **conditions** to monitor?
- What **alarms** should the FDI provide?
- What are acceptable **delays**?
- **Recall** finite (**bounded**) or infinite (**perfect**) observations?
- Composition with the plant (Synchronous vs Asynchronous)
- Environment/Operational assumptions (Context)

Diagnosis Condition



Safety condition defining a (set of) configurations of the system.

Diagnosis Condition

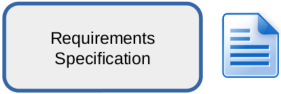
Requirements Specification

Safety condition defining a (set of) configurations of the system.

- Bad configuration of the system:

  Both engines are off

Diagnosis Condition



Safety condition defining a (set of) configurations of the system.

- Bad configuration of the system:

  Both engines are off

- Fault has occurred:

  The fuel valve is stuck-closed

Diagnosis Condition

Requirements Specification

Safety condition defining a (set of) configurations of the system.

- Bad configuration of the system:

  Both engines are off

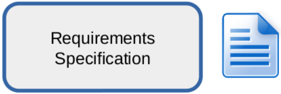
- Fault has occurred:

  The fuel valve is stuck-closed

- Conditions on the evolution of the system:

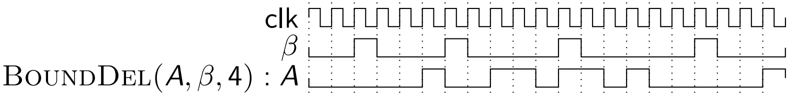
  The fuel valve has been stuck-closed for at least 3 time-units and the engines are currently on

Alarm Condition (Delays)



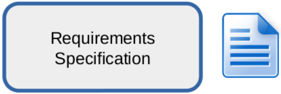
Delay between the **diagnosis condition** (β) and the **alarm** (A)

Whenever the fuel valve gets stuck-closed (β), the FDI should raise the alarm (A) within 4 time-units (BoundDel)

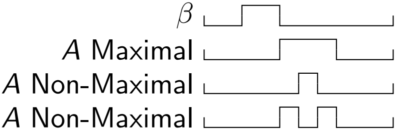


Also define ExactDel, FiniteDel

Maximality



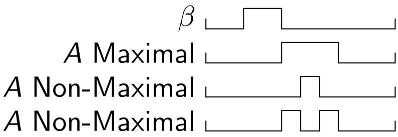
$\textsc{BoundDel}(A, \beta, 4)$



Maximality

Requirements Specification

$\textsc{BoundDel}(A, \beta, 4)$



The alarm should go up **as soon** and for **as long** as possible

$\Rightarrow$ **Deterministic** FDI $\Rightarrow$ **Equivalence Checking**

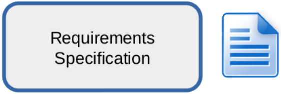
Alarm Specification Language (ASL_K)

Pattern-based language to capture requirements:

- Name of the Alarm
- Diagnosis Condition
- Delay information
- Maximality
- (Diagnosability)

$$\textsc{BoundDel}(A_{EnginesOff}, Engine_a = off \wedge Engine_b = off, 5, Max = True)$$

TFPG Analysis



What is an **acceptable** Delay?

- Alarm should fire early enough to **prevent** the **propagation** of the failure

⇒ Timed Failure Propagation Graphs

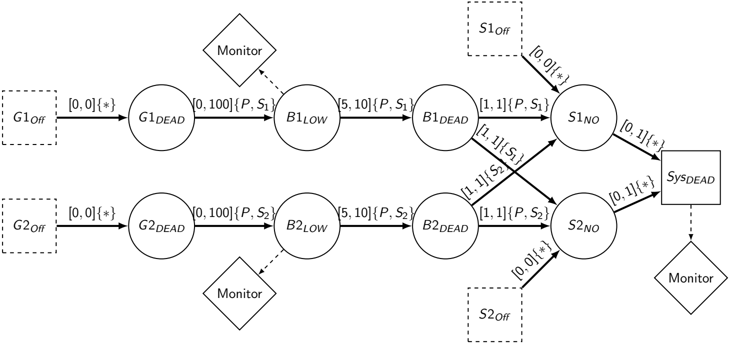
TFPG Analysis

Requirements Specification

What is an **acceptable** Delay?

- Alarm should fire early enough to **prevent** the **propagation** of the failure

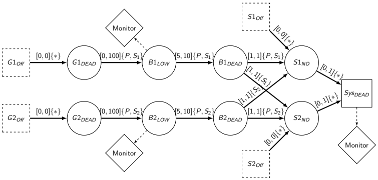
⇒ Timed Failure Propagation Graphs



TFPG Analysis

Requirements Specification

Model-Based Diagnosis: only **as good as the model**



The TFPG must be **validated**!

TFPG Validation

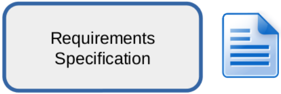
Requirements Specification

Encode all executions of the TFPG into a **Satisfiability Modulo Theory** (SMT) formula

Automated Reasoning:

- Which executions are (not) possible?
- Diagnosability, Diagnosis, Activability
- **Refinement**

TFPG Validation

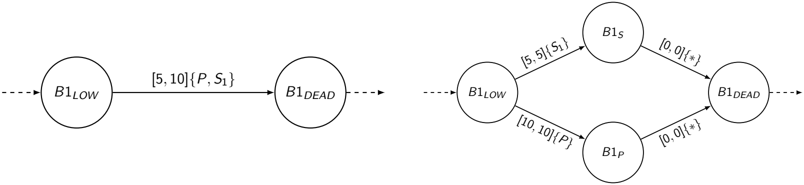


Encode all executions of the TFPG into a **Satisfiability Modulo Theory** (SMT) formula

Automated Reasoning:

- Which executions are (not) possible?
- Diagnosability, Diagnosis, Activability
- **Refinement**

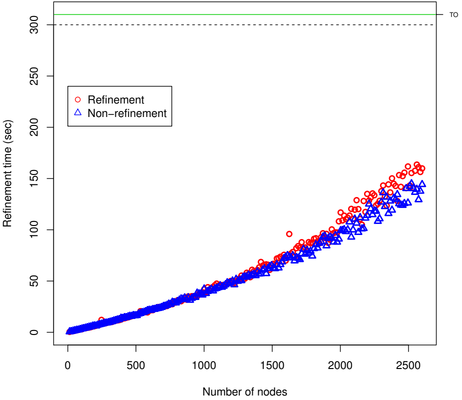
  every trace of Right can be mapped to a trace of Left

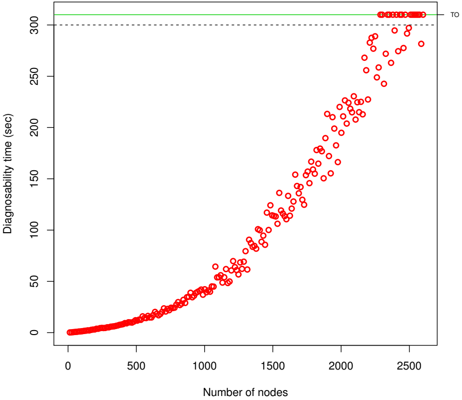


TFPG Validation

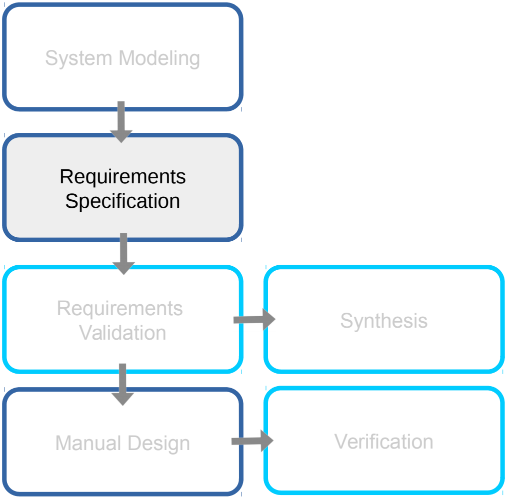
Requirements Specification

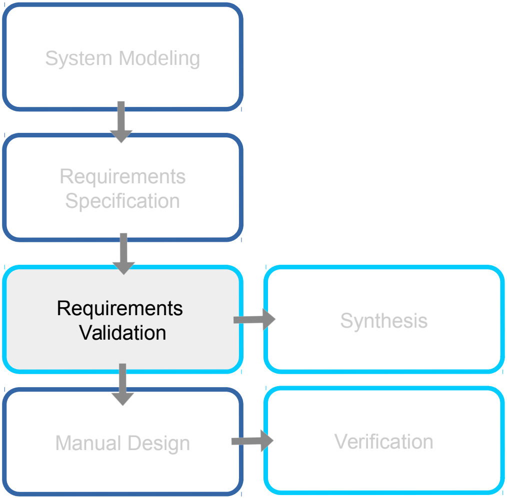
Prototype: MathSAT, Z3, pySMT





Easily handle >2000 nodes (**5x** industrial size)


System Modeling
Requirements Specification
Requirements Validation
Synthesis
Manual Design
Verification


System Modeling
Requirements Specification
Requirements Validation
Synthesis
Manual Design
Verification

Requirements Validation

- Over-constrained or Under-constrained
- Subsumption

- Over-constrained or Under-constrained
- Subsumption

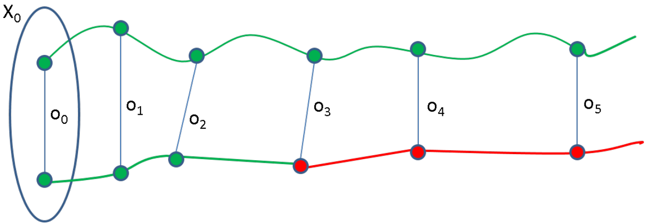
Is it **possible** to build a diagnoser for the System?

Diagnosability

Diagnosability

Requirements Validation

Observations might not be sufficient to disambiguate good and bad executions: **Critical Pair**



- No critical pair ⇒ **System diagnosable** [Sampath95]
- What if only 1 critical pair? Too **coarse**!

⇒ **Trace Diagnosability**: Diagnose **as much as possible**

Diagnosability

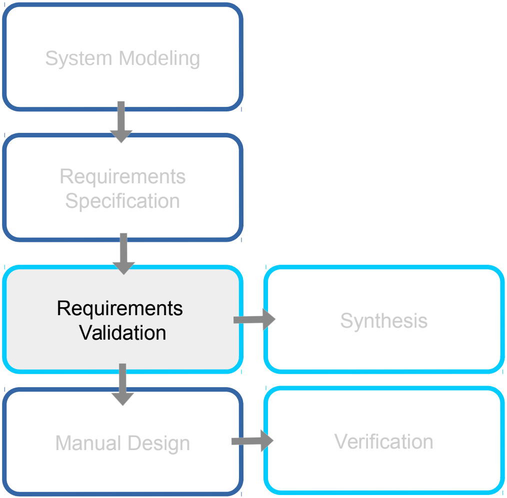
1. **Trace Diagnosability**: Diagnose **as much as possible**
   - 3-valued alarms:
     - Fault did not occur
     - Fault occurred
     - Uncertainty
2. Encode **System** and **Trace** Diagnosability as properties in Temporal Epistemic Logic
3. System Diagnosability Testing via **Twin-Plant** for ASL_K

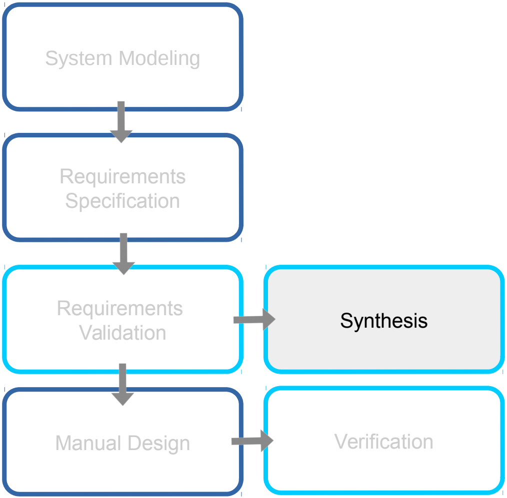
Diagnosability

Requirements Validation

1. **Trace Diagnosability**: Diagnose **as much as possible**
   - 3-valued alarms:
     - Fault did not occur
     - Fault occurred
     - Uncertainty
2. Encode **System** and **Trace** Diagnosability as properties in Temporal Epistemic Logic
3. System Diagnosability Testing via **Twin-Plant** for ASL_K
4. Optimization Problem ⇒ **Pareto Optimal Sensor Placement**

   Preserves diagnosability while reducing the sensors set to optimize a multi-cost function


System Modeling
Requirements Specification
Requirements Validation
Synthesis
Manual Design
Verification


System Modeling
Requirements Specification
Requirements Validation
Synthesis
Manual Design
Verification

Advantages:

- **Correct** by construction
- Proof of **realizability**
- Quick **prototype**

Challenges:

- Computationally **hard** (sometimes undecidable)
- Not **human readable**

State of the Art

		Transition System Type	
		Finite	Infinite
Recall	Bounded		
	Perfect	Sampath, Schumann	Timed, Stochastic

Limitations:

- Extend to ASL_K
- Avoid run-time computation

State of the Art

		Transition System Type	
		Finite	Infinite
Recall	Bounded	Parameter Synthesis	Parameter Synthesis
	Perfect	Sampath, Schumann	Timed, Stochastic

Limitations:

- Extend to ASL_K
- Avoid run-time computation

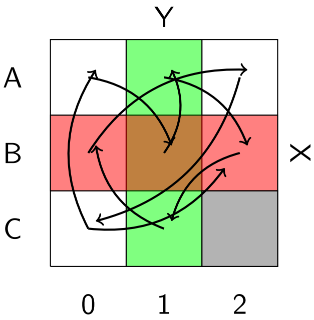
State of the Art

		Transition System Type	
		Finite	**Infinite**
Recall	**Bounded**	**Parameter Synthesis**	**Parameter Synthesis**
	Perfect	Sampath, Schumann **Belief Explorer**	Timed, Stochastic $\sim$ **(TFPG Abstraction)**

Limitations:

- Extend to ASL_K
- Avoid run-time computation

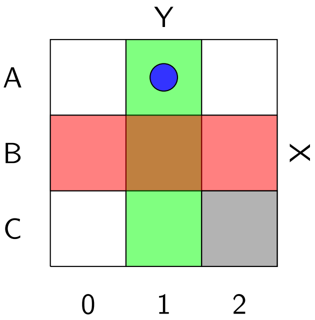
Bounded Recall



States			Observations		
t_0	t_1	t_2	t_0	t_1	t_2
A0	B1	A1	$(\overline{x},\overline{y})$	(x,y)	$(\overline{x},y)$
A1	B2	C1	$(\overline{x},y)$	$(x,\overline{y})$	$(\overline{x},y)$
A2	C0	A0	$(\overline{x},\overline{y})$	$(\overline{x},\overline{y})$	$(\overline{x},\overline{y})$
A2	C0	B2	$(\overline{x},\overline{y})$	$(\overline{x},\overline{y})$	$(x,\overline{y})$
B0	A2	C0	$(x,\overline{y})$	$(\overline{x},\overline{y})$	$(\overline{x},\overline{y})$
B1	A1	B2	(x,y)	$(\overline{x},y)$	$(x,\overline{y})$
B2	C1	B0	$(x,\overline{y})$	$(\overline{x},y)$	$(x,\overline{y})$
C0	A0	B1	$(\overline{x},\overline{y})$	$(\overline{x},\overline{y})$	(x,y)
C0	B2	C1	$(\overline{x},\overline{y})$	$(x,\overline{y})$	$(\overline{x},y)$
C1	B0	A2	$(\overline{x},y)$	$(x,\overline{y})$	$(\overline{x},\overline{y})$

$$Recall = 3$$

Bounded Recall

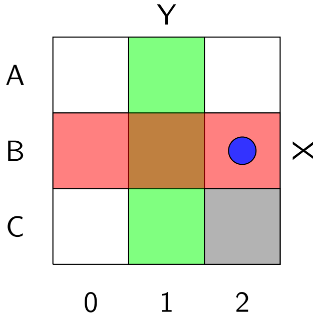


States			Observations		
t_0	t_1	t_2	t_0	t_1	t_2
A0	B1	A1			
A1	B2	C1	$(\overline{x}, y)$		
A2	C0	A0			
A2	C0	B2			
B0	A2	C0			
B1	A1	B2			
B2	C1	B0			
C0	A0	B1			
C0	B2	C1			
C1	B0	A2	$(\overline{x}, y)$		

$$Recall = 3$$

$(\overline{x}, y)$

Bounded Recall

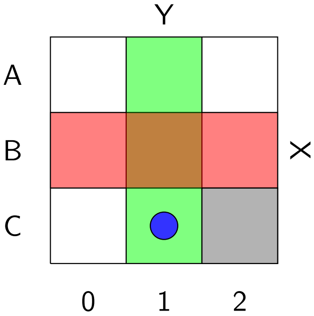


States			Observations		
t_0	t_1	t_2	t_0	t_1	t_2
A0	B1	A1			
A1	B2	C1	$(\overline{x}, y)$	$(x, \overline{y})$	
A2	C0	A0			
A2	C0	B2			
B0	A2	C0			
B1	A1	B2			
B2	C1	B0			
C0	A0	B1			
C0	B2	C1			
C1	B0	A2	$(\overline{x}, y)$	$(x, \overline{y})$	

$$Recall = 3$$

$(\overline{x}, y)(x, \overline{y})$

Bounded Recall

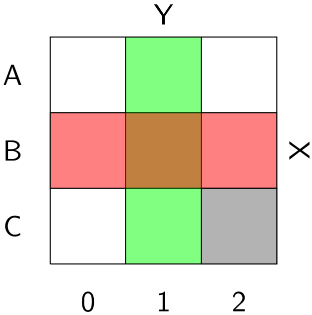


States			Observations		
t_0	t_1	t_2	t_0	t_1	t_2
A0	B1	A1			
A1	B2	C1	$(\overline{x}, y)$	$(x, \overline{y})$	$(\overline{x}, y)$
A2	C0	A0			
A2	C0	B2			
B0	A2	C0			
B1	A1	B2			
B2	C1	B0			
C0	A0	B1			
C0	B2	C1			
C1	B0	A2	$(\overline{x}, y)$	$(x, \overline{y})$	

$$Recall = 3$$

$$(\overline{x}, y)(x, \overline{y})(\overline{x}, y)$$

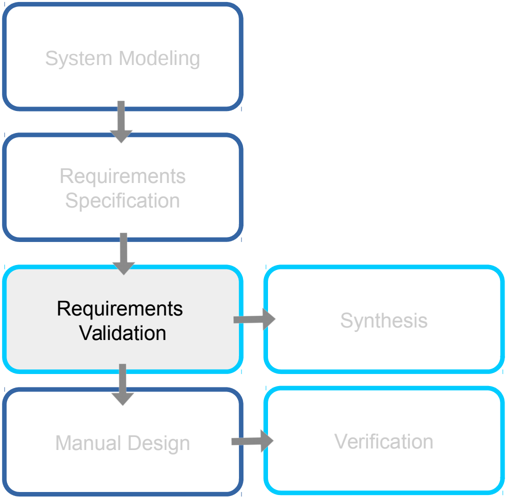
Bounded Recall

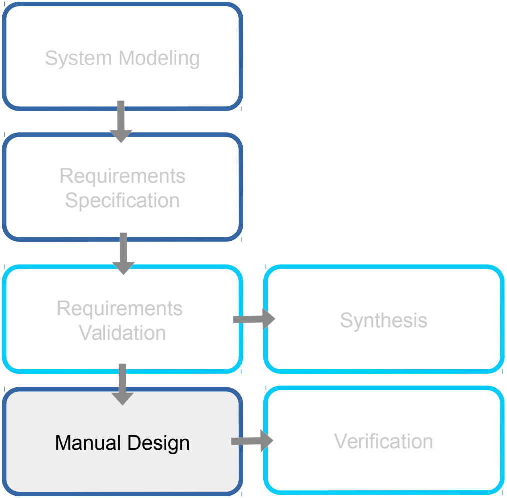


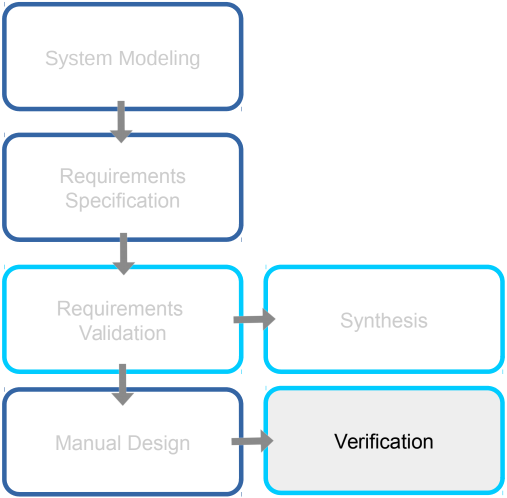
States			Observations		
t_0	t_1	t_2	t_0	t_1	t_2
A0	B1	A1			
A1	B2	C1	$(\overline{x}, y)$	$(x, \overline{y})$	$(\overline{x}, y)$
A2	C0	A0			
A2	C0	B2			
B0	A2	C0			
B1	A1	B2			
B2	C1	B0			
C0	A0	B1			
C0	B2	C1			
C1	B0	A2	$(\overline{x}, y)$	$(x, \overline{y})$	~~$(\overline{x}, \overline{y})$~~

$$Recall = 3$$

$$(\overline{x}, y)(x, \overline{y})(\overline{x}, y)$$


System Modeling
Requirements Specification
Requirements Validation
Synthesis
Manual Design
Verification


System Modeling
Requirements Specification
Requirements Validation
Synthesis
Manual Design
Verification


System Modeling
Requirements Specification
Requirements Validation
Synthesis
Manual Design
Verification

- Formal Model + Formal Requirements ⇒ **Model-Checking**
- Requirements translated into KL_1

⇒ Develop **effective** model-checking algorithms for KL_1 over **infinite state** transition systems

KL_1

- LTL + K_A Operator:

$$G(K_A Sunny)$$

  It is always the case that Agent A knows that it is Sunny

- K_A Operator: two points in a trace cannot be distinguished if they have the same observations (up to recall) for A
- KL_1: Disallow nesting of K (i.e., ~~$K_A K_B \varphi$~~)

KL_1 Example

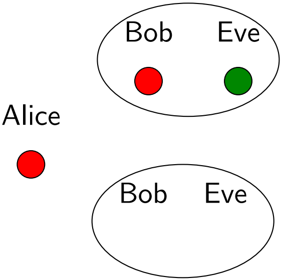
- 3 cards: 2 **Red**, 1 **Green**
- $(Alice = Red \wedge Bob = Red) \rightarrow K_{Alice}(Bob = Red)$?

Bob Eve

Alice

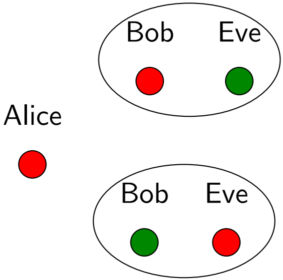
KL_1 Example

- 3 cards: 2 **Red**, 1 **Green**
- $(Alice = Red \wedge Bob = Red) \rightarrow K_{Alice}(Bob = Red)$?



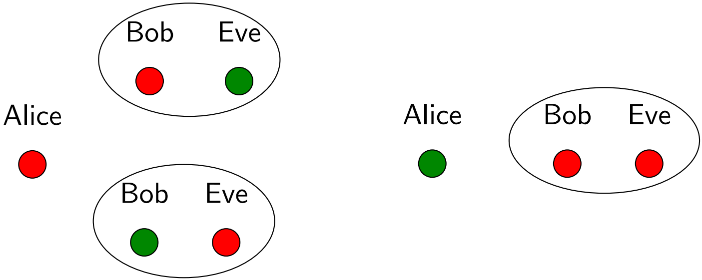
KL_1 Example

- 3 cards: 2 **Red**, 1 **Green**
- $(Alice = Red \wedge Bob = Red) \rightarrow K_{Alice}(Bob = Red)$? No!



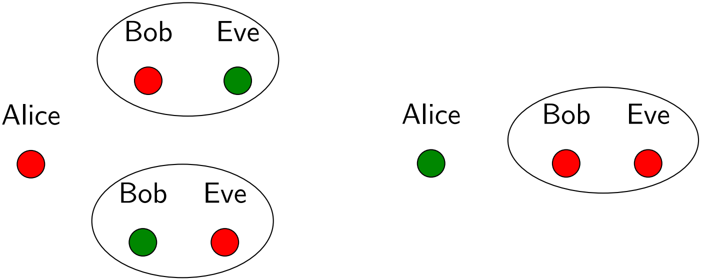
KL_1 Example

- 3 cards: 2 **Red**, 1 **Green**
- $(Alice = Red \wedge Bob = Red) \rightarrow K_{Alice}(Bob = Red)$? No!
- $(Alice = Green) \rightarrow K_{Alice}(Bob = Red)$?



KL_1 Example

- 3 cards: 2 **Red**, 1 **Green**
- $(Alice = Red \wedge Bob = Red) \rightarrow K_{Alice}(Bob = Red)$? No!
- $(Alice = Green) \rightarrow K_{Alice}(Bob = Red)$? Yes!



$\mathsf{ASL}_K \Rightarrow KL_1$

$$\textsc{BoundDel}_K(A_{EnginesOff}, Engine_a = off \wedge Engine_b = off, 5$$

$$Max = True, System)$$

$$\Downarrow$$

$$G(A_{EngineOff} \rightarrow O^{\leq 5}(Engine_a = off \wedge Engine_b = off)) \wedge$$

$$G(Engine_a = off \wedge Engine_b = off \rightarrow F^{\leq 5}A_{EngineOff}) \wedge$$

$$G(KO^{\leq 5}(Engine_a = off \wedge Engine_b = off) \rightarrow A_{EngineOff})$$

Unified Encoding $\Rightarrow$ **recall** and **synchronicity** embedded in K_A

$\mathsf{ASL}_K \Rightarrow KL_1$

Verification + FDI ?

	Template	$Maximality = False$	$Maximality = True$
$diag = System$	ExactDel	$G(A \to Y^n\beta) \wedge G(\beta \to X^nA)$	$G(A \to Y^n\beta) \wedge G(\beta \to X^nA) \wedge G(KY^n\beta \to A)$
	BoundDel	$G(A \to O^{\leq n}\beta) \wedge G(\beta \to F^{\leq n}A)$	$G(A \to O^{\leq n}\beta) \wedge G(\beta \to F^{\leq n}A) \wedge G(KO^{\leq n}\beta \to A)$
	FiniteDel	$G(A \to O\beta) \wedge G(\beta \to FA)$	$G(A \to O\beta) \wedge G(\beta \to FA) \wedge G(KO\beta \to A)$
$diag = Trace$	ExactDel	$G(A \to Y^n\beta) \wedge G((\beta \to X^nKY^n\beta) \to (\beta \to X^nA))$	$G(A \to Y^n\beta) \wedge G((\beta \to X^nKY^n\beta) \to (\beta \to X^nA)) \wedge G(KY^n\beta \to A)$
	BoundDel	$G(A \to O^{\leq n}\beta) \wedge G((\beta \to F^{\leq n}KO^{\leq n}\beta) \to (\beta \to F^{\leq n}A))$	$G(A \to O^{\leq n}\beta) \wedge G((\beta \to F^{\leq n}KO^{\leq n}\beta) \to (\beta \to F^{\leq n}A)) \wedge G(KO^{\leq n}\beta \to A)$
	FiniteDel	$G(A \to O\beta) \wedge G((\beta \to FKO\beta) \to (\beta \to FA))$	$G(A \to O\beta) \wedge G((\beta \to FKO\beta) \to (\beta \to FA)) \wedge G(KO\beta \to A)$

Correctness *Completeness* *Diagnosability* *Maximality*

Model-Checking KL_1

		Transition System Type	
		Finite	**Infinite**
Recall	**Bounded**	MCMAS, MCK	
	Perfect	MCK	

Model-Checking KL_1

		Transition System Type	
		Finite	**Infinite**
Recall	**Bounded**	MCMAS, MCK, **Lazy**	**Lazy**
	Perfect	MCK	

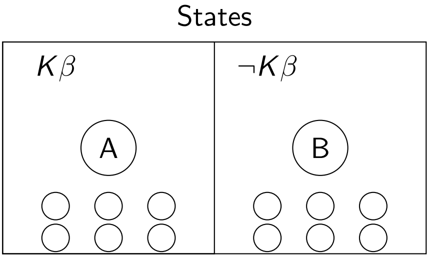
Lazy KL_1 Model-Checking

- **Bounded Recall**: Semantics depends only on the state
- Reduce to LTL + Set states satisfying $K\beta$

Lazy KL_1 Model-Checking

Verification + FDI ? ≡

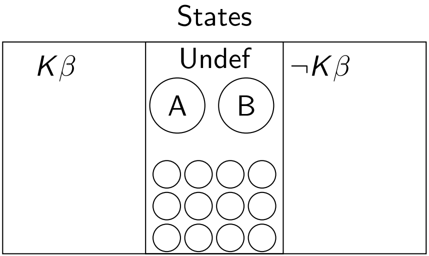
- **Bounded Recall**: Semantics depends only on the state
- Reduce to LTL + Set states satisfying $K\beta$



$M \models G(K\beta)$? Cex: A,B

Lazy KL_1 Model-Checking

- **Bounded Recall**: Semantics depends only on the state
- Reduce to LTL + Set states satisfying $K\beta$
- **Lazy**: **On-demand** computation of states in $K\beta$

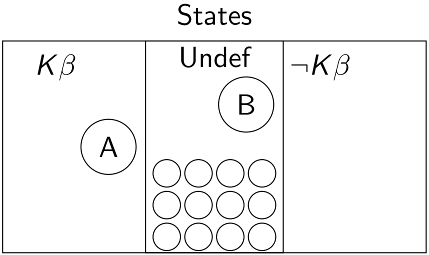


$M \models G(K\beta)$? Cex: A

Optimizations: Static Learning, Generalization, Dual-Rail Encoding, $InvKL_1$

Lazy KL_1 Model-Checking

- **Bounded Recall**: Semantics depends only on the state
- Reduce to LTL + Set states satisfying $K\beta$
- **Lazy**: **On-demand** computation of states in $K\beta$

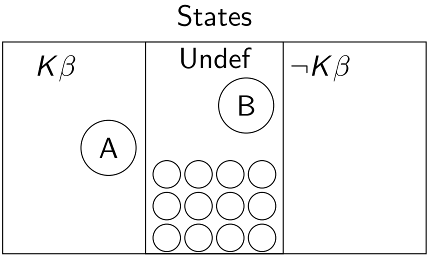


$M \models G(K\beta)$? Cex: A

Optimizations: Static Learning, Generalization, Dual-Rail Encoding, $InvKL_1$

Lazy KL_1 Model-Checking

- **Bounded Recall**: Semantics depends only on the state
- Reduce to LTL + Set states satisfying $K\beta$
- **Lazy**: **On-demand** computation of states in $K\beta$

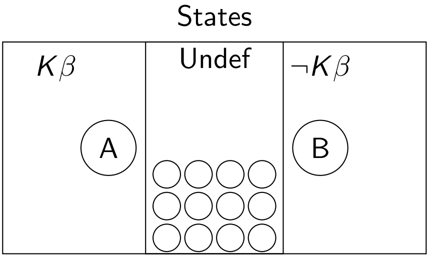


$M \models G(K\beta)$? Cex: A,B

Optimizations: Static Learning, Generalization, Dual-Rail Encoding, $InvKL_1$

Lazy KL_1 Model-Checking

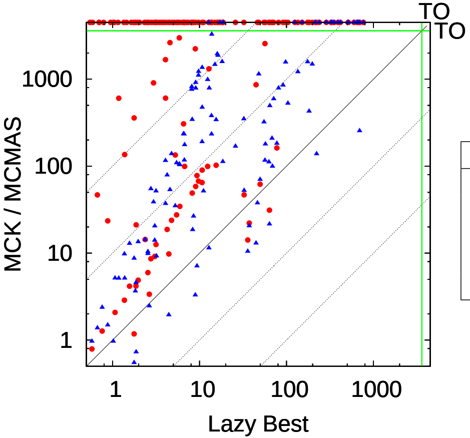
- **Bounded Recall**: Semantics depends only on the state
- Reduce to LTL + Set states satisfying $K\beta$
- **Lazy**: **On-demand** computation of states in $K\beta$



$M \models G(K\beta)$? Cex: A,B

Optimizations: Static Learning, Generalization, Dual-Rail Encoding, $InvKL_1$

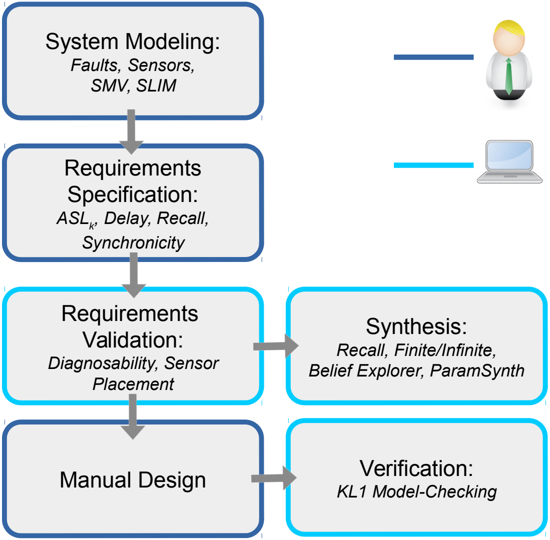
Experimental Results



Recall	#Obs	Lazy Basic	Lazy Best
0	11	3.28	1.46
5	66	3536.17	52.72
10	121	TO	103.47
20	231	TO	288.31
40	451	TO	981.11

- Improvement w.r.t. existing tools: **MCMAS** – **MCK**
- **Change in core technology** (BDD vs SAT)

Formal Model-Based Design of FDI



AUTOGEF and FAME

AUTOGEF

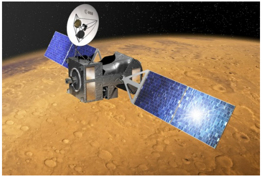
- Automated Generation of FDIR
- Specification & Synthesis
- Finite Discrete Time

FAME

- FDIR Development Methodology and V&V
- Process & Timed Propagation
- Infinite Continuous Time

Include **Fault Recovery** and coordination

Evaluation



Exomars TGO case-study:

- Non-experts in formal verification
- 6-10 Alarms ⇒ 700-2413 States
- Positive Evaluation:

  Reach a **better understanding** of System and FDIR designs

Conclusions

1. Formal FDI design:
   - **Unified specification** (ASL_K) that accounts for multiple issues such as synchronicity, recall, delays etc.
   - **Synthesis** of FDI components
   - **Diagnosability** testing for ASL_K, Algorithm for Pareto Optimal **Sensor Placement**
2. Validation of **Timed Failure Propagation Graphs** based on Satisfiability Modulo Theory
3. **Model-checking** KL_1 over infinite/finite state systems
4. ESA projects **AUTOGEF** and **FAME**

FDI Design and **Temporal Epistemic Logic**

Future Work

- **Case-studies**: End-to-end evaluation within industrial setting
- **Distributed FDI**: Architectural and Contract-Based Design
- **Model-Checking**: KL_n and improve performances

Thank You! Questions?

A Formal Foundation of FDI Design
via
Temporal Epistemic Logic

- FDI Specification and & ASL_K
- FDI Verification, Validation and Synthesis
- KL_1 Model-Checking
- TFPG Validation, Pareto Optimal Sensor Placement
- AUTOGEF, FAME

ASL_K

Pareto

KL_1

Synthesis

Industrial

Timed Failure Propagation Graphs

Conclusion

Summary View

Recall	Task	Plant	Logic / Problem	Tool / Algorithm
BR	Diagnosability, Verification	Infinite	KL_1	Lazy
	Synthesis	Infinite	Parameter Synthesis	nuXmv
PR	Diagnosability, Verification	Finite	KL_1	MCK
		Infinite	LTL	nuXmv
		Infinite	KL_1	OPEN
	Synthesis	Finite	Belief Explorer	xSAP
		Infinite	OPEN	OPEN/Abstraction

Diagnosis Condition

Safety condition defining a (set of) configurations of the system.

Diagnosis Condition

Safety condition defining a (set of) configurations of the system.

- Bad configuration of the system:

  Both engines are off

Diagnosis Condition

Safety condition defining a (set of) configurations of the system.

- Bad configuration of the system:

  Both engines are off

- Fault has occurred:

  The fuel valve is stuck-closed

Diagnosis Condition

Safety condition defining a (set of) configurations of the system.

- Bad configuration of the system:

  Both engines are off

- Fault has occurred:

  The fuel valve is stuck-closed

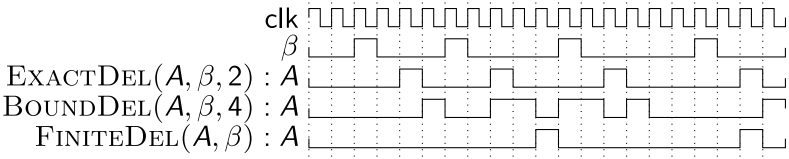
- Conditions on the evolution of the system:

  The fuel valve has been stuck-closed for at least 3 time-units and the engines are currently on

Alarm Condition (Delays)

Delay between the diagnosis condition β and the alarm A

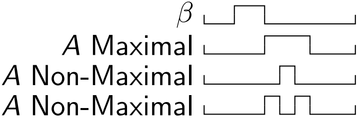
Whenever the fuel valve gets stuck-closed, the FDI should raise the alarm within 4 time-units (BoundDel)



Maximality

- The alarm should go up as soon and for as long as possible

$$\textsc{BoundDel}(A, \beta, 4)$$



Maximality removes ambiguity

Diagnosability

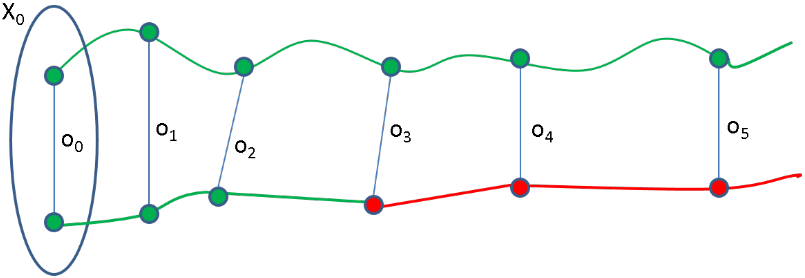
Always possible to satisfy an Alarm Condition?

Diagnosability

Always possible to satisfy an Alarm Condition?

No! Observations might not be sufficient to disambiguate:

Critical Pair



Diagnosability

- No critical pair = System diagnosable [Sampath]
- Too coarse-grained? E.g., 1 critical pair?

$\Rightarrow$ Trace Diagnosability: Diagnose as much as possible

Example of an ASL requirement

- Detect when both engines are off.
- Delay of at most 5 time-units.
- Require maximality and system diagnosability

Example of an ASL requirement

- Detect when both engines are off.
- Delay of at most 5 time-units.
- Require maximality and system diagnosability

$$\textsc{BoundDel}(A_{EnginesOff}, Engine_a = off \wedge Engine_b = off, 5$$
$$Max = True, System)$$

From ASL to Logic

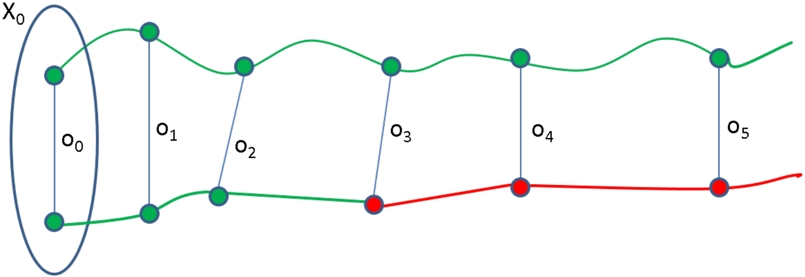
Linear **T**emporal **L**ogic

- Describes properties of a single trace.

From ASL to Logic

Linear **T**emporal **L**ogic

- Describes properties of a single trace.
- Cannot encode Diagnosability (neither Maximality)



From ASL to Logic

Linear **T**emporal **L**ogic

- Describes properties of a single trace.
- Cannot encode Diagnosability (neither Maximality)

What could an ideal diagnoser **know** with the available sensors?

Reasoning about Knowledge: **Epistemic** Logic

From ASL to Logic

ASL	$maximality = False$	$maximality = True$
$diag = System$	Single Trace (LTL)	
$diag = Trace$		

From ASL to Logic

ASL	*maximality* = *False*	*maximality* = *True*
diag = *System*	Single Trace (LTL)	Set of Traces (LTL+K)
diag = *Trace*	Set of Traces (LTL+K)	Set of Traces (LTL+K)

LTL+K: LTL + **Epistemic** operator K

Alarm Specification Language (ASL)

- Alarm Variable A
- Diagnosis Condition β (Past-only LTL)
- Delay n (Bounded LTL operators)

$\Rightarrow$ Formalization of:

Correctness: Alarm occurrence implies occurrence of the fault in the past.

Completeness: Fault occurrence implies Alarm occurrence in the future.

$$\textsc{BoundDel}(A, \beta, n) : G(A \rightarrow O^{\leq n}\beta) \wedge G(\beta \rightarrow F^{\leq n}A)$$

Temporal Epistemic Logic

$K\phi$ holds at time n in a trace σ_1 iff ϕ holds at time n in all traces that are observational equivalent to σ_1.

$$\sigma_1, n \models K\phi \text{ iff } \forall \sigma_2.\ obs(\sigma_1^n) = obs(\sigma_2^n) \Rightarrow\ \sigma_2, n \models \phi.$$

Diagnosability as Epistemic Property

Could the ideal diagnoser detect the diagnosis condition?

Diagnosability : Whenever the diagnosis condition β occurs the ideal diagnoser will (eventually) know that it occurred.

$\textsc{BoundDel}(A, \beta, n)$: $G(\beta \rightarrow F^{\leq n} K O^{\leq n} \beta)$

Epistemic Encoding provides a unified way of dealing with the problem.

Maximality as Epistemic Property

The alarm should go up as soon and for as long as possible

Maximality: As long as the ideal diagnoser can be certain about the occurrence of β the alarm A will be active.

$\textsc{BoundDel}(A, \beta, n)$: $G(KO^{\leq n}\beta \rightarrow A)$

ASL_K (Overview)

	Template	$maximality = False$	$maximality = True$
$diag = System$	ExactDel	$G(A \rightarrow Y^n\beta) \wedge G(\beta \rightarrow X^n A)$	$G(A \rightarrow Y^n\beta) \wedge G(\beta \rightarrow X^n A) \wedge G(KY^n\beta \rightarrow A)$
	BoundDel	$G(A \rightarrow O^{\leq n}\beta) \wedge G(\beta \rightarrow F^{\leq n} A)$	$G(A \rightarrow O^{\leq n}\beta) \wedge G(\beta \rightarrow F^{\leq n} A) \wedge G(KO^{\leq n}\beta \rightarrow A)$
	FiniteDel	$G(A \rightarrow O\beta) \wedge G(\beta \rightarrow FA)$	$G(A \rightarrow O\beta) \wedge G(\beta \rightarrow FA) \wedge G(KO\beta \rightarrow A)$
$diag = Trace$	ExactDel	$G(A \rightarrow Y^n\beta) \wedge G((\beta \rightarrow X^n KY^n\beta) \rightarrow (\beta \rightarrow X^n A))$	$G(A \rightarrow Y^n\beta) \wedge G((\beta \rightarrow X^n KY^n\beta) \rightarrow (\beta \rightarrow X^n A)) \wedge G(KY^n\beta \rightarrow A)$
	BoundDel	$G(A \rightarrow O^{\leq n}\beta) \wedge G((\beta \rightarrow F^{\leq n} KO^{\leq n}\beta) \rightarrow (\beta \rightarrow F^{\leq n} A))$	$G(A \rightarrow O^{\leq n}\beta) \wedge G((\beta \rightarrow F^{\leq n} KO^{\leq n}\beta) \rightarrow (\beta \rightarrow F^{\leq n} A)) \wedge G(KO^{\leq n}\beta \rightarrow A)$
	FiniteDel	$G(A \rightarrow O\beta) \wedge G((\beta \rightarrow FKO\beta) \rightarrow (\beta \rightarrow FA))$	$G(A \rightarrow O\beta) \wedge G((\beta \rightarrow FKO\beta) \rightarrow (\beta \rightarrow FA)) \wedge G(KO\beta \rightarrow A)$

Correctness *Completeness* *Diagnosability* *Maximality*

Example

$$\textsc{BoundDel}_K(A_{EnginesOff}, Engine_a = off \land Engine_b = off, 5$$

$$Max = True, System)$$

$$\Downarrow$$

$$G(A_{EngineOff} \rightarrow O^{\leq 5}(Engine_a = off \land Engine_b = off)) \land$$

$$G(Engine_a = off \land Engine_b = off \rightarrow F^{\leq 5} A_{EngineOff}) \land$$

$$G(KO^{\leq 5}(Engine_a = off \land Engine_b = off) \rightarrow A_{EngineOff})$$

Related

- Jiang and Kumar[3]: Specification as LTL
- Ezekiel et al. [1] Huang [2]: Diagnosability as epistemic

Our framework goes in the same directions: unifying view of other aspects of the design process (e.g., validation and synthesis), and considering key problems such as delays, maximality and trace-diagnosability.
Synthesis procedure similar to Schumann's[5], but we capture more expressive diagnosis conditions, and introduce delays.

Theorem

Let α be a propositional formula, α is d-delay diagnosable (ala Sampath) in P iff $\textsc{BoundDel}(A, O\alpha, d)$ *is diagnosable in P.*

Sensor Placement

Requirements Validation

Can we **minimize** the number of sensors, while preserving diagnosability? **Sensor Placement**

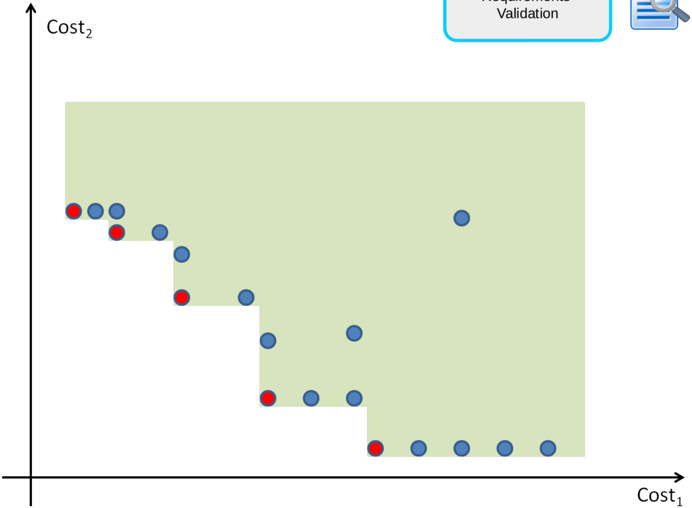
- Subset of sensors that preserves diagnosability
- Sensors have a cost (energy, weight, monetary)

  E.g., Trade-off between cost and delay

$\Rightarrow$ **Pareto Optimality**

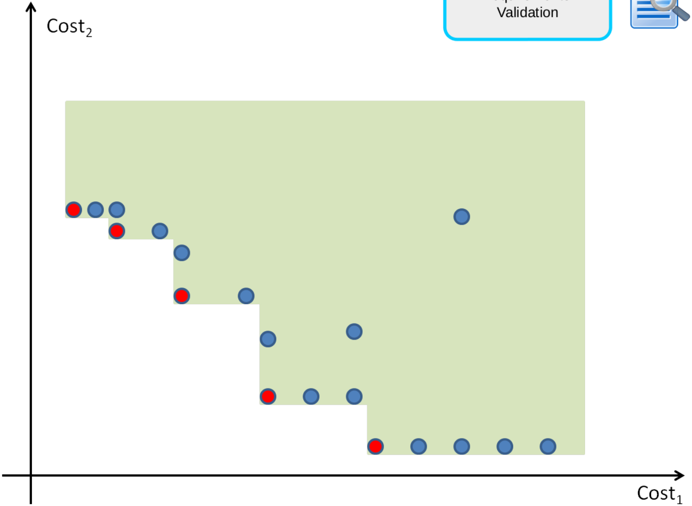
Pareto Optimal Sensor Placement

Requirements Validation



Pareto Optimal Sensor Placement

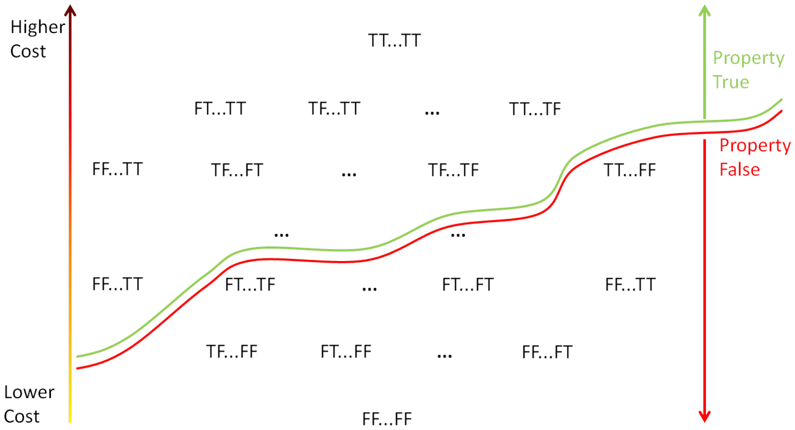
Requirements Validation



Simple version: 43/81 – **Optimized** version: **81/81**

>170 bits and **40** obs.

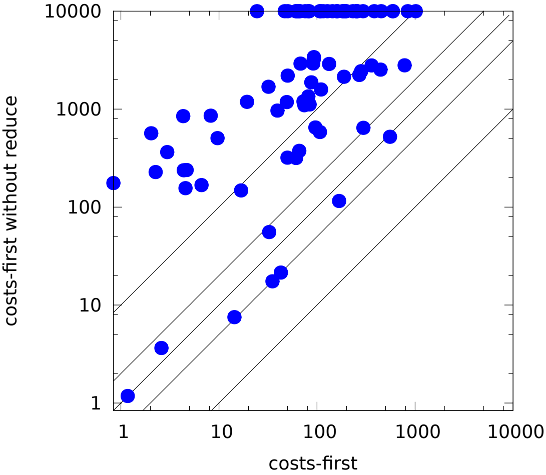
Property-Monotonicity and Cost-Monotonicity



Experiments: solved instances

Family	#Instances	valuations-first	one-cost slicing	costs-first
c432	32	11	13	**32**
cassini	21	6	12	**21**
elevator	4	4	4	4
orbiter	4	4	4	4
roversmall	4	4	4	4
roverbig	4	4	4	4
x34	4	4	4	4
product lines	8	6	4	**8**
TOTAL	81	43	49	**81**

Experiments: Impact of REDUCE in costs-first



Background

- Transition system $S \doteq (X, X_0, I, T)$, set of states X, initial states X_0, inputs I and the transition relation $T \subseteq X \times I \times X$
- Trace $\sigma \doteq x_0, i_1, x_1, \cdots,\ x_0 \in X_0$, $\forall j.(x_j, i_{j+1}, x_{j+1}) \in T$
- Observation $obs(x_j) \doteq o_j \in O$ (similarly for i_j)
- Observable Trace $obs(\sigma) \doteq obs(x_0), obs(i_1), obs(x_1), \ldots$
- State x defines unobservable condition of the system:

  The fuel valve is closed
- Observation $obs(x)$ defines an observable situation:

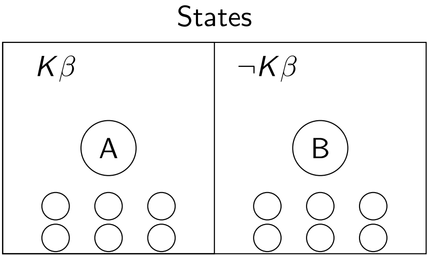
  No fuel is coming out of the pipe

Lazy KL_1 Model-Checking

- **Bounded Recall**: Semantics depends only on the state
- Reduce to LTL + Set states satisfying $K\beta$

Lazy KL_1 Model-Checking

- **Bounded Recall**: Semantics depends only on the state
- Reduce to LTL + Set states satisfying $K\beta$

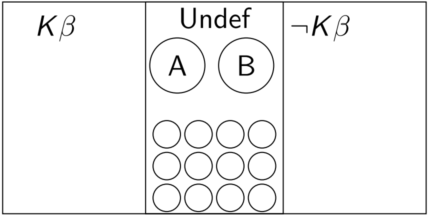


$M \models G(K\beta)$? Cex: A,B

Lazy KL_1 Model-Checking

- **Bounded Recall**: Semantics depends only on the state
- Reduce to LTL + Set states satisfying $K\beta$
- **Lazy**: **On-demand** computation of states in $K\beta$

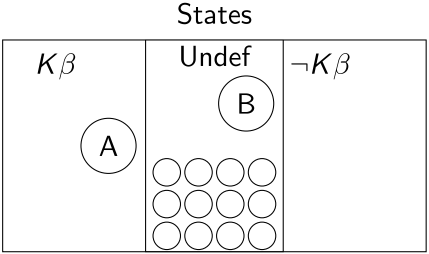
States



$M \models G(K\beta)$? Cex: A,B

Lazy KL_1 Model-Checking

- **Bounded Recall**: Semantics depends only on the state
- Reduce to LTL + Set states satisfying $K\beta$
- **Lazy**: **On-demand** computation of states in $K\beta$

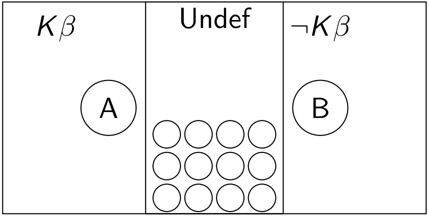


$M \models G(K\beta)$? Cex: A,B

Lazy KL_1 Model-Checking

- **Bounded Recall**: Semantics depends only on the state
- Reduce to LTL + Set states satisfying $K\beta$
- **Lazy**: **On-demand** computation of states in $K\beta$

States



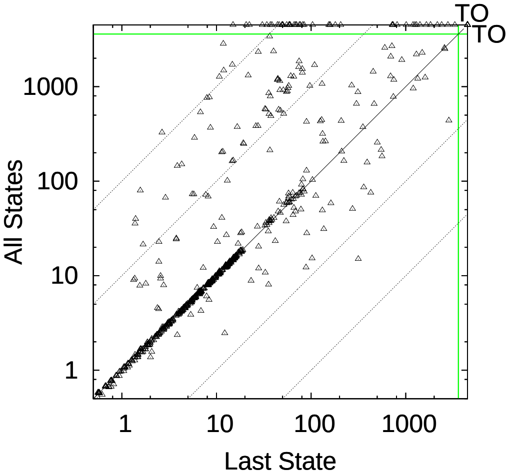
$M \models G(K\beta)$? Cex: A,B

Experimental Results

Optimizations are **crucial**:

- Static Learning,
- Generalization,
- Dual-Rail Encoding,
- $InvKL_1$

Recall	#Obs	Lazy Basic	Lazy Best
0	11	3.28	1.46
5	66	3536.17	52.72
10	121	TO	103.47
20	231	TO	288.31
40	451	TO	981.11



#DC	MCMAS	Lazy
40	3.66	1.83
80	26.57	8.54
120	169.43	25.9
160	322.45	55.2
200	528.42	104.02
240	1582.68	174.86
280	TO	287.06

Lazy algorithm

function VERIFY(M, φ)
    ϕ_ρ, placeholders := BOOL_ABSTRACTION(φ)
    M_ρ := EXTEND(M, placeholders)
    loop
        cex := $M_\rho \models \phi_\rho$
        if not cex **then**
            return “Satisfied”
        end if
        if IS_SPURIOUS(M, cex, placeholders) **then**
            ϕ_ρ := LEARN_LEMMA(M, cex, placeholders, ϕ_ρ)
        else
            return cex
        end if
    end loop
end function

Lazy algorithm

```
function IS_SPURIOUS(M, cex, placeholders)
    for state ∈ cex do
        for ρ_{K_A β} ∈ placeholders do
            p_value := ρ_{K_A β}(state)
            if not ((state ∈ ⟦K_A β⟧) ↔ p_value) then
                return True // Spurious!
            end if
        end for
    end for
    return False
end function
```

Background

LTL with Past:

- X in the next state, Y in the previous state.
- $X^n\varphi \doteq XX^{n-1}\varphi$ ($X^0\varphi = \varphi$)
- F Finally, O Once.
- $F^{\leq n}\varphi \doteq \varphi \vee X\varphi \vee \cdots \vee X^n\varphi$
- Similar definitions for Y^n, $O^{\leq n}$

$$\sigma_1, n \models K\phi \text{ iff } \forall\sigma_2.\ obs(\sigma_1^n) = obs(\sigma_2^n) \Rightarrow\ \sigma_2, n \models \phi.$$

Temporal Epistemic Logic

- LTL + K

$K\phi$ holds at time n in a trace σ_1 iff ϕ holds at time n in all traces that are observational equivalent to σ_1.

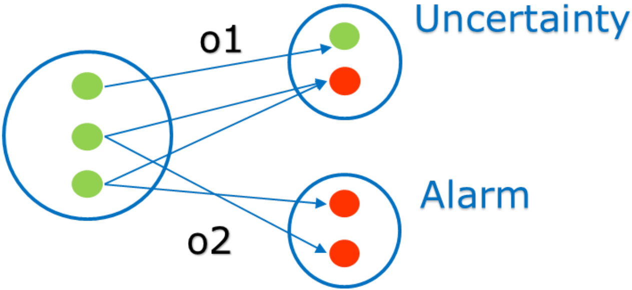
$$\sigma_1, n \models K\phi \text{ iff } \forall \sigma_2.\ obs(\sigma_1^n) = obs(\sigma_2^n) \Rightarrow\ \sigma_2, n \models \phi.$$

$K\phi$ as ideal diagnoser "Knows" that ϕ

Synthesis

Given a System and a Specification, we build a diagnoser that is:

- Correct,
- Trace Complete,
- Maximal



Synthesis as Param Synthesis

States			Observations		
t_0	t_1	t_2	t_0	t_1	t_2
A0	B1	A1	$(\overline{x}, \overline{y})$	(x, y)	$(\overline{x}, y)$
A1	B2	C1	$(\overline{x}, y)$	$(x, \overline{y})$	$(\overline{x}, y)$
A2	C0	A0	$(\overline{x}, \overline{y})$	$(\overline{x}, \overline{y})$	$(\overline{x}, \overline{y})$
A2	C0	B2	$(\overline{x}, \overline{y})$	$(\overline{x}, \overline{y})$	$(x, \overline{y})$
B0	A2	C0	$(x, \overline{y})$	$(\overline{x}, \overline{y})$	$(\overline{x}, \overline{y})$
B1	A1	B2	(x, y)	$(\overline{x}, y)$	$(x, \overline{y})$
B2	C1	B0	$(x, \overline{y})$	$(\overline{x}, y)$	$(x, \overline{y})$
C0	A0	B1	$(\overline{x}, \overline{y})$	$(\overline{x}, \overline{y})$	(x, y)
C0	B2	C1	$(\overline{x}, \overline{y})$	$(x, \overline{y})$	$(\overline{x}, y)$
C1	B0	A2	$(\overline{x}, y)$	$(x, \overline{y})$	$(\overline{x}, \overline{y})$

Figure: Traces and Observations for Recall 2

FDI Synthesis

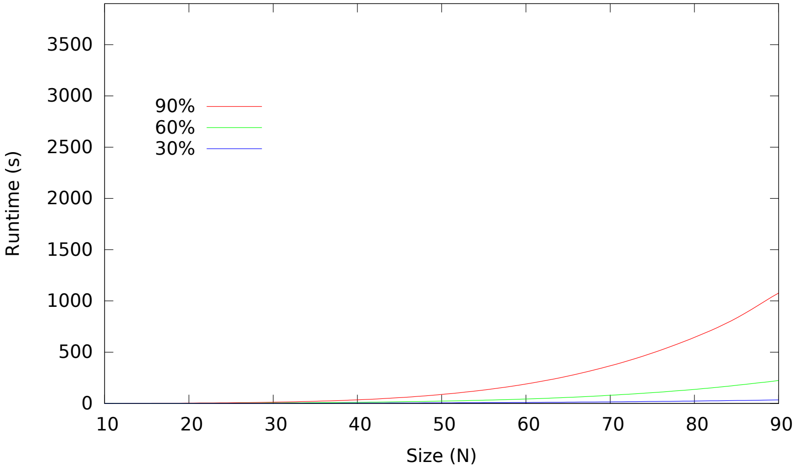


Figure: BR MB Free

FDI Synthesis

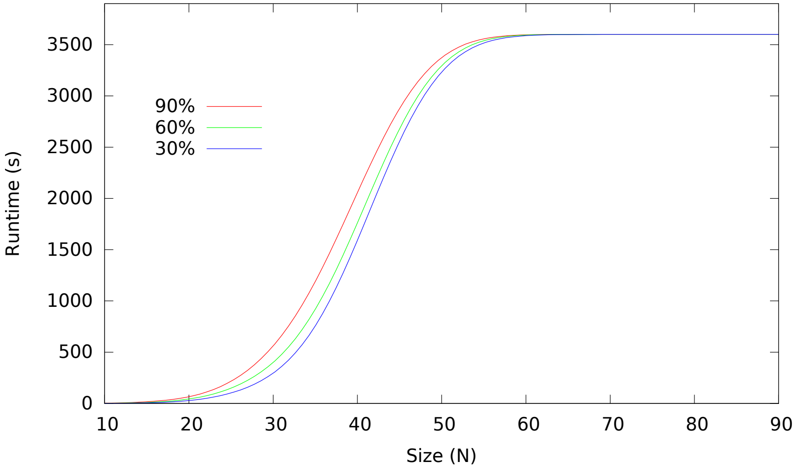


Figure: BR MB InitX

FDI Synthesis

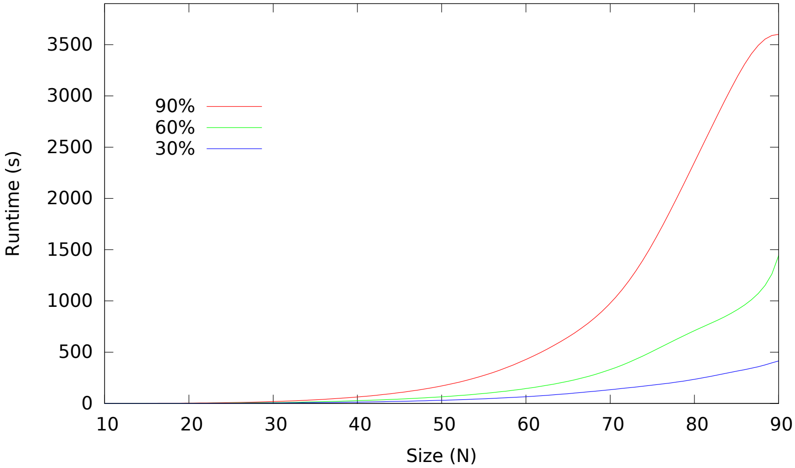


Figure: PR MB Free

FDI Synthesis

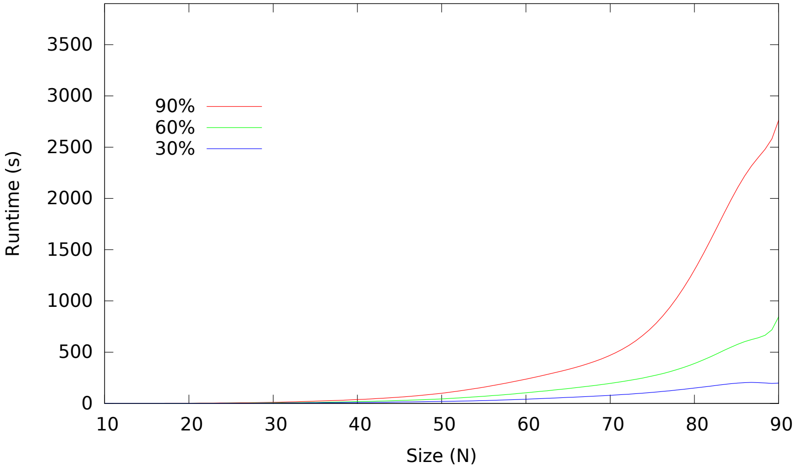


Figure: PR MB InitX

FDI Synthesis

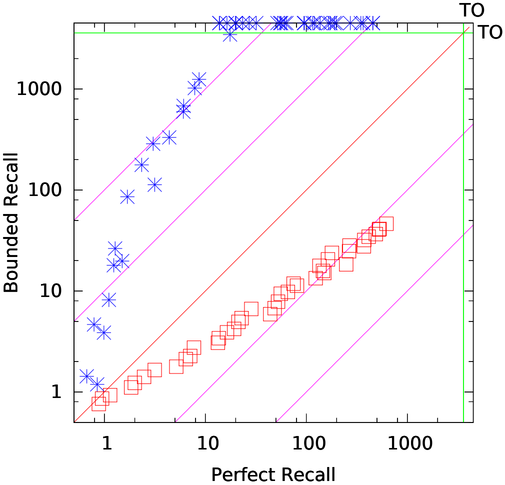


Figure: BS vs PR (Blue: InitX – Red: Free)

FDI Synthesis

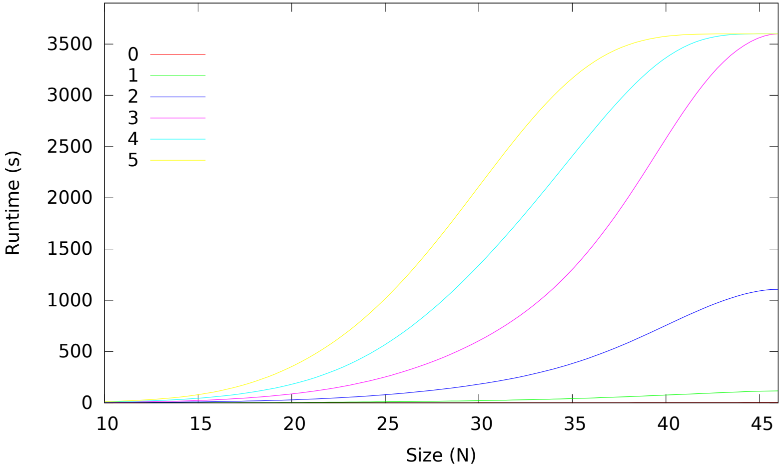
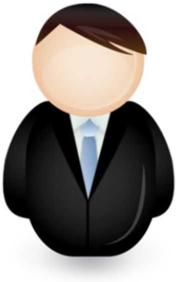


Figure: BR MB Free (obs 30%)



Invitation to Tender

- AUTOGEF (Automated Generation of FDIR)
- FAME (FDIR Development Methodology and V&V)

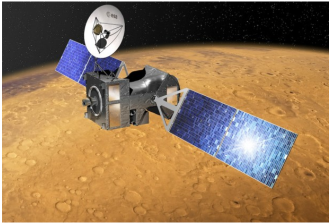
AUTOGEF and FAME

Process and Tools for:

- Definition FDIR Requirements
- Performing Validation, Verification and Synthesis

Tools built on existing ESA COMPASS technology.

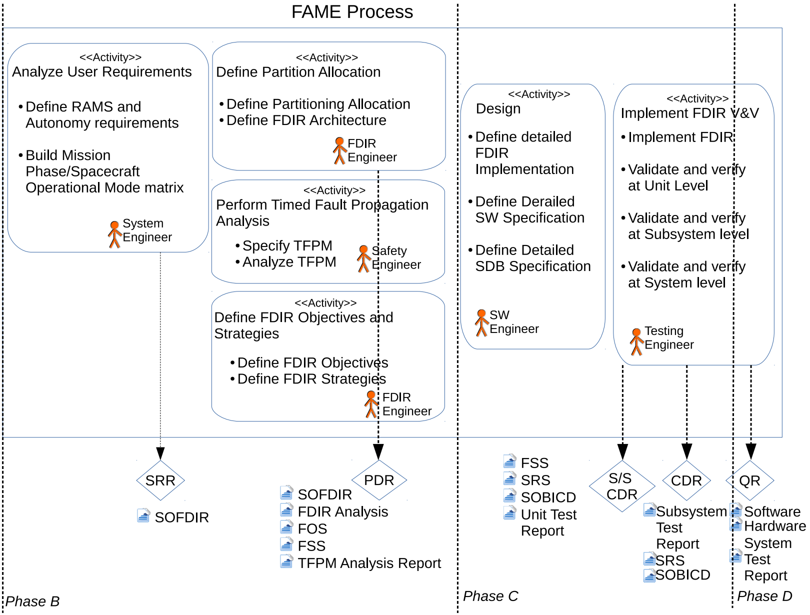
Evaluation



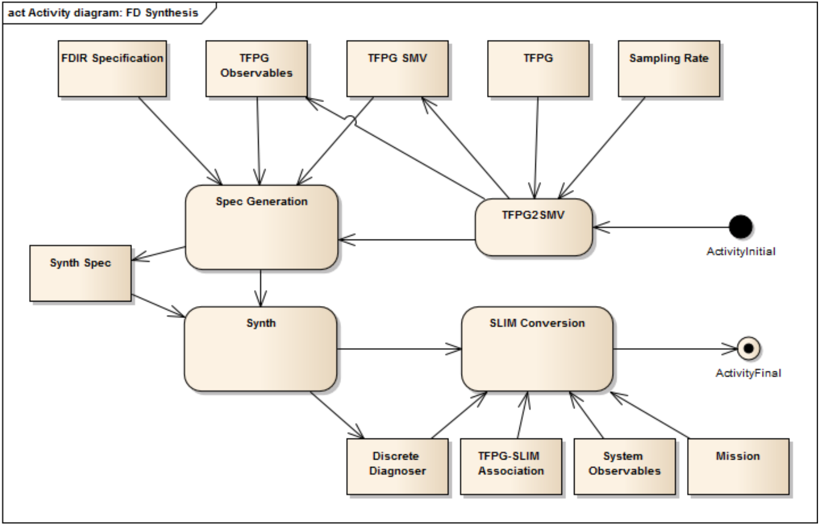
Exomars TGO case-study:

- Usage by non-experts in formal verification
- Synthesis of FDI with 750 states in seconds
- Automated synthesis enables faster design iterations
- Positive feedback by ESA and industrial partners

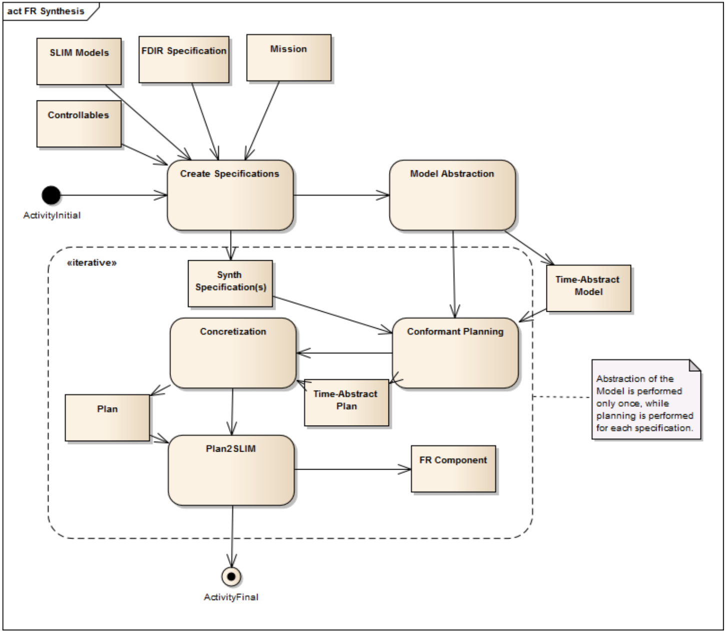
FAME flow


FAME Process
<<Activity>>
Analyze User Requirements
• Define RAMS and Autonomy requirements
• Build Mission Phase/Spacecraft Operational Mode matrix
System Engineer
<<Activity>>
Define Partition Allocation
• Define Partitioning Allocation
• Define FDIR Architecture
FDIR Engineer
<<Activity>>
Perform Timed Fault Propagation Analysis
• Specify TFPM
• Analyze TFPM
Safety Engineer
<<Activity>>
Define FDIR Objectives and Strategies
• Define FDIR Objectives
• Define FDIR Strategies
FDIR Engineer
<<Activity>>
Design
• Define detailed FDIR Implementation
• Define Derailed SW Specification
• Define Detailed SDB Specification
SW Engineer
<<Activity>>
Implement FDIR V&V
• Implement FDIR
• Validate and verify at Unit Level
• Validate and verify at Subsystem level
• Validate and verify at System level
Testing Engineer
SRR
SOFDIR
PDR
SOFDIR
FDIR Analysis
FOS
FSS
TFPM Analysis Report
FSS
SRS
SOBICD
Unit Test Report
S/S CDR
CDR
Subsystem Test Report
SRS
SOBICD
QR
Software
Hardware
System
Test
Report
Phase B
Phase C
Phase D

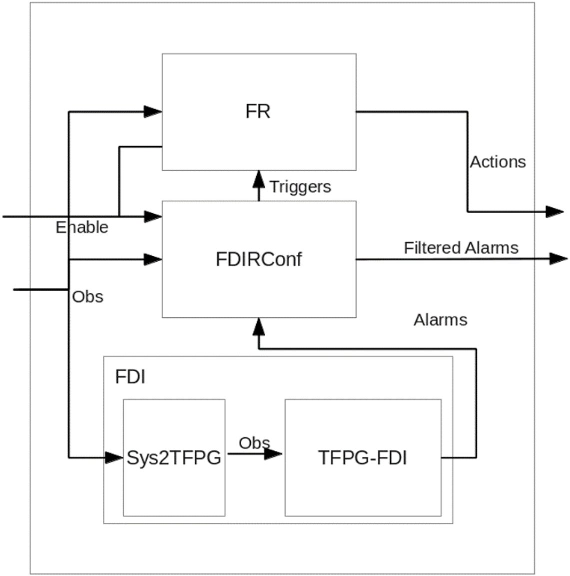
FAME flow


act Activity diagram: FD Synthesis
FDIR Specification
TFPG Observables
TFPG SMV
TFPG
Sampling Rate
Spec Generation
TFPG2SMV
ActivityInitial
Synth Spec
Synth
SLIM Conversion
ActivityFinal
Discrete Diagnoser
TFPG-SLIM Association
System Observables
Mission

FAME flow


act FR Synthesis
SLIM Models
FDIR Specification
Mission
Controllables
Create Specifications
Model Abstraction
ActivityInitial
«iterative»
Synth Specification(s)
Time-Abstract Model
Concretization
Conformant Planning
Plan
Time-Abstract Plan
Abstraction of the Model is performed only once, while planning is performed for each specification.
Plan2SLIM
FR Component
ActivityFinal

FAME flow



FR
Actions
Triggers
Enable
FDIRConf
Filtered Alarms
Obs
Alarms
FDI
Sys2TFPG
Obs
TFPG-FDI

FAME flow

Table: Process break-down

Phase	Steps	COMPASS
Analyze User Requirements	System Modeling & Fault Extension	Formal system modeling – nominal and faulty behavior (in SLIM); automatic model extension
	Formal Analyses	Derive requirements on FDIR design
	Mission Modeling	Definition of mission, phases, and spacecraft configurations
Perform Timed Failure Propagation Analysis	Formal Analyses	Derive information on causality and fault propagation (input for TFPG modeling)
	TFPG Modeling/Synthesis	TFPG modeling, editing, synthesis
	TFPG Analyses	TFPG behavioral validation, TFPG effectiveness validation
Define FDIR Objectives and Strategies	FDIR Requirements Specification	Modeling of FDIR objectives and strategies, definition of pre-existing components to be re-used, and FDIR hierarchy
Design the FDIR	FDIR Modeling/Synthesis	Formal modeling and automatic synthesis of FDIR
	Formal Analyses	FDIR effectiveness verification

Failure Propagation

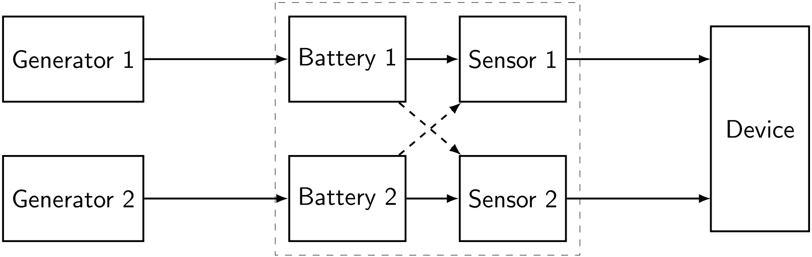
Failures can propagate through-out the system:

Many off-nominal behaviors $\Rightarrow$ **Masking** of faults

Failure Propagation

Failures can propagate through-out the system:

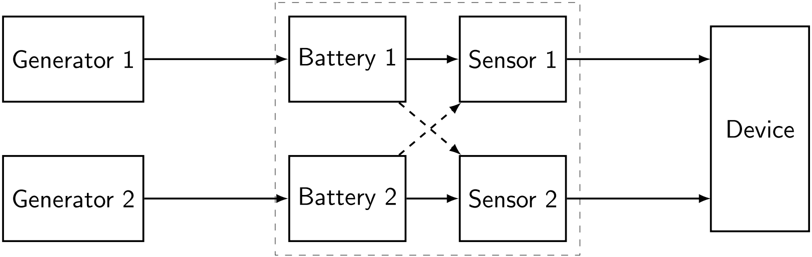
Many off-nominal behaviors $\Rightarrow$ **Masking** of faults



Failure Propagation

Failures can propagate through-out the system:

Many off-nominal behaviors ⇒ **Masking** of faults



Generator broken? Generator AND Sensor broken?

TFPG Analysis

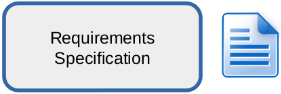
Requirements Specification

What is an **acceptable** Delay?

- Alarm should fire early enough to **prevent** the **propagation** of the failure

⇒ Timed Failure Propagation Graphs

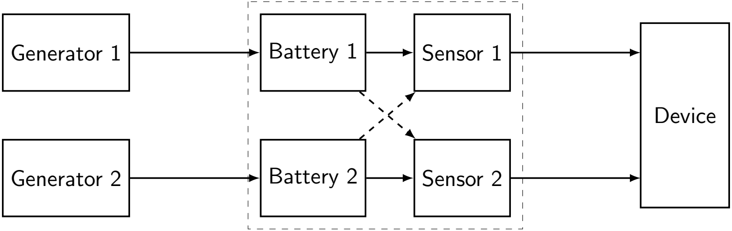
TFPG Analysis



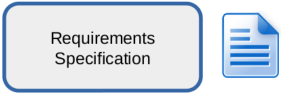
What is an **acceptable** Delay?

- Alarm should fire early enough to **prevent** the **propagation** of the failure

⇒ Timed Failure Propagation Graphs



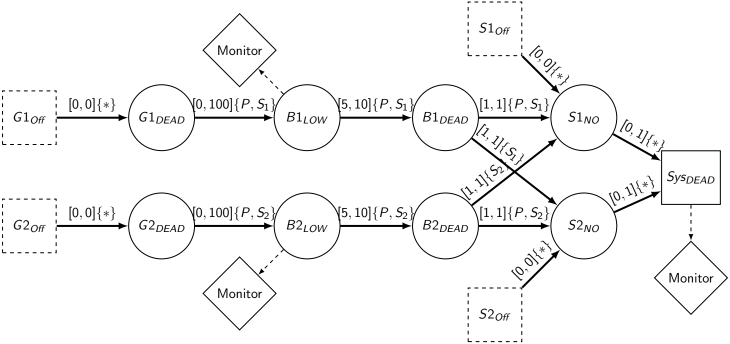
TFPG Analysis



What is an **acceptable** Delay?

- Alarm should fire early enough to **prevent** the **propagation** of the failure

⇒ Timed Failure Propagation Graphs



Goal: Model **propagation** of failures to perform better **reasoning**

TFPG Validation

Encode all executions of the TFPG into a **Satisfiability Modulo Theory** (SMT) formula

TFPG Validation

Encode all executions of the TFPG into a **Satisfiability Modulo Theory** (SMT) formula

- Each node only depends on its **predecessors**
- Separation of **Boolean** and **Temporal** part
- Need to capture only 2 semantics: **OR** and **AND** nodes

TFPG Validation

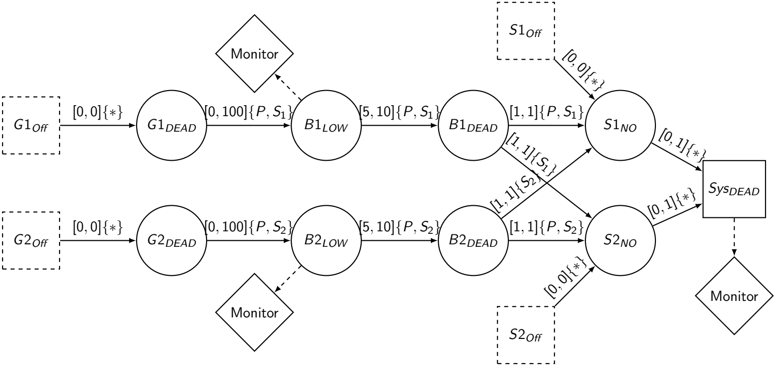
Encode all executions of the TFPG into a **Satisfiability Modulo Theory** (SMT) formula

- Each node only depends on its **predecessors**
- Separation of **Boolean** and **Temporal** part
- Need to capture only 2 semantics: **OR** and **AND** nodes

$$\varphi(\vec{u}, m) = \bigwedge_{v \in D.\ DC(v)=\texttt{OR}} B_{or}(v, m) \ \wedge\ T_{or}(v, m) \ \wedge$$
$$\bigwedge_{v \in D.\ DC(v)=\texttt{AND}} B_{and}(v, m) \ \wedge T_{and}(v, m)$$

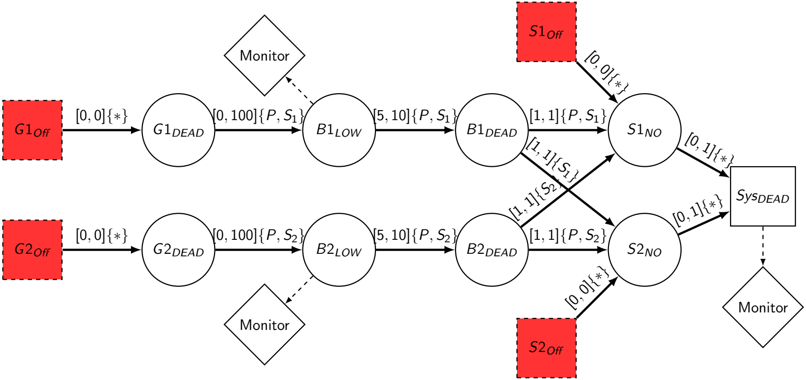
where $\vec{u}$ vector of activation states and times, m mode of the system, D set of discrepancies.

Timed Failure Propagation Graphs



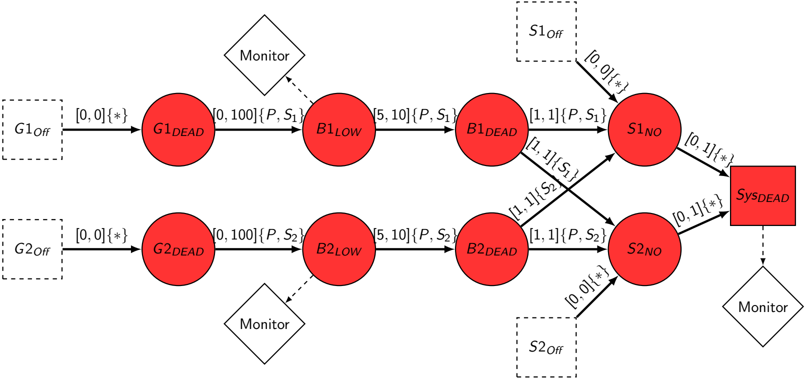
Captures the **causal** and **temporal** relation of **off-nominal** conditions

Timed Failure Propagation Graphs



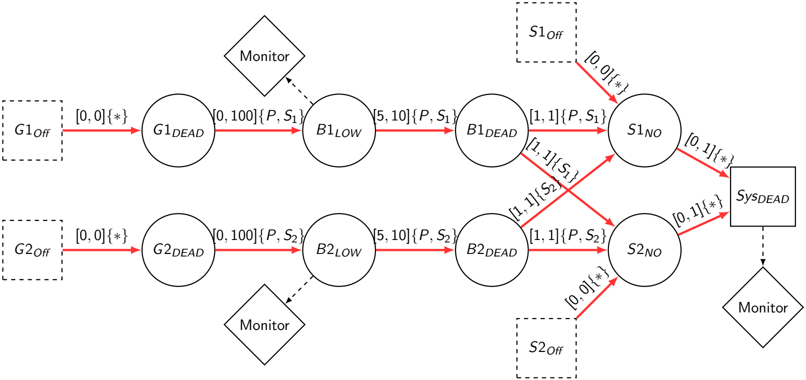
Failure modes describe the basic faults

Timed Failure Propagation Graphs



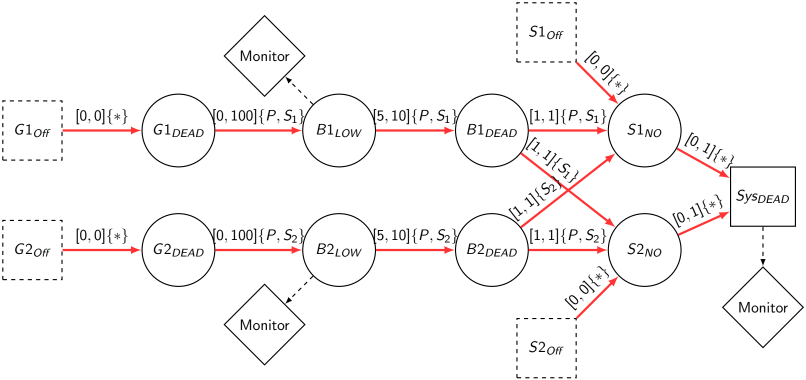
Discrepancies describe off-nominal conditions

Timed Failure Propagation Graphs



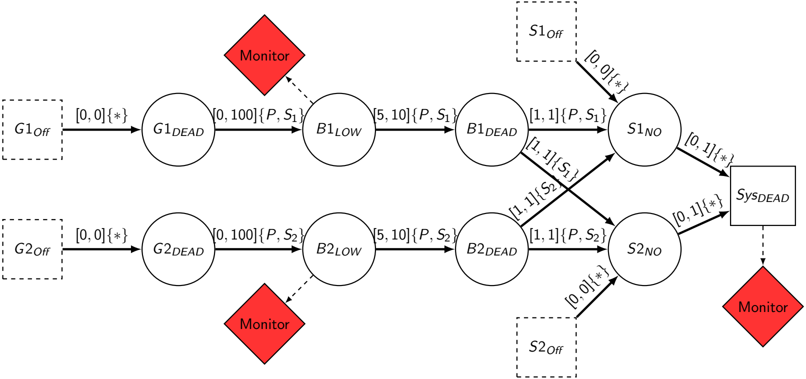
Propagations are indicated with edges:

Timed Failure Propagation Graphs



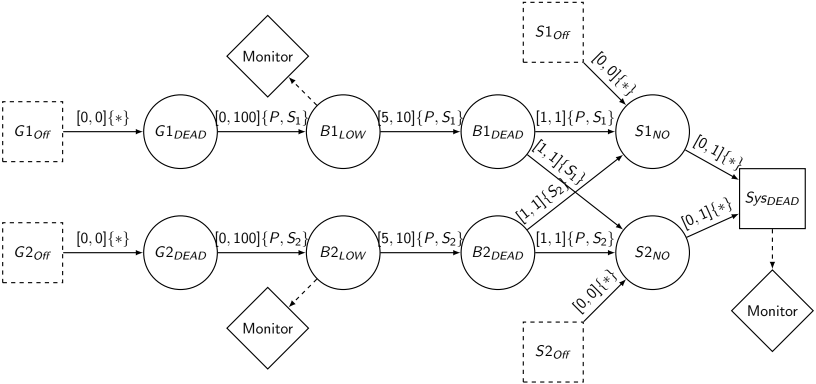
Propagations are indicated with edges: **Time-bounds** and **Modes**

Timed Failure Propagation Graphs

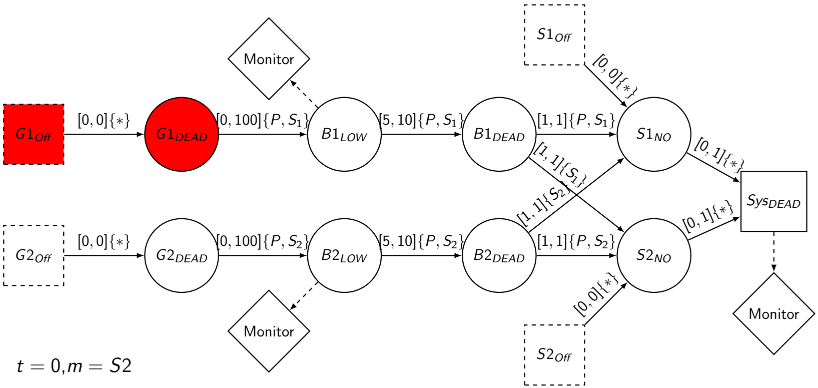


Monitors observe a discrepancy.

Example

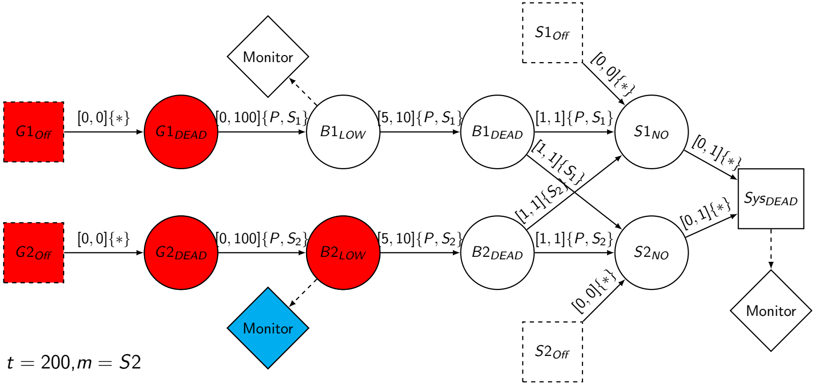

Monitor
$S1_{Off}$
$[0,0]\{*\}$
$G1_{Off}$
$[0,0]\{*\}$
$G1_{DEAD}$
$[0,100]\{P,S_1\}$
$B1_{LOW}$
$[5,10]\{P,S_1\}$
$B1_{DEAD}$
$[1,1]\{P,S_1\}$
$S1_{NO}$
$[0,1]\{*\}$
$[1,1]\{S_1\}$
$[1,1]\{S_2\}$
Sys_{DEAD}
$[0,1]\{*\}$
$G2_{Off}$
$[0,0]\{*\}$
$G2_{DEAD}$
$[0,100]\{P,S_2\}$
$B2_{LOW}$
$[5,10]\{P,S_2\}$
$B2_{DEAD}$
$[1,1]\{P,S_2\}$
$S2_{NO}$
$[0,0]\{*\}$
Monitor
Monitor
$S2_{Off}$

Example



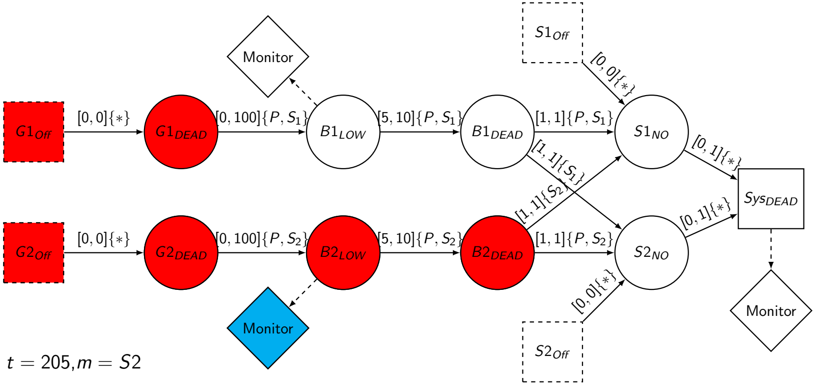
$t = 0, m = S2$

Example



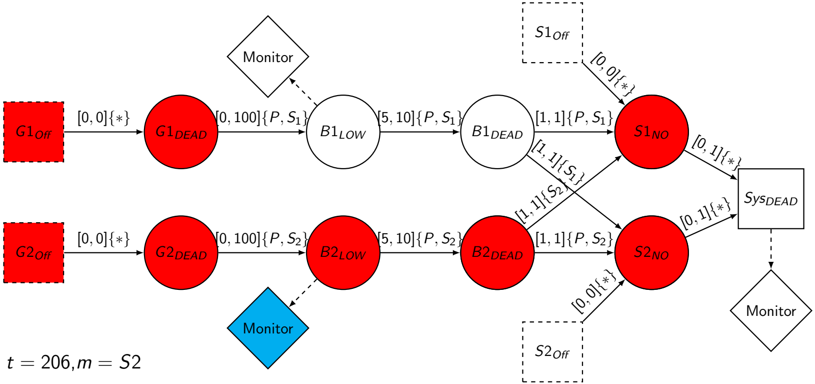
$t = 200, m = S2$

Example



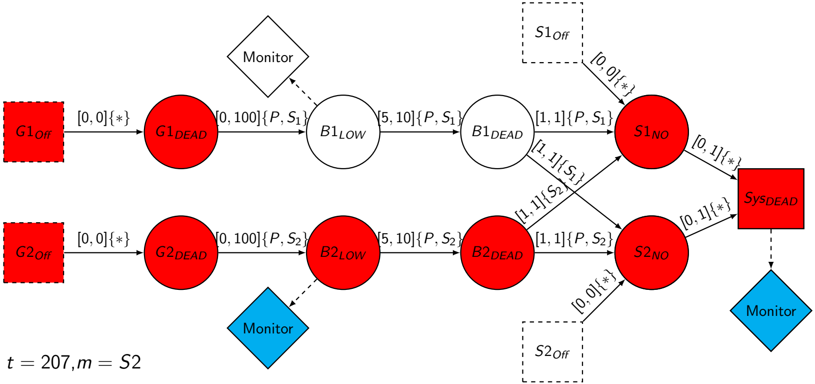
$t = 205, m = S2$

Example



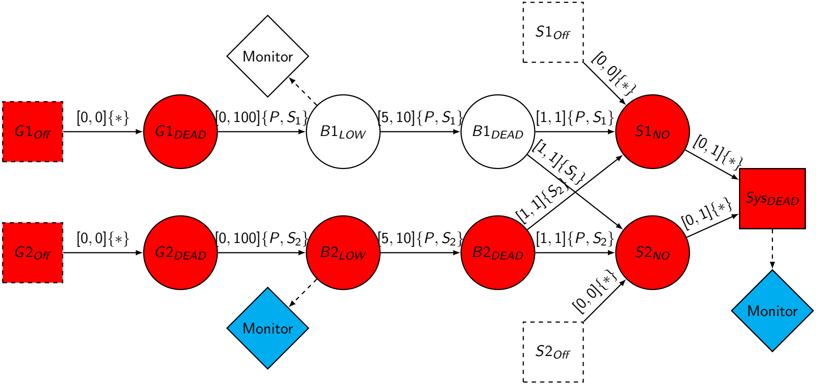
$t = 206, m = S2$

Example



$t = 207, m = S2$

Example



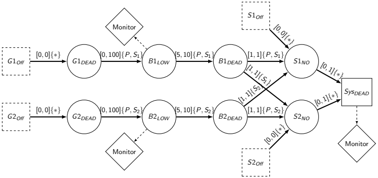
Monitor
S1Off
[0,0]{*}
G1Off
[0,0]{*}
G1DEAD
[0,100]{P,S1}
B1LOW
[5,10]{P,S1}
B1DEAD
[1,1]{P,S1}
S1NO
[0,1]{*}
[1,1]{S1}
[1,1]{S2}
SysDEAD
[0,1]{*}
G2Off
[0,0]{*}
G2DEAD
[0,100]{P,S2}
B2LOW
[5,10]{P,S2}
B2DEAD
[1,1]{P,S2}
S2NO
[0,0]{*}
Monitor
S2Off
Monitor

Semantics

- **Active** discrepancy: the failure effects reached that node (permanent effect);
- **Active Edge**: the starting node is active, and the current mode is compatible with the edge mode ($m \in EM(e)$);
- The **activation time** t' of an `OR` node must satisfy $e.tmin \leq t' - t \leq e.tmax$, where t is the activation time of its predecessor;
- The **activation time** t' of an `AND` node is the composition of the activation periods for each incoming edge; the *tmax* can be violated by all but one of the predecessors;
- **Memoryless Edges**: if deactivated (due to mode change) the propagation stops and resets.

Problem

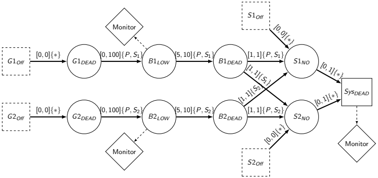
- Are all (important) **executions** of the system **captured** by the TFPG? Are the **models** of the system and TFPG **aligned**?



- Model-Based Diagnosis: only **as good as the model**

Problem

- Are all (important) **executions** of the system **captured** by the TFPG? Are the **models** of the system and TFPG **aligned**?



- Model-Based Diagnosis: only **as good as the model**

The TFPG must be **validated**!

Related Work

- Introduced by Vanderbilt University in 2003 (Abdelwahed, Karsai, and Biswas)
- Used both in hardware, system and software **monitoring**, **diagnosis**, **prognosis**
- Studied in **aerospace** setting: Boeing, NASA, ESA

Related Work

- Introduced by Vanderbilt University in 2003 (Abdelwahed, Karsai, and Biswas)
- Used both in hardware, system and software **monitoring**, **diagnosis**, **prognosis**
- Studied in **aerospace** setting: Boeing, NASA, ESA
- Validation: **Alarm-Sequence Maturation**: Data-driven correction of the TFPG

$\Rightarrow$ Data-driven = The system must already be operational!

How do you **validate** a TFPG?

Reasoning Tasks

- Necessity, Possibility
- Refinement
- Diagnosability

Reasoning Tasks

- Necessity, Possibility : Do all (resp. some) executions of the TFPG satisfy a given condition?
- Refinement
- Diagnosability

Reasoning Tasks

- Necessity, Possibility : Do all (resp. some) executions of the TFPG satisfy a given condition?
- Refinement : Given two TFPG, can every execution of one TFPG be mapped on the other?
- Diagnosability

Reasoning Tasks

- Necessity, Possibility : Do all (resp. some) executions of the TFPG satisfy a given condition?
- Refinement : Given two TFPG, can every execution of one TFPG be mapped on the other?
- Diagnosability : Given a diagnosis condition, is it possible to detect the occurrence of the condition given the available monitors?

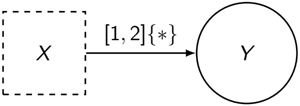
Reasoning Tasks

- Necessity, Possibility : Do all (resp. some) executions of the TFPG satisfy a given condition?
- Refinement : Given two TFPG, can every execution of one TFPG be mapped on the other?
- Diagnosability : Given a diagnosis condition, is it possible to detect the occurrence of the condition given the available monitors?

Frozen Mode Assumption: Mode does not change

Traces of a TFPG

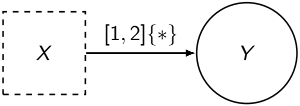
TFPG as the **set of traces** satisfying the definition.



X State	X Time	Y State	Y Time
Off	–	Off	–
On	0	On	1
On	1	On	2
...	...	...	...

Traces of a TFPG

TFPG as the **set of traces** satisfying the definition.

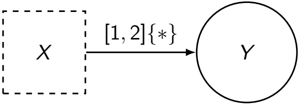


X State	X Time	Y State	Y Time
Off	–	Off	–
On	0	On	1
On	1	On	2
...	...	...	...

- **Infinite** Table

Traces of a TFPG

TFPG as the **set of traces** satisfying the definition.

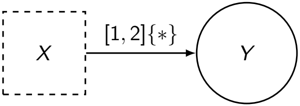


X State	X Time	Y State	Y Time
Off	–	Off	–
On	0	On	1
On	1	On	2
...	...	...	...

- **Infinite** Table
- X State = On, X Time = 0, Y State = Off is **not** in the table

Traces of a TFPG

TFPG as the **set of traces** satisfying the definition.



X State	X Time	Y State	Y Time
Off	–	Off	–
On	0	On	1
On	1	On	2
...	...	...	...

- **Infinite** Table
- X State = On, X Time = 0, Y State = Off is **not** in the table

Goal: Symbolic representation of this Infinite Table.

Boolean Logic and SAT

Boolean logic represents similar tables:

p	q	$p \wedge q$
F	F	F
F	T	F
T	F	F
T	T	T

Boolean Logic and SAT

Boolean logic represents similar tables:

p	q	p ∧ q
T	T	T

$p \wedge q$ represents only the rows in which it is **T**rue.

Boolean Logic and SAT

Boolean logic represents similar tables:

p	q	p ∧ q
T	T	T

$p \wedge q$ represents only the rows in which it is **T**rue.

SAT

Given a formula φ, the boolean satisfiability problem (SAT) is the problem of finding a model (i.e., a line in which the formula is True)

Satisfiability Modulo Theory (SMT)

SMT extends from boolean atom to **Theory** atoms

Satisfiability Modulo Theory (SMT)

SMT extends from boolean atom to **Theory** atoms

E.g., Theory of Rational Arithmetic ($\mathcal{LA}(\mathbb{Q})$)

$$\varphi \doteq (x > 2) \land (x < 8) \land ((x < 1) \lor (x > 7))$$

Satisfiability Modulo Theory (SMT)

SMT extends from boolean atom to **Theory** atoms

E.g., Theory of Rational Arithmetic ($\mathcal{LA}(\mathbb{Q})$)

$$\varphi \doteq (x > 2) \wedge (x < 8) \wedge ((x < 1) \vee (x > 7))$$

SAT: $\mathrm{x} \doteq 7.5$ (One of the infinitely many models)

Satisfiability Modulo Theory (SMT)

SMT extends from boolean atom to **Theory** atoms

E.g., Theory of Rational Arithmetic ($\mathcal{LA}(\mathbb{Q})$)

$$\varphi \doteq (x > 2) \wedge (x < 8) \wedge ((x < 1) \vee (x > 7))$$

SAT: $x \doteq 7.5$ (One of the infinitely many models)

$$\phi \doteq (x + y > 2) \wedge (x < 0) \wedge (y < 0)$$

Satisfiability Modulo Theory (SMT)

SMT extends from boolean atom to **Theory** atoms

E.g., Theory of Rational Arithmetic ($\mathcal{LA}(\mathbb{Q})$)

$$\varphi \doteq (x > 2) \wedge (x < 8) \wedge ((x < 1) \vee (x > 7))$$

SAT: $\mathtt{x} \doteq 7.5$ (One of the infinitely many models)

$$\phi \doteq (x + y > 2) \wedge (x < 0) \wedge (y < 0)$$

UNSAT: No model exists

Small Digression

Why SMT ?

Example Theories in SMT

- **Difference logic** ($\mathcal{DL}$):
  $((x = y) \wedge (y - z \leq 4)) \rightarrow (x - z \leq 6)$
- **Linear arithmetic over the rationals** ($\mathcal{LA}(\mathbb{Q})$):
  $(T_\delta \rightarrow (s_1 = s_0 + 3.4t - 3.4t_0)) \wedge (\neg T_\delta \rightarrow (s_1 = s_0))$

Examples from R. Sebastiani (`http://disi.unitn.it/~rseba/DIDATTICA/SAT_BASED14/2_smt_slides.pdf`)

Example Theories in SMT

- **Difference logic** ($\mathcal{DL}$):
  $((x = y) \wedge (y - z \leq 4)) \rightarrow (x - z \leq 6)$
- **Linear arithmetic over the rationals** ($\mathcal{LA}(\mathbb{Q})$):
  $(T_\delta \rightarrow (s_1 = s_0 + 3.4t - 3.4t_0)) \wedge (\neg T_\delta \rightarrow (s_1 = s_0))$
- Equality and Uninterpreted Functions ($\mathcal{EUF}$):
  $((x = y) \wedge (y = f(z))) \rightarrow (g(x) = g(f(z)))$
- Arrays ($\mathcal{AR}$):
  $(i = j) \vee \mathit{read}(\mathit{write}(a, i, e), j) = \mathit{read}(a, j)$
- Bit vectors ($\mathcal{BV}$):
  $x_{[16]}[15:0] = (y_{[16]}[15:8] :: z_{[16]}[7:0]) << w_{[8]}[3:0]$
- Non-Linear arithmetic over the reals ($\mathcal{NLA}(\mathbb{R})$):
  $((c = ab) \wedge (a_1 = a - 1) \wedge (b_1 = b + 1)) \rightarrow (c = a_1 b_1 + 1)$

Examples from R. Sebastiani (`http://disi.unitn.it/~rseba/DIDATTICA/SAT_BASED14/2_smt_slides.pdf`)

SMT in practice

- Many **Applications**: Hardware and Software Model Checking, Automatic test generation, Constraint Programming, Answer Set Programming, $\cdots$
- Many **Solvers**: MathSAT (FBK – Italy), CVC4 (NYU/University of Iowa), Z3 (Microsoft), Yices (SRI), Boolector (JKU – Austria), $\cdots$

Beyond the SAT/UNSAT answer

- Model construction
- Proofs of unsatisfiability
- Optimization
- Model enumeration (All-SMT)

Back to **TFPGs**

Encoding

Observations:

- Each node only depends on its **predecessors**
- Separation of **Boolean** and **Temporal** part
- Need to capture only 2 semantics: **OR** and **AND** nodes

Encoding

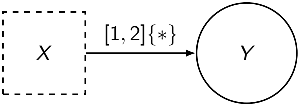
Observations:

- Each node only depends on its **predecessors**
- Separation of **Boolean** and **Temporal** part
- Need to capture only 2 semantics: **OR** and **AND** nodes

$$\varphi(\vec{u}, m) = \bigwedge_{v \in D.\ DC(v)=\texttt{OR}} B_{or}(v, m) \ \wedge \ T_{or}(v, m) \ \wedge$$
$$\bigwedge_{v \in D.\ DC(v)=\texttt{AND}} B_{and}(v, m) \ \wedge T_{and}(v, m)$$

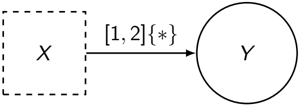
where $\vec{u}$ vector of activation states and times, m mode of the system, D set of discrepancies.

Example Encoding



$$\varphi(\vec{u}, m) = B_{or}(Y, m) \ \wedge \ T_{or}(Y, m)$$

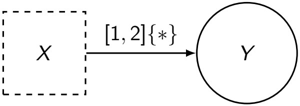
Example Encoding



$$\varphi(\vec{u}, m) = B_{or}(Y, m) \ \wedge \ T_{or}(Y, m)$$

$$B_{or}(Y, m) = \vec{ud}(Y) \leftrightarrow [\vec{ud}(X) \ \wedge M((X, Y), m)]$$

Example Encoding

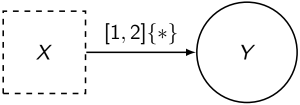


$$\varphi(\vec{u}, m) = B_{or}(Y, m) \ \wedge \ T_{or}(Y, m)$$

$$B_{or}(Y, m) = \vec{ud}(Y) \leftrightarrow [\vec{ud}(X) \ \wedge M((X, Y), m)]$$

$$T_{or}(Y, m) = \vec{ud}(Y) \rightarrow \Big(\vec{ud}(X) \ \wedge \ (\vec{udt}(Y) - \vec{udt}(X)) \in ET((X, Y))\Big)$$

Example Encoding



$$\begin{aligned}\varphi(\vec{u}, m) =& \vec{ud}(Y) \leftrightarrow \vec{ud}(X) \wedge \\ & \vec{ud}(Y) \rightarrow \left(\vec{ud}(X) \ \wedge \ (\vec{udt}(Y) - \vec{udt}(X)) \in [1, 2]\right)\end{aligned}$$

Full Encoding

$$\varphi(\vec{u}, m) = \bigwedge_{v \in D.\ DC(v) = \texttt{OR}} B_{or}(v, m) \ \wedge \ T_{or}(v, m) \wedge$$
$$\bigwedge_{v \in D.\ DC(v) = \texttt{AND}} B_{and}(v, m) \ \wedge T_{and}(v, m)$$

Full Encoding

$$\varphi(\vec{u}, m) = \bigwedge_{v \in D.\ DC(v) = \texttt{OR}} B_{or}(v, m) \ \wedge \ T_{or}(v, m) \ \wedge$$

$$\bigwedge_{v \in D.\ DC(v) = \texttt{AND}} B_{and}(v, m) \ \wedge T_{and}(v, m)$$

Size: $O(|E|)$ $\mathcal{RDL}$ atoms

Full Encoding

$$B_{or}(v,m) = \vec{ud}(v) \leftrightarrow \bigvee_{(w,v)\in E} [\vec{ud}(w) \ \wedge M((w,v),m)]$$

$$B_{and}(v,m) = \vec{ud}(v) \leftrightarrow \bigwedge_{(w,v)\in E} [\vec{ud}(w) \ \wedge M((w,v),m)]$$

Full Encoding

$$T_{or}(v, m) = \vec{ud}(v) \rightarrow [$$

$$\bigvee_{(w,v)\in E} \Big(\vec{ud}(w) \ \wedge \ (\vec{udt}(v) - \vec{udt}(w)) \in ET((w, v))\Big) \ \wedge$$

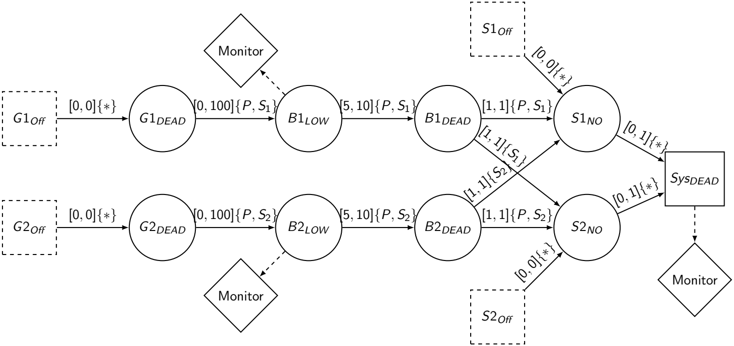
$$\bigwedge_{(w,v)\in E} \Big(\vec{ud}(w) \rightarrow \ (\vec{udt}(v) - \vec{udt}(w)) \leq t_{max}((w, v))\Big) \ ]$$

$$T_{and}(v, m) = \vec{ud}(v) \rightarrow [$$

$$\bigwedge_{(w,v)\in E} \Big(\vec{ud}(w) \ \wedge (\vec{udt}(v) - \vec{udt}(w)) \geq t_{min}((w, v))\Big) \ \wedge$$

$$\bigvee_{(w,v)\in E} \Big(\vec{ud}(v) \wedge \ (\vec{udt}(v) - \vec{udt}(w)) \leq t_{max}((w, v))\Big) \ ]$$

Necessity and Possibility

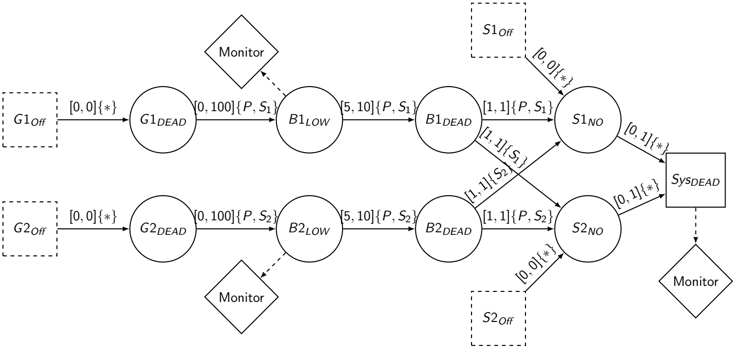


Possibility: Can $B2_{LOW}$ be active?

$$\varphi(\vec{u}, m) \wedge \tau(\vec{u}, m)$$

with $\tau(\vec{u}, m) = \vec{ud}(B2_{LOW})$ is **SAT** ?

Necessity and Possibility

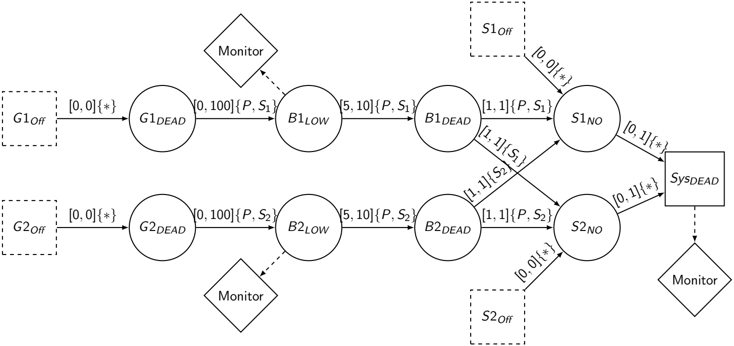


Possibility: Can $B2_{LOW}$ be active?

$$\varphi(\vec{u}, m) \wedge \tau(\vec{u}, m)$$

with $\tau(\vec{u}, m) = \vec{ud}(B2_{LOW})$ is **SAT** ? YES

Necessity and Possibility

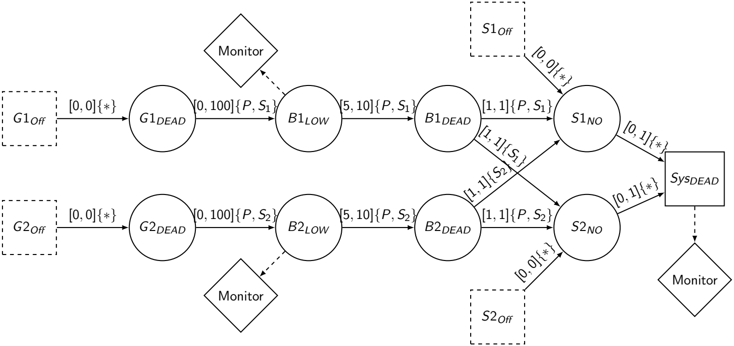


Possibility: Can $B2_{LOW}$ be active in mode $Secondary_1$?

$$\varphi(\vec{u}, m) \wedge \tau(\vec{u}, m) \wedge m = Secondary_1$$

with $\tau(\vec{u}, m) = \vec{ud}(B2_{LOW})$ is **SAT**?

Necessity and Possibility

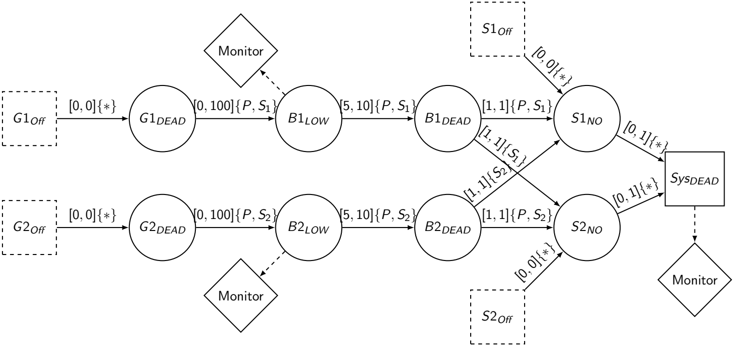


Possibility: Can $B2_{LOW}$ be active in mode $Secondary_1$?

$$\varphi(\vec{u}, m) \wedge \tau(\vec{u}, m) \wedge m = Secondary_1$$

with $\tau(\vec{u}, m) = \vec{ud}(B2_{LOW})$ is **SAT**? NO

Necessity and Possibility

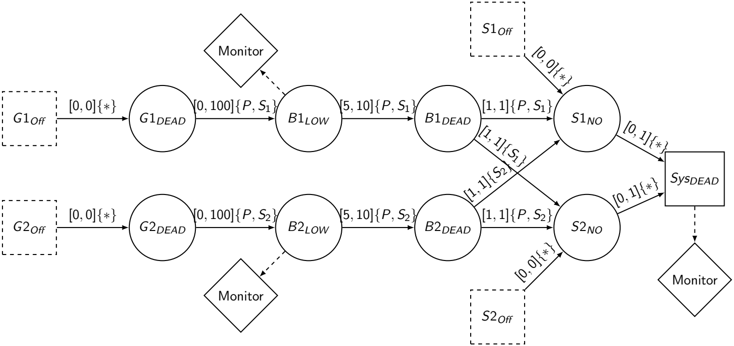


Necessity: For $B2_{LOW}$ to be active, $G2_{Off}$ must be active.

$$\varphi(\vec{u}, m) \wedge \neg\tau(\vec{u}, m)$$

with $\tau(\vec{u}, m) = \vec{ud}(B2_{LOW}) \rightarrow \vec{ud}(G2_{Off})$ is **UNSAT**?

Necessity and Possibility



Necessity: For $B2_{LOW}$ to be active, $G2_{Off}$ must be active.

$$\varphi(\vec{u}, m) \wedge \neg\tau(\vec{u}, m)$$

with $\tau(\vec{u}, m) = \vec{ud}(B2_{LOW}) \rightarrow \vec{ud}(G2_{Off})$ is **UNSAT**? YES

Refinement

- After **modifying** the TFPG, what can we say on the relation between the original and the new?

Refinement

- After **modifying** the TFPG, what can we say on the relation between the original and the new?
- Refinement:

Refinement

- After **modifying** the TFPG, what can we say on the relation between the original and the new?
- Refinement:

  Given two TFPGs G_1, G_2 and a (partial) mapping $\gamma(\vec{u}_1, \vec{u}_2)$ between their nodes, we say that G_1 refines G_2 if **every trace of G_1 can be mapped to a trace of G_2**

Refinement

- After **modifying** the TFPG, what can we say on the relation between the original and the new?
- Refinement:

  Given two TFPGs G_1, G_2 and a (partial) mapping $\gamma(\vec{u}_1, \vec{u}_2)$ between their nodes, we say that G_1 refines G_2 if **every trace of G_1 can be mapped to a trace of G_2**

$$\forall \vec{u}_1, m.\ \varphi_{G1}(\vec{u}_1, m) \rightarrow \exists \vec{u}_2.(\gamma(\vec{u}_1, \vec{u}_2) \wedge \varphi_{G2}(\vec{u}_2, m))$$

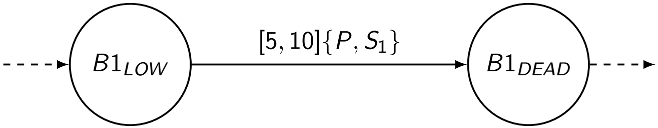
Refinement

- After **modifying** the TFPG, what can we say on the relation between the original and the new?
- Refinement:

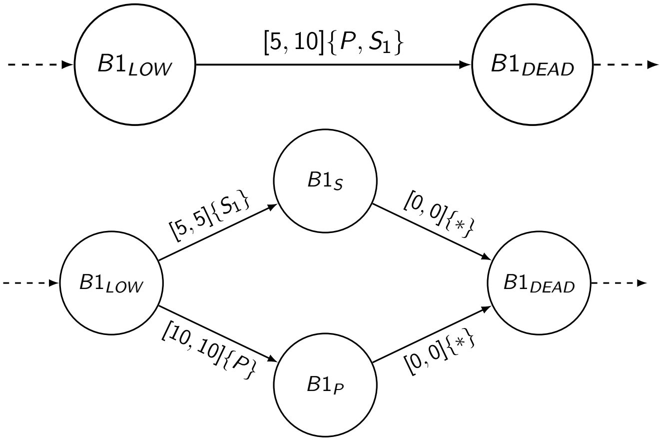
  Given two TFPGs G_1, G_2 and a (partial) mapping $\gamma(\vec{u}_1, \vec{u}_2)$ between their nodes, we say that G_1 refines G_2 if **every trace of G_1 can be mapped to a trace of G_2**

$$\varphi_{G1}(\vec{u}_1, m) \wedge \forall \vec{u}_2 . \neg(\gamma(\vec{u}_1, \vec{u}_2) \wedge \varphi_{G2}(\vec{u}_2, m))$$

Refinement Example



Refinement Example



Diagnosis and Observations

- **Monitors** tell us when a discrepancy is activated;
- **Diagnosis**: Infer the state of the system (e.g., other discrepancies and failure modes) based on the observations (Monitors).

Partially-Observable TFPG:

$$\psi(\vec{o}, \vec{u}, m) = \varphi(\vec{u}, m) \wedge \bigwedge_{v \in D . DS(v) = \mathtt{M}} (\vec{od}(v) = \vec{ud}(v) \wedge \vec{odt}(v) = \vec{udt}(v))$$

Diagnosis and Observations

- **Monitors** tell us when a discrepancy is activated;
- **Diagnosis**: Infer the state of the system (e.g., other discrepancies and failure modes) based on the observations (Monitors).

Partially-Observable TFPG:

$$\psi(\vec{o}, \vec{u}, m) = \varphi(\vec{u}, m) \wedge \bigwedge_{v \in D . DS(v) = \mathtt{M}} (\vec{od}(v) = \vec{ud}(v) \wedge \vec{odt}(v) = \vec{udt}(v))$$

What if monitors can break?

Diagnosis and Observations

- **Monitors** tell us when a discrepancy is activated;
- **Diagnosis**: Infer the state of the system (e.g., other discrepancies and failure modes) based on the observations (Monitors).

Partially-Observable TFPG:

$$\psi(\vec{o}, \vec{u}, m, \vec{h}) = \varphi(\vec{u}, m) \wedge \bigwedge_{v \in D.DS(v)=\mathtt{M}} \vec{h}(v) \rightarrow (\vec{od}(v) = \vec{ud}(v) \wedge \vec{odt}(v) = \vec{udt}(v))$$

What if monitors can break? $\Rightarrow$ Health Variables!

Diagnosability

- Diagnosis condition $\beta(\vec{u})$

Diagnosability

- Diagnosis condition $\beta(\vec{u})$
  E.g., $\beta = \vec{ud}(fm) \wedge 5 \leq \vec{udt}(fm) \leq 10$

Diagnosability

- Diagnosis condition $\beta(\vec{u})$
  E.g., $\beta = \vec{ud}(fm) \wedge 5 \leq \vec{udt}(fm) \leq 10$
  E.g., $\beta(\vec{u}) = \vec{ud}(D1) \wedge \vec{ud}(D2) \wedge \vec{udt}(D1) \leq \vec{udt}(D2)$

Diagnosability

- Diagnosis condition $\beta(\vec{u})$
  E.g., $\beta = \vec{ud}(fm) \wedge 5 \leq \vec{udt}(fm) \leq 10$
  E.g., $\beta(\vec{u}) = \vec{ud}(D1) \wedge \vec{ud}(D2) \wedge \vec{udt}(D1) \leq \vec{udt}(D2)$
- **Diagnosability**: Can we always detect β using the monitors?

Diagnosability

- Diagnosis condition $\beta(\vec{u})$
  E.g., $\beta = \vec{ud}(fm) \wedge 5 \leq \vec{udt}(fm) \leq 10$
  E.g., $\beta(\vec{u}) = \vec{ud}(D1) \wedge \vec{ud}(D2) \wedge \vec{udt}(D1) \leq \vec{udt}(D2)$
- **Diagnosability**: Can we always detect β using the monitors?
- Twin-plant construction: Is this **UNSAT**?

$$\psi(\vec{o}, \vec{u}_1, m, \vec{h}_1) \wedge \psi(\vec{o}, \vec{u}_2, m, \vec{h}_2) \wedge$$
$$\beta(\vec{u}_1) \wedge \neg\beta(\vec{u}_2) \wedge Healthy(\vec{h}_1, \vec{h}_2)$$

Diagnosability

- Diagnosis condition $\beta(\vec{u})$
  E.g., $\beta = \vec{ud}(fm) \wedge 5 \leq \vec{udt}(fm) \leq 10$
  E.g., $\beta(\vec{u}) = \vec{ud}(D1) \wedge \vec{ud}(D2) \wedge \vec{udt}(D1) \leq \vec{udt}(D2)$
- **Diagnosability**: Can we always detect β using the monitors?
- Twin-plant construction: Is this **UNSAT**?

$$\psi(\vec{o}, \vec{u}_1, m, \vec{h}_1) \wedge \psi(\vec{o}, \vec{u}_2, m, \vec{h}_2) \wedge$$
$$\beta(\vec{u}_1) \wedge \neg\beta(\vec{u}_2) \wedge Healthy(\vec{h}_1, \vec{h}_2)$$

A model for the formula is a **critical pair**: A pair of observationally-equivalent traces s.t. one satisfies β but the other does not.

Reasoning Tasks (Revisited)

- Possibility:
- Necessity:
- Refinement:

- Diagnosability:

Reasoning Tasks (Revisited)

- Possibility: SAT: $\varphi(\vec{u}, m) \wedge \tau(\vec{u}, m)$
- Necessity:
- Refinement:
- Diagnosability:

Reasoning Tasks (Revisited)

- Possibility: SAT: $\varphi(\vec{u}, m) \wedge \tau(\vec{u}, m)$
- Necessity: UNSAT: $\varphi(\vec{u}, m) \wedge \neg\tau(\vec{u}, m)$
- Refinement:
- Diagnosability:

Reasoning Tasks (Revisited)

- Possibility: SAT: $\varphi(\vec{u}, m) \wedge \tau(\vec{u}, m)$
- Necessity: UNSAT: $\varphi(\vec{u}, m) \wedge \neg\tau(\vec{u}, m)$
- Refinement: UNSAT:

$$\varphi_{G1}(\vec{u}_1, m) \wedge \forall \vec{u}_2 . \neg(\gamma(\vec{u}_1, \vec{u}_2) \wedge \varphi_{G2}(\vec{u}_2, m))$$

- Diagnosability:

Reasoning Tasks (Revisited)

- Possibility: SAT: $\varphi(\vec{u}, m) \wedge \tau(\vec{u}, m)$
- Necessity: UNSAT: $\varphi(\vec{u}, m) \wedge \neg\tau(\vec{u}, m)$
- Refinement: UNSAT:

$$\varphi_{G1}(\vec{u}_1, m) \wedge \forall \vec{u}_2 . \neg(\gamma(\vec{u}_1, \vec{u}_2) \wedge \varphi_{G2}(\vec{u}_2, m))$$

- Diagnosability: UNSAT:

$$\psi(\vec{o}, \vec{u}_1, m, \vec{h}_1) \wedge \psi(\vec{o}, \vec{u}_2, m, \vec{h}_2) \wedge \beta(\vec{u}_1) \wedge \neg\beta(\vec{u}_2) \wedge \mathit{Healthy}(\vec{h}_1, \vec{h}_2)$$

Reasoning Tasks (Revisited)

- Possibility: SAT: $\varphi(\vec{u}, m) \wedge \tau(\vec{u}, m)$
- Necessity: UNSAT: $\varphi(\vec{u}, m) \wedge \neg\tau(\vec{u}, m)$
- Refinement: UNSAT:

$$\varphi_{G1}(\vec{u}_1, m) \wedge \forall \vec{u}_2 . \neg(\gamma(\vec{u}_1, \vec{u}_2) \wedge \varphi_{G2}(\vec{u}_2, m))$$

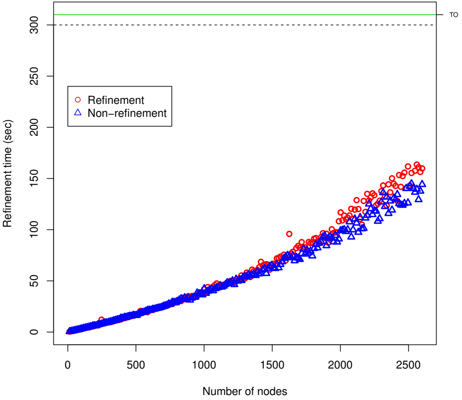
- Diagnosability: UNSAT:

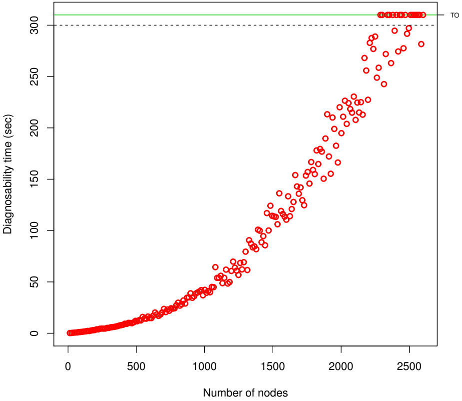
$$\psi(\vec{o}, \vec{u}_1, m, \vec{h}_1) \wedge \psi(\vec{o}, \vec{u}_2, m, \vec{h}_2) \wedge \beta(\vec{u}_1) \wedge \neg\beta(\vec{u}_2) \wedge Healthy(\vec{h}_1, \vec{h}_2)$$

- . . .

Experimental Experience

Case-Study (FAME Project) + Random Benchmark





Easily handle **more than 2000** nodes

Future

- More **case-studies**
- Integration with other tools (E.g., **user-friendly** interface)
- **Synthesis** from models
- Automata-based techniques to remove **frozen mode assumption** (Preliminary work using NuSMV)
- Framework to explore the design space: **parameter synthesis**

Summary

- TFPG describe **temporal** and **causal** relation of off-nominal conditions in a system;
- **Validation** is important but was mainly **unexplored**;
- **Possible executions** of the TFPG as an **SMT** formula;
- Uniform encoding for multiple types of reasoning tasks: **Necessity**, **Possibility**, **Refinement**, **Diagnosability** and more ...
- Experimental evaluation shows **applicability** of the approach for **examples of considerable size**.

References I

J. Ezekiel, A. Lomuscio, L. Molnar, and S.M. Veres.
Verifying Fault Tolerance and Self-Diagnosability of an Autonomous Underwater Vehicle.
In IJCAI, pages 1659–1664, 2011.

Xiaowei Huang.
Diagnosability in concurrent probabilistic systems.
In Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems, 2013.

Shengbing Jiang and Ratnesh Kumar.
Failure diagnosis of discrete event systems with linear-time temporal logic fault specifications.
In IEEE Transactions on Automatic Control, pages 128–133, 2001.

References II

M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis.
Failure diagnosis using discrete-event models.
4(2):105–124, 1996.

Anika Schumann.
Diagnosis of discrete-event systems using binary decision diagrams.
Workshop on Principles of Diagnosis (DX'04), pages 197–202, 2004.